**Review- Semandaq: A Data Quality System Based on Conditional Functional Dependencies**

Wenfei Fan, Floris Geerts, Xibei Jia

**Summary:** In today’s data driven world, obtaining clean and consistent data is a tedious task. Hence, there is need of a automated system that could identify errors and clean the data. The discussed paper addresses the issue by concepting Semendaq (Semantic Data Quality), a prototype system for improving the quality of relational data. Semandaq uses Conditional Functional Dependencies (CFDs) to detect inconsistencies and errors.

**Introduction:** Semandaq is a constraint-based system for data cleaning that introduces an interface to specify Conditional Functional Dependencies and helps detect violations in relational data with the use of SQL. With the help of CFDs, Semandaq reduces the human effort by automating the data cleaning process.

**Literature Review:** Most of the previous systems like AJAX and Potter’s wheel were dominated by transformation-based approaches and these systems mostly support specific types of data. Furthermore, the existing constraint-based systems made inefficient selection of constraints. Semandaq uses conditional Functional dependencies as constraints for data cleaning.

**Methods:** Semandaq uses multilayer approach starting from the lowest layer formed by the relational Database. The second layer is made up of components made using Java Beans namely, constraint engine, error detector, data auditor, data cleanser, and data monitor. Third layer uses data explorer to access the functionalities of the second layer. Fourth layer uses a browser interface that uses JS and XML for interaction by user. Constraint Engine is used to manage the CFDs used to specify the consistency of the data. The users can explicitly specify the CFDs. Or they can be automatically discovered from the reference data. It leverages the use of optimizations provided by DBMS in storage and manipulation of CFDs. Error Detector takes the data in the relational database and violations of the CFDs are detected. There are 2 main kinds of violations identified, Single tuple violation: and multiple tuples. After error detection, the component assigns a value to each tuple indicating the total number of CFDs it violated and also keeps track of information about the violations. Data Auditor provides a summary of inconsistencies.

Data Cleanser, on the violated attribute, evaluates candidate repair values. The algorithm chooses a repair value that ‘minimally differs’ from the original. Data Monitor is used to consider that real world data keeps getting updated. The data monitor is used to prevent degradation of data after it has been updated. Data Explorer provides a graphical interface to the user for the functionalities mentioned above. It is designed as a rich Internet Application and is separate from the data quality servers. This provides a way to introduce human intelligence to the automated cleansing process and helps users understand the data quality issues.

**Discussion:** Although Semandaq provides automated error detection and repairs, it might not give consistent results if the conditional dependencies are inconsistent. Moreover, the authors haven’t provided any comment on the scalability of the system.

**Conclusion:** Authors successfully presented Semandaq, a partially automated system for error detection and repair mechanism using user friendly graphical interface. However, there are some issues like scalability that needs to be discussed.