A FRAMEWORK FOR DATA CLEANING IN DATA WAREHOUSES

Taoxin Peng
School of Computing, Napier University, 10 Colinton Road, Edinburgh, EH10 5DT, U.K.

Keywords: Data Cleaning, Data Quality, Data Integration, Data Warehousing.

Abstract: It is a persistent challenge to achieve a high quality of data in data warehouses. Data cleaning is a crucial task for such a challenge. To deal with this challenge, a set of methods and tools has been developed. However, there are still at least two questions needed to be answered: How to improve the efficiency while performing data cleaning? How to improve the degree of automation when performing data cleaning? This paper challenges these two questions by presenting a novel framework, which provides an approach to managing data cleaning in data warehouses by focusing on the use of data quality dimensions, and decoupling a cleaning process into several sub-processes. Initial test run of the processes in the framework demonstrates that the approach presented is efficient and scalable for data cleaning in data warehouses.

1 INTRODUCTION

The popularity of data warehouses (DWs) in recent years confirms the importance of data quality in today's business success. There are many reasons for a data warehouse project to fail. One of them is the poor data quality. It is estimated that as high as 75% of the effort spent on building a data warehouse can be attributed to back-end issues, such as readying the data and transporting it into the data warehouse. Data cleansing will absorb nearly half of that time (Atre, 1998). To overcome this problem, there are at least two questions needed to be answered: How to improve the efficiency when performing data cleaning? How to improve the degree of automation when performing data cleaning?

In past decade, many researchers have challenged the problems raised regarding the quality of data since data warehousing techniques became important in decision support information systems. Topics range from data cleaning methods and approaches (Galhardas *et al*, 2001, Hipp, Guntzer and Grimmer, 2001, Liu, Shah and Jiang, 2004, Raman and Hellerstein, 2001, Sung, Li and Sun, 2002, Winkler, 2003) to quality-oriented data warehouse design (Jarke *et al*, 1999, Mecella *et al*, 2003). Generally speaking, data cleaning deals with detecting and correcting errors and inconsistencies from data. There are several reasons for databases to have data quality problems, which include typos during data entry, missing values, uncertainty, inconsistency etc., especially when multiple data sources need to be integrated, e.g., in data warehouses. Halevy *et al* recently outlined some future challenges to data integration research in (Halevy, Rajaraman and Ordille, 2006), where they claimed that "data integration has been referred to as a problem as hard as Artificial Intelligence, maybe even harder!". To overcome data quality problems, a variety of methods has been developed. However, few efforts have been dedicated to frameworks for making data cleaning more efficient, leaving the above two questions still open.

This paper investigates current data cleaning methods, approaches, and data quality oriented data warehousing design mechanisms, and then challenge above two questions by presenting a novel framework, which provides an approach to managing data cleaning in data warehouses by focusing on integrating data quality factors into mechanisms of processing data cleaning, and decoupling the cleaning process into several sub-processes. The rest of this paper is structured as follows. Basic concepts of data quality, methods of data cleaning and special data quality issues in data warehouses are introduced in next section. The main contribution of this paper is presented in section 3 that describes a novel framework for dealing with data quality in data warehouses. An example for illustrating the proposed framework is shown in

Peng T. (2008).
A FRAMEWORK FOR DATA CLEANING IN DATA WAREHOUSES.
In *Proceedings of the Tenth International Conference on Enterprise Information Systems - DISI*, pages 473-478
DOI: 10.5220/0001706004730478

section 4. Finally, this paper is concluded and future work pointed out in section 5.

2 DATA QUALITY AND DATA CLEANING

2.1 Data Quality

There is a variety of definitions for data quality. From literature, data quality can be defined as "fitness for use", i.e., the ability of data to meet user's requirement. The nature of this definition directly implies that the concept of data quality is relative.

Data quality is a multi-dimensional concept and each of its dimensions is specifically related to a particular aspect of data such as data views, data values, and data representation. It is often represented as a set of quality dimensions (Wang, Storey and Firth, 1995), which can be refined as:

- Accuracy – conformity of the recorded value with the actual value;
- Timeliness – the recorded value is not out of date;
- Completeness – all values for a certain variable are recorded;
- Consistency – the representation of data is uniform in all cases.

Recall the second question raised in Introduction: How to improve the automation when perform data cleaning? Two of the dimensions discussed above will be concerned: timeliness and consistency. Temporal information represents change. Data in a database is static, but the real world keeps changing. In general, the temporal dimension plays an important role in ensuring high quality of data. Whether the representation of data is consistent or not is another concern. For instant, to represent a property's (flat) address, there are three options: 10/3, 2F1 10 or Flat 3 10. All these three representations share the same address: the third flat at property number 10. If a customer uses version 2F1 10 to register in a bank, it is very likely that this customer will be treated as two different customers when the customer uses 10/3 to check his/her credit. This is a popular example for inconsistency. In section 4, an example is used to show a way of using such data quality dimensions in the proposed framework.

2.2 Data Cleaning

Data cleaning is required whenever there is any level of doubt existing. The aim of data cleaning is to detect, correct errors and inconsistencies from data. It is proved to be a difficult but unavoidable task for any information system. Several researches have been done to investigate, examine problems, methods, and approaches to data cleaning (Muller and Freytag, 2003, Rahm and Do, 2000).

Ideally, the processes of detecting, correcting should be performed automatically. However, it is known that fully automatically performing data cleaning is nearly impossible in most of cases. Therefore, declarative, semi-automatic approaches are feasible and acceptable for developing data cleaning tools. To actually perform data cleaning, for each type of data errors, an appropriate data cleaning method (approach) need to be selected and then applied. Choosing such a method is proved to be a difficult task (Luebbers, Grimmer and Jarke, 2003). It depends on several factors, such as the problem domain, the nature of errors, etc. If there is more than one method involved, how to apply them, i.e., in which order makes difference. To develop an effective data cleaning tool, it is therefore necessary to allow users to select appropriate methods for specified domains. It should also provide a mechanism to allow users to decide an appropriate order in which the selected methods are performed. These tasks need experienced technical and business experts.

2.3 Special Issues

A DW is a repository storing integrated data for efficient querying and analysis. It needs a set of tools for extracting data from multiple operational data sources, for cleaning, transforming and integrating the data; for loading data into the data warehouse; and for regularly updating the warehouse. This is called ETL process (Extraction, Transformation, Loading). During this process, data cleaning is typically performed in a separate data area, called Cleaning Staging Area (CSA). The resulting cleaned, standardised and integrated data are loaded as materialised views into the data warehouse.

Considering that the purpose of data warehousing is supporting decision making, the quality of data in a data warehouse is vital. Since the volume of data from multi sources is huge, and the data format might be different from one source to another, there is a high probability of errors

presenting in data while doing the unavoidable data integration.

In recent years, several research contributions have tackled the Data Warehouse Quality (DWQ) problem (Jarke *et al*, 1999, Mecella *et al*, 2003). Data quality issues discussed in these contributions are mainly focused on data warehouse quality design. However, issues on managing data cleaning, integrating data efficiently are not addressed.

Since performing data cleaning in very large databases, especially in data warehouses is costly and time consuming, it is necessary to employ techniques or approaches to, on one hand, reduce cleaning time, and on the other hand, maximise the degree of automation. The experiment done by Galhards *et al* (2001) shows that the time used for executing algorithms on a relatively large set of data is significantly long. This case will be normal if the data cleaning is for a data warehouse.

3 A FRAMEWORK

3.1 Some Definitions

Data errors are usually classified into several types. Muller and Freytag (2003) classify errors into three groups: syntactical, semantic and coverage. Each group includes a set of error types which have similar characteristics. In order to illustrate the proposed framework and the following examples explicitly, only two error types are discussed here: outliers and duplicates:

- **Outliers** – are observations that are unexpectedly different from the majority in the sample;
- **Duplicates** – are two or more tuples representing the same entity.

To separate errors particularly relevant to single-source and multi-source databases respectively, the following definitions are needed:

- **Single-source Errors.** An error of this type is present in single source databases, such as misspelling, invalid, out of dated, outliers, etc.;
- **Multi-source Errors.** An error of this type is present when data from more than one data source is integrated, such as inconsistent data, duplicates, etc.

Also from the computational point of view, the following two definitions are made:

- **Computational-costly Errors.** an error of this type costs time significantly when detecting it, such as duplicates. Computational-costly is a relative measure. In practice, this measurement should be determined by the user;
- **Non-computational-Costly Errors.** An error of this type is an opposite of the computational-costly errors, such as out of date data, outliers.

Furthermore, from the execution point of view, the following two definitions can be defined:

- **Automatic-removable Errors.** An error of this type can be detected, then removed/corrected without any human interruption;
- **Non-automatic-Removable Errors.** an error of this type can't be detected then removed/corrected without any human interruption.

3.2 The Framework

Briefly, the strategy is to break the whole data cleaning process into two stages. At the first stage, single-source errors are dealt with. At the second stage, multi-source errors are treated. At each stage, the process is further divided into two sub-processes. One deals with Automatic-removable errors, while the other deals with Non-automatic-removable errors. This strategy is more important at the second stage because it is believed that more complicated errors that are difficult to deal with in a fully automatic way present at this stage, such as inconsistencies, duplicates, etc. The proposed framework includes the following components:

1) **Auto-dealing Process for Single-source Errors.** For each data source, tasks include:
   a) Identify (not detect) all Single-source and Automatic-removable errors by the user;
   b) Select appropriate algorithms, which can be used for detecting, then removing/correcting errors identified in a) automatically;
   c) Decide an order in which the algorithms are applied, based on error types involved;
   d) Execute the algorithms in the order decided in c);
2) **Semi-auto-dealing Process for Single-source Errors.** For each data source, tasks include:
   a) Identify (not detect) all Single-source and Non-automatic-removable errors by the user;

b) Select appropriate algorithms, which can be used for detecting errors identified in a);
c) Decide an order in which the algorithms are applied, based on error types involved;
d) Execute the algorithms in the order decided in c);
e) Correct/remove detected errors;

3) **Auto-dealing Process for Multi-source Errors.** Tasks include
   a) identify (not detect) all multi-source and Automatic-removable errors by the user;
   b) Select appropriate algorithms, which can be used for detecting, then removing/correcting errors identified in a) automatically;
   c) Decide an order in which the algorithms are applied, based on error types involved;
   d) Execute the algorithms in the order decided in c);

4) **Semi-auto-dealing Process for Multi-source Errors.** Tasks include
   a) Identify (not detect) all Multi-source and Non-automatic-removable errors by the user;
   b) Select appropriate algorithms, which can be used for detecting errors identified in a);
   c) Decide an order in which the algorithms are applied, based on error types involved;
   d) Execute the algorithms in the order decided in c);
   e) Correct/remove detected errors;
5) **Transformation Process.** Tasks include
   a) Format data/schema;
   b) Integrate data;
   c) Aggregate data.

This framework can be further outlined as follows (see Figure 1): There are two main stages: one deals with single-source errors; the other deals with multi-source errors. At each stage, there are two sub-processes, *auto-dealing process* and *semi-auto-dealing process*. First data is dealt with individually, cleaning data in source databases one by one. At this stage, domain knowledge is used to help to identify possible single-source data errors presenting in each database. These errors are further classified into two groups, automatic removable errors and non-automatic-removable errors. Automatic-removable errors are considered first in the auto dealing process. These errors are detected and removed/corrected by use of appropriate algorithms, which are performed sequentially in a specified order. Basically, algorithms dealing with non-computational-costly errors are put at the front in the order. This strategy aims to minimise the processing time. It also ensures that when it's time to perform computational-costly errors, such as duplicates, the data volume should be significantly reduced. Hence, the processing time needed is reduced. In auto-dealing process, errors should be detected and removed/corrected automatically, without any interruption. Several factors may play a role in identifying such errors. Temporal factor is one of them.

After the auto-dealing process, a semi-auto-dealing process is performed. Again, possible errors need to be identified, then appropriate algorithms are selected together with a decision of an order in which the selected algorithms are performed. Similarly, the strategy of dealing with non-computational errors first applies.

As a result of this stage, the data volume is supposed to be reduced significantly.

At the second state, dealing with multi-source errors, again, auto-dealing process is performed first. Both domain and technical knowledge are needed for identifying multi-source errors. The rest is similar to the processes at stage one. Rules, strategies for selecting algorithms, deciding orders are needed. After the above two stages, the data left in the CSA should be cleaned and ready for transformation. The transformation process deals with data/schema format, data integration and data aggregation. At this stage, if it is necessary, the Semi-Auto-Dealing process or Auto-Dealing process will be triggered again. At the end, the data should be ready for loading to the DW.

There are three core characteristics in this framework. The first is the identification of data error types based on different purposes, such as time saving, automatic processing etc. There is no generic approach to do this at present. Developing such approaches remains our future work. The

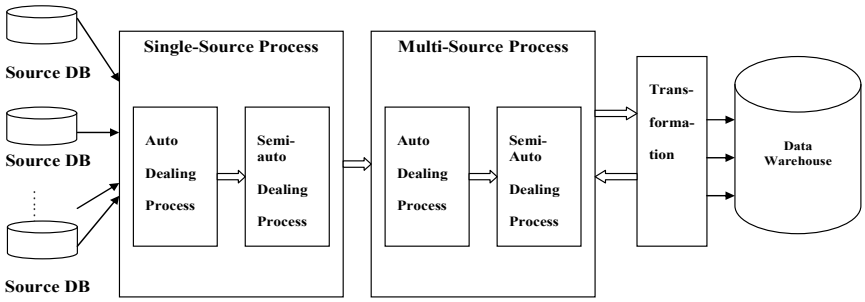

Figure 1: The cleaning process.

second core characteristic is the decision of an order for executing a set of selected algorithms. The third one is the decoupling of the data cleaning process into two processes, based on the ability of performing automatic process. These two processes will be performed at two different stages, based on single or multi data sources. The objective of this approach is to answer the two questions raised in Introduction: How to improve the efficiency when performing data cleaning? How to improve the level of automation when performing data cleaning?

4 AN EXAMPLE

To illustrate the basic idea, the framework has been prototyped and applied to some artificially designed databases, which have some of the data quality dimensions embedded. This section presents a simple example to show the efficiency of the proposed framework.

In UK, National Health Services (NHS) is a nation wide organisation, which provides health services to all residents. To simplify the description, suppose every city has a single database. The DW needs information from all cities. The scenario for this example is as follows. University students may move to other cities after they graduate. They register their doctors in a city (CityA) where their universities are located. After they graduate and find a job or a further study position in another city (CityB), it is likely that they register a doctor again in CityB, without informing their doctors in CityA. Table 1 (*CityA_Patient*) and Table 2 (*CityB_Patient*) show the samples of data in each of the two cities, where VST stands for valid start time, and VET for valid end time.

These two tables include some duplicates. As described in the proposed framework, some particular data quality factors are needed for this purpose. In this example, the temporal factor is identified as a data quality factor, which can be used to make the decision whether a possible duplicate is genuine or not. Therefore, an automatic process can be formed. By using values of attributes VST and VET, we are able to make such decisions.

It is important to notice that the use of data quality factors plays a crucial role in this framework. It helps to set rules for removals. Such factors can only be identified by business domain experts. In this example, the rule for the confirmation of duplicates is described as follows. For each pair (two tuples) of a possible duplicate, if the value of attribute Post in one of the tables is *Student,* suppose in table 1, and value of attribute VST in table 2 is about 4 years later than the VST value in table 1, a duplicate is confirmed.

Given the data quality factor, time, the rules, and the two tables, as the result of running the Auto-Dealing process, tuples 1, 2 and 4 are removed from table 1 because of duplication.

Although this is a simple example, the result shows that with the use of proper quality factors some of the detecting and removing tasks can be done as one unified process.

5 CONCLUSIONS AND FUTURE WORK

This paper has investigated current data cleaning methods, approaches and data quality oriented data warehousing design and implementation mechanisms. Based on this investigation, a novel framework has been proposed, which aims to challenge two issues: minimising the data cleaning time and improving the degree of automation in data cleaning. These two issues are closely relevant to the two questions raised in the introduction. The minimisation and improvement are achieved by introducing a mechanism of using

Table 1: CityA_Patient.

No.	L. Name	F. Name	Age	Address	Post	VST	VET
1	Cole	Liam	23	City A	Student	28-09-2003	Now
2	Gerrard	John	25	City A	Student	02-10-2001	Now
3	Small	Helen	23	City A	Student	26-09-2002	Now
4	Smith	William	24	City A	Student	01-10-2001	Now
5	Smith	Mary	28	City A	Student	12-10-2001	10-09-2005

Table 2: CityB_Patient.

No.	L. Name	F. Name	Age	Address	Post	VST	VET
1	Cole	Liam	23	City B	Engineer	20-08-2007	Now
2	Gerrard	John	25	City B	Engineer	18-09-2005	Now
3	Small	Helen	23	City B	Student	28-09-2003	Now
4	Smith	William	24	City B	Student	08-10-2005	Now
5	Smith	Kirsty	30	City B	Engineer	10-10-2005	Now

data quality dimensions and decoupling the data cleaning process into two sub-processes, base d on different purposes. This framework retains the most appealing characteristics of existing data cleaning approaches, and enjoys being able to improve the efficiency of data cleaning in data warehouse applications.

The work introduces a number of further investigations, including: a) to examine further characteristics of data quality dimensions, in order to develop a detailed guidance for determining the choice of a particular strategy for data cleaning in data warehouses; b) to develop a comprehensive data cleaning tool for data warehouses based on the framework proposed in this paper; and c) to test the framework by applying it onto bigger multi data sources. The successful outcome of such future work would certainly enhance the performance of data cleaning systems in data warehouses.

REFERENCES

Atre, S., 1998. Rules for data cleansing. *Computerworld*.

Galhardas, H., Florescu, D., Shasha, D., 2001. Declaratively Data Cleaning: Language, Model, and Algorithms. In *Proceedings of the 27th International Conference on Very Large Databases (VLDB)*, Roma, Italy.

Halevy, A., Rajaraman, A., Ordille, J., 2006. Data Integration: The Teenage Years. In *the 32nd International Conference on Very Large Databases*. Seoul, Korea.

Hipp, J., Guntzer, U., Grimmer, U., 2001. Data Quality Mining. In *the 6th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*.

Jarke, M. M., Jeusfeld, A., Quix, C., Vassiladis, P., 1999. Architecture and Quality in Data Warehouses: An Extended Repository Approach. *Information Systems*, 24(3).

Luebbers, D., Grimmer, U., Jarke, M., 2003. Systematic Development of Data Mining-Based Data Quality Tools. In *the 29th International Conference on Very Large Databases*, Berlin, Germany.

Liu, H., Shah, S., Jiang, W., 2004. On-line Outlier Detection and Data Cleaning. *Computers and Chemical Engineering* 28.

Mecella, M., Scannapieco, M., Virgillito, A., Baldoni, R., Catarci, T., Batini, C., 2003. The DAQUINCIS Broker: Querying Data and Their Quality in Cooperative Information Systems. *Journal of Data Semantics*, Vol. 1, LNCS 2800.

Muller, H., Freytag, J. C., 2003. Problems, Methods, and Challenges in Comprehensive Data Cleansing. *Technical. Report*, HUB-1B-164.

Rahm, E., Do, H., 2000. Data Cleaning: Problems and Current Approaches. *IEEE Bulletin of the Technical Committee on Data Engineering*, Vol 23, No. 4.

Raman, V., Hellerstein, J., 2001. Potter's Wheel: An Interactive Data Cleaning System. In *the* 27th *International Conference on Very Large Databases.* Roma, Italy.

Sung, S., Li, Z., Sun, P., 2002. A fast Filtering Scheme for Large Database Cleaning. In *the 11th International Conference on Information and Knowledge Management*, Virginia, USA.

Winkler, W., 2003. Data Cleaning Methods, In *the Conference SIGKDD*, Washington DC, USA.

Wang, Y., Storey, V., Firth, C., 1995. A Framework for Analysis of Data Quality Research, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 7, No. 4.