

iCleaner: A Data Cleansing Tool for Outlier Detection in a Data Warehousing Environment

**Original Research
Article**

Abstract

The implementation of Data Cleansing (DC) in Data Warehousing (DW) is essential in recent years. Organizations around the world generate huge amount of data from their day-to-day activities for their operations. These organizations will not survive if the data they generate remains *dirty* or *erroneous*. There are errors or outliers that make the data become *dirty* such as data entry errors, outdated data in the database, data migrated from old databases, and changes made at the source repository. The changing needs by customers to update their records (for example customer attributes such as marital status, phone number or address changes with time) cause records to become obsolete and reducing the quality of data. In order to obtain high quality data over time, the data requires cleansing. The proposed Integrated Cleaning (*iCleaner*) tool is developed to facilitate the data cleaning process and to address the problems associated with duplicated records. In addition, the proposed cleaning tool is able to detect and update missing data by merging key columns within the records. The system is flexible to use and comes with a convenient user-friendly interface designed for the data cleansing process. We provide an efficient, but simple algorithms designed to perform these functionalities and provide the running time for the system performance.

Keywords: Data Cleansing; Data Warehousing; Automatic Periodic Scheduler; Duplicates; Missing data; Key Column

2016 Computer Science Subject Classification: 53C25; 83C05; 57N16

1 Introduction

The advances in technological development have seen rise in businesses generating huge data from daily transactions around the globe. The recent adaptation of new tools to preserve the integrity of these huge amount of data by businesses have seen an increase in the realization of their strategic goals [Müller (2005)] and maintained higher credibility with their customers and suppliers [Wayne (2004)]. Data warehouse (DW) is a modern proven technique for handling and managing diversity in data sources, data format, and structure. Recent advances in database technologies are leading to the proliferation of different kinds of information design with independent supporting hardware and software. This technology is developed to integrate heterogeneous information sources for data analysis purposes [JerryYao, Sarrah, and Aldabbas (2012)]. DW integrates data from several operational sources (databases) all over the organizations[Bradji and Boufaida (2011)]. Potter's wheel: An interactive data cleaning system

In recent times, data collection have become a ubiquitous, large organizations collect data not only for record keeping, but to support variety of data analysis tasks that are vital in the achievement of the organization's mission and exponentially support the growth of the organization [Hellerstein (2008)]. As a result, the integrity of the data should be of much interest to stakeholders. When the data gathered is found to contain outliers, data cleansing process should be employed to deal with these outliers. Researching into various aspects of data cleansing (DC) to find procedures to automatically or semi-automatically identify and correct outliers (errors) in large datasets then become relevant. DC typically deals with the identification of corrupted data in large datasets in order to enhance its quality [Karr,Sanil and Banks (2006)] causing the data to be *dirty*. Data Cleansing is an activity performed to detect and correct the data quality problems, and inconsistencies from the dirty data[Adu-Manu and Arthur (2013)].

It is vital for organizations to manage data by constantly monitoring and cleaning datasets obtained from their databases and/or file systems in order to maintain the data quality. According to the Data Warehousing Institute (DWI), data quality problems cost businesses in United States of America over 600 billion dollars per year[Adu-Manu and Arthur (2013)]. The development of commercial data quality tools for performing data cleansing to obtain quality data have been proposed and developed by several authors [Raman and Hellerstein (2001)], [?], [?], [MLee, Ling, and Low (2000)] to solve this pertinent problem. But most of these commercial data cleansing tools are moving beyond simply auditing single specified fields to tackling multiple fields and other data types[Adu-Manu and Arthur (2013)].

It is therefore paramount for organizations to perform DC to achieve optimal performance. Developing an efficient data cleansing tool to detect and remove outliers such as duplicates, missing data (or empty fields/null fields), incomplete data, and typographical errors automatically from the organizations databases and file systems [Raman and Hellerstein (2001)] will support and increase the quality of data to enhance business solutions. These outliers in datasets (records) result in inconsistencies in data analyses and reports which affects the performance of the organization in decision making. In our previous work, we investigated some existing commercial data cleansing tools taking into account their strengths and weaknesses [Adu-Manu and Arthur (2013)], [Adu-Manu and Arthur (2013)] and we also proposed a conceptual framework which included an automatic periodic scheduler (APS) for the DC process[Adu-Manu,Davis, and Panford (2013)].

In this paper, we employ system analysis and design methodology tools to develop and test the performance of our proposed system (*iCleaner*). We obtained data from students database and used this data as input to test the systems performance. Several datasets of varied sizes (ranging from Kilobytes to Gigabytes) were used for the system testing. The research results strongly revealed that maintenance of the clean data within periodic intervals, the file format, the error type, the number of processors and memories are major factors in the data cleansing process. Two main strategies are proposed: 1) the strategy of storing all the strings resulting from the *addition of columns* in a dictionary to identify duplicates and 2) the strategy of *comparing key columns* to identify empty or null

fields. These strategies made it easy for searching through the datasets.

The rest of the paper is organized as follows. In Section 2, we provide the motivation of the paper. The background and related literature are presented in Section 3. The conceptual model of the proposed algorithms are discussed in Section 4. The implementation and testing are discussed in Section 5 and we conclude in Section 6.

2 Motivation

The concept of data warehouse and data cleansing is gaining popularity in the business environment. Companies are challenged with the increasing number of data collected from its customers and employees. Data transformed to information is relevant for decision making by management for reporting purposes and data analysis. The task of keeping large datasets clean (free from errors) and consistent by these organizations their databases turns to be a difficulty [MLee, Ling, and Low (2000)]. But this alone does not ensure that all the managerial activities or functions and decisions could be smoothly undertaken. There is a compulsive requirement for the data to be meaningful or, in other words, data quality is of utmost importance if management is to take any advantage of the data at its disposal.

Data collected from different sections or units in organizations representing any such entity as name, date employed, height, weight etc are collected and processed for management to keep track of customers, and employees, and track the volume of data from its numerous branches, analyze and interpret the data to support decision making processes, and support the business process.

During data collection, several errors occur. Among these are omissions, data entry problems and double entry problems rendering databases redundant. The data residing in operational databases, though perhaps of sufficient quality for transactional procession, contain duplications, inconsistencies, errors, missing data, and other issues that make them unsuitable for a decision support data warehouse, without considerable cleansing of the data [Mohan, Wilshire, and Schroeder (1998)], [MLee, Ling, and Low (2000)], [Kimball (1996)]. The researchers are motivated to undertake this research so as to develop a data cleansing tool to enable companies, government agencies, educational institutions and other related agencies that collect and process huge data for reporting and analysis to get rid of some common errors.

3 Background and Related Literature

Most organizations are faced with the data quality problems and spend huge amount of money on poor data annually. It is expensive to deal with poor data especially when reporting on the company's performance over a period of time. It also affects the organization's business decisions and customer relationship management [Hammad and Jihad (2011)]. Data tends to be the core business asset that needs to be managed with optimal quality if an organization is to generate a return from it.

Several works have attempted to solve the problems related to poor data being transferred for storage in a data warehouse. There are many issues in DC that researchers have tackled using different tools and approaches. Of particular interest here, is the search context for what is called in literature and the business world as *dirty data* [Nemani and Konda (2009)], [Vassiliadis, Vagena, Skiadopoulos, Karayannidis, and Sellis (2001)], [Mallach (2000)]. Recently, [Vassiliadis, (2003)] proposed taxonomy for *dirty data*. The search context is an important issue which have attracted the attention of researchers and practitioners in the field. It is the first step in defining and understanding the data cleansing process. *Dirty data* is caused as a result of misuse of abbreviations, data entry problems, control information hiding, missing fields, spellings, outdated code [MLee, Ling, and Low (2000)], [Kimball (1996)].

Data cleaning addresses the issues of detecting and removing errors and inconsistencies from data in order to improve the quality of the data. In general, the architecture of DC process involves five phases: 1) Data Analysis in order to detect the inconsistencies; 2) Definition and choice of transformations; 3) Verification of the correctness and effectiveness of transformations; 4) Execution of the transformation; 5) Feedback of cleaned data [Vassiliadis (2007)], [Hernandez, and Stolfo (1998)].

In data warehousing, data cleaning becomes relevant when data is drawn from several databases to be merged at one point. There are a number of ways to represent or refer to the same entity in different data formats. Erroneous data in databases can be reported and captured in a data warehouse. Data cleaning have three components: auditing data to find discrepancies, choosing transformations to fix the problems, and applying transformations on the dataset [Naumann (2002)]. They come in two forms: auditing tools and transformation tools. The data often have many hard-to-find special cases, so this process of auditing and transformation must be repeated until the *data quality* is good enough. This approach has two problems – lack of interactivity and need for much user efforts [Naumann (2002)].

3.1 High Quality Data

Data can be described as high quality when it satisfies a set of criteria. In general, quality is defined as an aggregated value over a set of quality criteria [Wang, Storey, and Firth (1995)]. High quality of data warehouse is a key to make smart strategic decisions. Hence, correctness of data is essential for well-informed and reliable decision making [Galhardas, Florescu, Shasha, and Simon (2000)]. The data cleaning is a program that performs to deal with the quality problems of data extracted from operational sources before their loading into data warehouse [Hernandez, and Stolfo (1998)]. Data quality is data that is fit for use by data consumers ?. Data cleaning addresses the issues of detecting and removing errors and inconsistencies from data in order to improve the quality of the data. Data quality has many attributes [Lin and Haug (2008)]. An aggregated value over the criteria of integrity, consistency and density is referred to as accuracy. Intuitively this describes an exact, uniform and complete representation of data collection not containing any defined anomalies except for duplicates [Bradji and Boufaida (2011)], [Kimball (1996)].

3.2 Data Cleaning Work Flow

Data cleansing work flow follows a predefined process from the beginning to completion to achieve a specific task [Jonathan and Marcus (2000)]. The cleaning process is complex and begins when the data is imported into the application for example, Potter's Wheel [Raman and Hellerstein (2001)], AJAX [Galhardas, Florescu, Shasha, and Simon (2000)], ARKTOS [Vassiliadis, Vagena, Skiadopoulos, Karayannidis, and Sellis (2001)], and IntelliClean [MLee, Ling, and Low (2000)]. In some cases, the data cleaning tools listed above are unable to remove completely all the anomalies and therefore the user involvement in the data cleaning process cannot be overlooked. The user has the choice of selecting the type of error within the database to work on and to verify that he is satisfied with the cleaning process. The user can then push the clean data, void of duplicates and/or missing data items into a separate window to complete the verification process. The clean data is then exported to another file.

In DC, statistical methods are used for auditing the data to detect some form of data anomalies and contradictions [Rahm and Do (2000)], [Müller (2005)]. The process of auditing the data supports the achievement of high quality data as the end product [Müller (2005)], [Galhardas, Florescu, Shasha, and Simon (2000)] which is useful, relevant, accurate, and concise. A proper workflow can be used as an alternative solution for common errors such as typographical errors [Rahm and Do (2000)] which can be corrected, for example by considering the layout of the keyboard. Hence, whenever the data cleaning process or workflow fails to correct errors automatically, then errors are corrected by the user manually [Müller (2005)].

4 Data Cleansing Design

The conceptual framework in ?, illustrates how data is extracted, cleaned and transformed and loaded from source computers into the central database. The design of our proposed system takes into consideration the file format, the user interactivity, the cleaning process, and maintenance.

4.1 Design and Model

In this section, we present the design of the proposed data cleaning tool. As part of our discussion, we provide the flowchart and the framework for carrying out the DC process. The conceptual model shown in Figure 5, illustrates how data is extracted, cleansed, transformed, and loaded from the source computers into a central database. When the data is loaded into the data cleansing tool

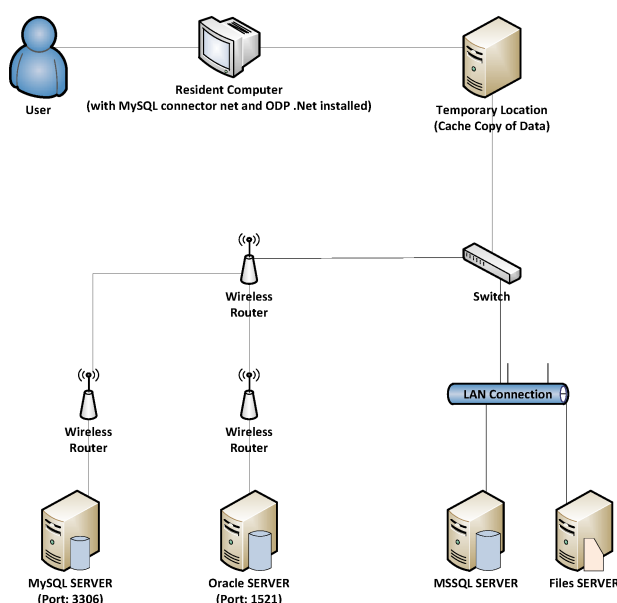


Figure 1: Data Cleansing Framework

iCleaner, the following steps are performed: 1) A cache copy of the data is created temporarily in the system. 2) The type of error to remove is selected by the user (e.g. duplicate or missing data). 3) When *duplicate* is selected, then the *iCleaner* system will scan through to detect all duplicates by adding all the columns in each rows to obtain a unique value referred to as the *key*. 4) When the error type selected is *missing data*, then the rows are updated accordingly by comparing rows based on key columns. 5) The cleaned data is validated by the user before uploading the data into the desired location on a secondary storage device or the warehouse, otherwise the data is sent back for further cleansing.

4.2 Detecting duplicates - Addition of Columns

This section describes the *addition of column* method used for the identification of duplicated data. To achieve this, *the individual columns in each row are added to obtain a single value*. The system

is designed to search through the datasets using the binary search method. This search method is used in order to obtain optimal performance and response time. In order to detect duplicates in any data set, all the key columns are concatenated (or added) together and converted into a single string. It is imperative to note that there is a consistency among columns within the various tables in a database. In the method of detecting duplicates, each row that has been concatenated or added generates a single value (this is a string).

Table 1: Sample Data of Students Records

Student ID	Title	Last Name	First Name	Other Name	Gender	DOB	POB
2125600	Miss	Adu	Christiana	NULL	Female	6/30/1981	Tema
2125620	Miss	Agbeko	Mavis	NULL	Female	NULL	Kumasi
2125631	Mrs	Afrifa	Yvette	Akosua	Female	10/25/1987	Accra
2125642	Mr	Arthur	John	Kingsley	Male	3/14/1993	Mankesim
2125633	Mr	Ofori	Amanfo	Emma	Male	1/12/1991	Sefwi Wiaso
2125624	Miss	Aidoo	Patience	NULL	Female	5/15/1978	Tarkwa
2125627	Miss	Akafia	Lawrencia	NULL	Female	6/4/1980	Accra
2125677	Mrs	Okoe	Theodora	NULL	Female	7/14/1984	Sandema
2125648	Mr	Ampofo	David	NULL	Male	NULL	Navrongo
2125611	Mr	Poku	Kwame	Nana	Male	11/13/1984	Tamale
2125610	Miss	Opoku	Berlinda		Female	9/23/1991	Wenchi
2125688	Mr	Danso	NULL	Prosper	Male	12/10/1982	Koforidua

From Table 2, concatenating the fourth row (i.e. adding each value in the column; here each value represents any data type found within the column), the following string is generated *2125642MrArthurJohnKingsleyMale3/14/1993Mankesim*. From Table 2, after the string has been generated, each of these strings are stored in a single value dictionary and they are compared with the rest of the generated strings in the dictionary to detect all occurrences of the same string (i.e. duplicates). Each value item is used in the comparison process. During the comparison of the strings

, each single value or string that does not recur in the list is considered as unique and it is stored in a hash table (unique key dictionary).

A unique key is defined as the single value or string that has no recurrent (repeated) value in the single key dictionary. When the first string is compared to the other strings and a match is found then that string value is considered duplicated and this string value is stored in non-unique key dictionary. It should be noted that every key in the dictionary acts as a pointer to the record within the rows fetched from the main table. To identify duplicates, we define the error instance as E , the dataset is represented as D , each row instance is denoted by R , where $R=\{d_1, d_2, d_3, \dots, d_i, \dots, d_n\}$ and where d_i is the i^{th} data value. Also, each record is denoted as F , where $F=\{f_1, f_2, f_3, \dots, f_i, \dots, f_n\}$ and where f_i is the i^{th} field and f_1 is a subset D . Each table is represented by T with r records, where The value of d_i is also a subset of T . We also define C to represent the summation of columns, where C represents duplicated or unique values, S_d denotes all duplicated rows and S_u all unique rows representing the variables to store differences between two columns C_k and C_{k+1} . We compute S_d by assuming that $k=0$ and $i=0$, then $S_d = C_k - C_{k+1} = 0$, this results in the detection of duplicated records. Hence, we calculate duplicate as, $dup = \sum_{k=1}^n C_k - C_{k+1} = 0$, whereas unique values are determined as follows: $S_u = C_k - C_{k+1} \neq 0$ and $uq = \sum_{k=1}^n C_k - C_{k+1} \neq 0$ to obtain unique keys.

On the other hand, to determine duplicated rows we concatenate all the columns within a particular row. The string value obtained should not be equal to the addition/concatenated value of any other key value within table with n number of rows. If this occur then that record is a duplicated record. Assuming $k = 0$, $key = ""$, $m = 0$, and $i = 0$, where key is the unique identifier, m is the variable holding the concatenated values of all fields in a specified row, i is the value of each field and R_n is the last row within the dataset. We determine the duplicated data as shown in Table 2.

Table 2: Searching for Duplicates

Algorithm 1
<pre> while (k = R_n) { $m = R_k \times F_n + R_k \times F_{i+1} + \dots + R_k \times F_n$ $m = \sum_{k=0}^n \times \sum_{i=1}^n R_k \times F_{i-1}$ if (key.contains (m)) { println("Yes, duplicate found, clean error instance E") } else { println("No, duplicate not found, ignore error instance E") key = key + m } k = k+1 } </pre>

4.3 Detecting Missing Data—key columns

To detect missing data, the system specifically identifies fields with *Null values and Blanks*. This method used to detect missing data (that is, an empty field) is based on the key columns within the rows. The key columns in this case are determined by the user.

Although in this paper key column is explained as a unique field required in the record as shown in Figure 4, the system/tool is designed to allow the user to have privileges in selecting and determining the key columns to use as the search criteria. In searching for missing data, we propose a technique where in each row the key columns are verified for *blanks or NULL*.

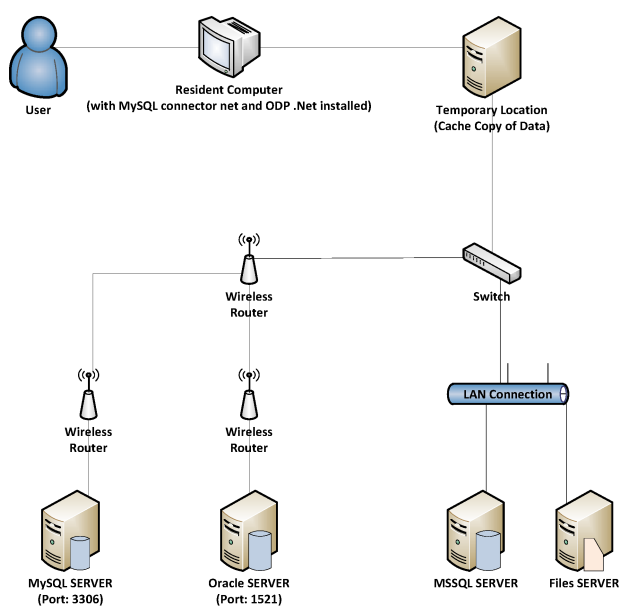


Figure 2: Architecture

In this case, two pointers are distinctively relevant— the blank or null column pointers. To this point, the missing data is identified in a row when one of the column data is not available or present. The algorithm identifies particular columns of interest referred to as *key columns*. For example, in Table 2 there column with heading *OtherName* is not considered a key column because not all persons have other names but considering columns such as *FirstName*, *LastName*, *DOB* and *POB*, such fields are required attributes about every students. This is further illustrated in Figure 4. In order to detect any missing fields in a particular data set, the following procedure was experimented. A key column in a row that requires verification is selected by the user for *null values or blank spaces*. The system then performs a search in the entire database for the selected key columns. These key columns are compared to all the fields in the database.

When a related field is found in the comparison operation of the key columns with the other columns, the system automatically updates the field with the new data value and merge the two fields. On the other hand, if no such record is found the system maintains the null value or empty fields and imports the row to a new table for verification by the user before pushing it into the DW. Finally, when the system searches the database and the key columns selected is not equal to null, then there exist no empty field. We denote d_a and F_i as row and field respectively. We determine the missing data as shown in Table 3.

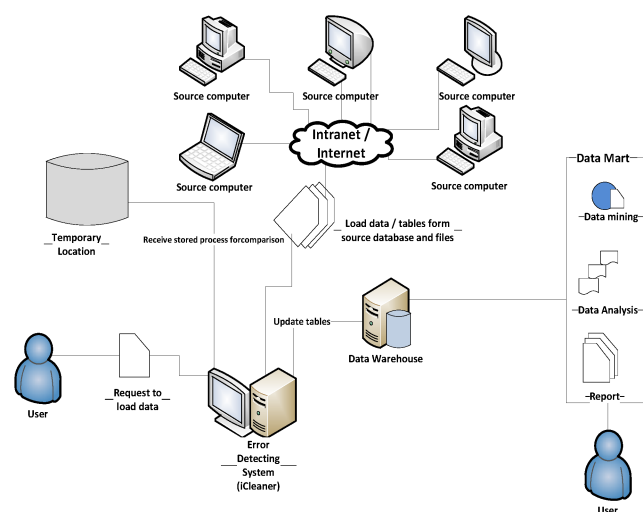


Figure 3: System model

LastName	FirstName	OtherName
Adjetey	Sarah	Adjorkor
Adje	Ivy	NULL
Afrifa	Linda	NULL
Agbeko	Mary	NULL
Agbo	Joshua	Elkplim
Agyare	Yvonne	Adjepong
Agyei	Paa Kwesi	P.
Agyei-Baah	John	NULL

Key columns

Not a key column

Figure 4: Key column identification

4.4 System Complexity

The process of detecting and eliminating approximated duplicated records and missing data from operational database into a Data Warehouse (DW) and applying the Data Cleansing (DC) technique is a very complicated task. In order to identify and remove duplicated records and missing data from different sources, the column addition technique using binary search tree and pair-wise strategy for comparing various columns is proposed. The proposed system is expected to generate records from the appropriate tables in the databases 100% of the time. The speed of the proposed system will depend on the hardware requirement rather than the systems characteristics. The proposed algorithm exhibited a quadratic time or order n^2 . Hence, the running time of the algorithm is $\mathcal{O}(n^2)$. This is so because the running time of the algorithm depends much on the input size of the dataset that is loaded into the system (*iCleaner*) to perform the cleansing process. The algorithm's running time does not depends data loaded into the cache copy but during the cleansing process.

The proposed system can run with the minimum hardware requirement but comparatively performs better on hardware with good specifications. The algorithm's space complexity is essentially related to the memory size. After running the system several times with different input sizes, we realized that the system does not uses so much of memory to process data because the data is loaded into cache.

When the input size is very huge it still uses a reasonable amount of memory and processing time for the cleansing operation to complete.

Table 3: Searching for Missing Data

Algorithm 2
<pre> for a = 0 to D { for i = 1 to n { if ($d_a \times F_i$) = NULL { return true println(Missing data found in the row) } else { println(No Missing data found in the row) } } } </pre>

5 Implementation

The cleaning tool presented in this paper will benefit organizations, institutions, and companies in the following ways: 1) increase the accuracy and consistency of records; 2) facilitate proper data analysis and 3) increase the integrity of the organization's data for reporting and strategic planning. The proposed data cleansing tool is easy to use, saves a great of time, and allows data to be imported and exported within the shortest possible time. In order to develop the tool, a series of steps were followed to design the system flowchart as shown in Figure 6. The flowchart depicts the flow of operations in the data cleansing model. The flowcharts begins by accepting the type of error and ends with the user validation and then terminates. The system flowchart serves as the roadmap for the data cleansing tool because it is showing how all the major components interact and fit together.

5.1 Deployment

The implementation of the integrated algorithm for data cleansing is achieved through the coding of the pseudo-code algorithm inferred from Figure 6. The proposed system was developed using C-sharp programming language. The Integrated Cleaner (iCleaner) is a powerful data cleansing tool developed to clean and correct duplicates and missing data. The proposed system is capable of cleaning data retrieved from Database and File Servers such as MSSQL, MySQL, Oracle, and Text files with these extensions: .txt and .csv, and Excel files with the following extensions: .xls, .xlsx,

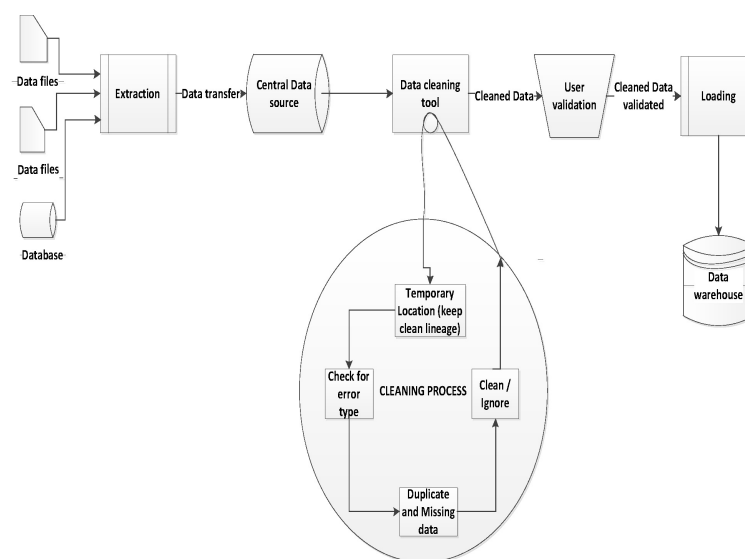


Figure 5: Data Cleansing Framework

xls, and xlsb as illustrated in Figure 2. Once you are able to obtain the necessary credentials (server name/instance, user name, and password) for the relational databases, (MSSQL, MySQL, and Oracle) using the system model in Figure 3, the user will be able to access all the databases retrieved from the Internet or Intranet, and the access all tables within the databases. The data is then imported into the *iCleaner* for onward cleansing. The clean data is exported to the Data Warehousing system.

5.2 Graphical User Interface (GUI)

User interface design requires easy usage of all the component parts of the system and should allow for effective use of the system. The user interface is intuitive as compared to other tools in the market. The GUI shows the various procedures the user must implement to clean data after loading the data from different operational sources. Users begin by selecting the type of database or file type, enter the required server name, username, password in order to load the data for the data cleansing process. Figure 7 shows the process a user goes through to enter database credentials, (for example at the data source enter *192.128.15.10*, User name enter *schoolMSsystem*, and Password enter *passw0rd*, and then select the table to work on). This data is obtained from MSSQL database. The the user clicks on the *Test Connection button*, to load a cache copy of the data into a temporary location for the cleansing process to begin. Then afterwards, the type of error to work on is selected. Figure 8 illustrates this next step in the data cleansing process. The imported data is cached so that the original data coming from the operational source is not tempered with.

5.3 System Testing and Performance Analysis

We tested *iCleaner* using real datasets from a student database. The database had over 4000 records and a sample table is shown in Figure 1. The results are discussed in the subsections that follow.

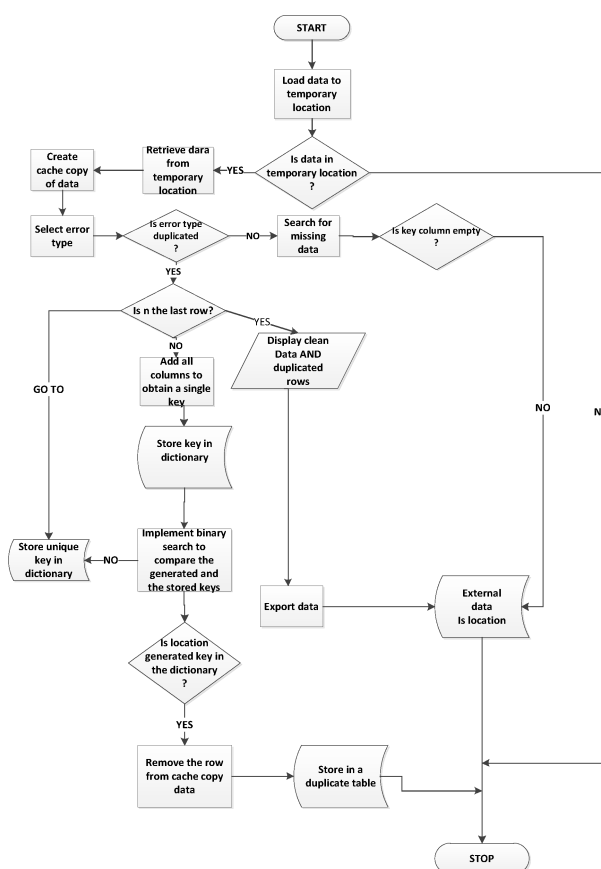


Figure 6: Flowchart representation

5.3.1 Duplicates

In order for the tool developed to meet its expected performance for detecting and cleansing duplicates, the cleansing tool will take advantage of the advances in processor and memory development in recent times and in the near future. The system was tested with a minimum data size of 5GB and a maximum data size of 30 GB. It was tested on a computer system with a single processor, and multiple processors both with a 64 MB of memory. It is platform independent. The performance of the system greatly improve on devices with multiple processing power as shown in Figure 9. This characteristic is known as the scale-up factor, i.e. the amount of speed or the throughput gained by the application depend on the systems performance, which is a key factor. It is fast and more reliable as compared to Winpure clean and match data cleaning tool ?.

5.3.2 Missing Data

With respect to missing data set, the data cleaning or cleansing tool relies on the *key columns* selected by the user. These columns are used as the search criteria during the data cleansing process. At this stage, all the columns in the table can be used as part of the search criteria or single

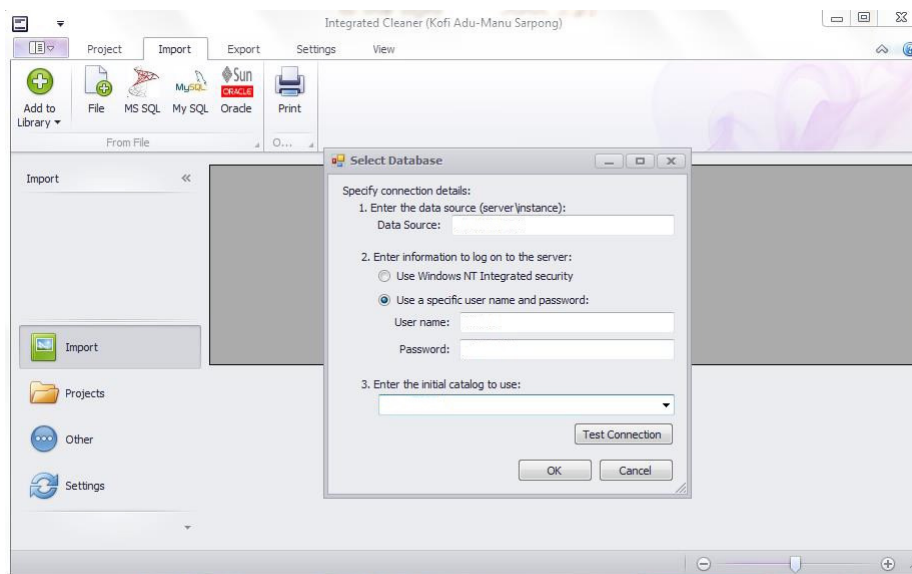


Figure 7: Interface for connecting to a database

field or multiple fields can be used as the criteria for searching for records with missing fields. There is tradeoff between processing time and memory usage when all fields are used as the search criteria.

Therefore, the best policy will be to use the maximum number of relevant fields to accomplish the desired results. Figure 10 shows that the cleansing or cleaning time achieved in identifying missing data is far less than the time for cleaning duplicates. When performing data cleaning or cleansing to remove duplicates, all the columns are factored into the searching criteria. In the case of identifying missing data fields there is an option for the user to make his choice in discovering missing records. Figure 10 shows the running time for the data cleansing tool/system in identifying missing follows quadratic function which is $\mathcal{O}(n^2)$.

6 CONCLUSIONS

We proposed a data cleansing tool to remove duplicates and missing data records from a databases and files by implementing a simple but efficient algorithm. In addition the system is designed with high flexibility and can be scaled to add other errors by integrating it in the programming logic due to the usage of object-oriented programming tools. The introduction of duplicated records in files and databases, missing datasets, and other errors have adverse effects on strategic decision making by business owners and the company as a whole. Research has shown that *dirty* data causes a lot of inconsistencies in data reporting and analyzes which causes the organizations to lose goodwill, money, and also breakdown customer relationships and satisfaction. Several research works have been done in the area of data cleansing in the past to overcome the inconsistencies that arise as a result of erroneous data. We have provided a new approach to efficiently perform data cleansing with a simple tool (iCleaner) with enhanced GUI. The algorithms presented and explored in this paper relates to error detection and data cleansing by generating a key and storing it in a dictionary for use during the searching of these records. The designed tool (iCleaner) worked effectively to obtain quality results.

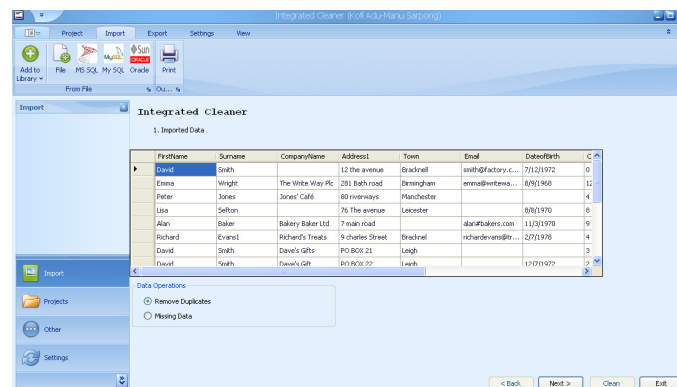


Figure 8: Interface for cleansing Error

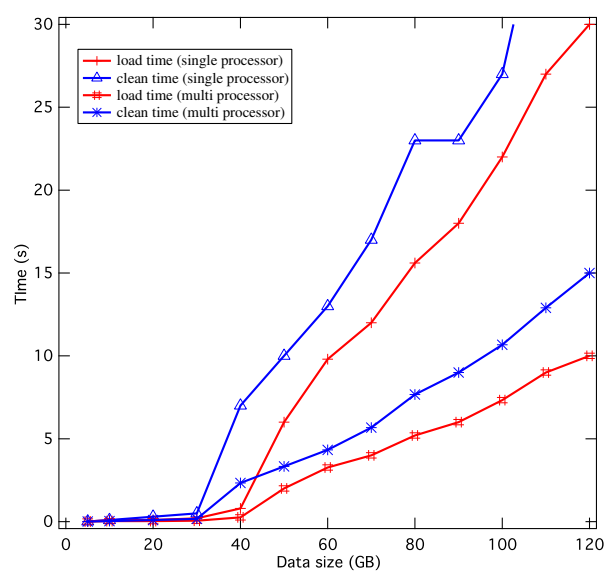


Figure 9: System testing to remove duplicates

References

- Adu-Manu, K-S. and Arthur, J-K. (2013). Analysis of Data Cleansing Approaches regarding Dirty Data-A Comparative Study. International Journal of Computer Applications.
- Adu-Manu, K-S. and Arthur, J-K., 2013. A Review of Data Cleansing Concepts-Achievable Goals and Limitations.International Journal of Computer Applications.
- Adu-Manu, K-S., Davis,J-S., and Panford, J-K. A Conceptual Framework for Data Cleansing-A Novel Approach to Support the Cleansing Process.International Journal of Computer Applications.

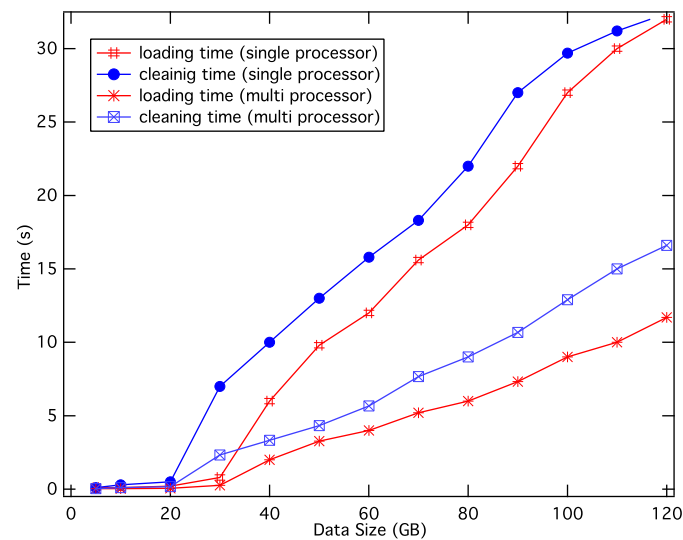


Figure 10: System testing to remove missing data

- Bradji, L. and Boufaida, M. (2011). Open User Involvement in Data Cleaning for Data Warehouse Quality. *International Journal of Digital Information and Wireless Communications*.pp.536-544.
- Galhardas, H., Florescu, D., Shasha, D. and Simon, E. (2000). Ajax: An Extensible Cleaning Tool. In: *ACM Sigmod Record* Vol. 29. pp. 590.
- Hamad, M-M. and Jihad, A-A.(2011).An Enhanced Technique to Clean Data in the Data Warehouse. In *Developments in E-systems Engineering*.IEEE.pp. 306-311.
- Hellerstein, J-M. (2008).Quantitative data cleaning for large databases. United Nations Economic Commission for Europe (UNECE).
- Herbert, K-G. and Wang, J-T.(2007). Biological data cleaning: a case study.*International Journal of Information Quality*.pp.60-82.
- Hernndez, M-A. and Stolfo, S-J.(1998). Real-world data is dirty: Data cleansing and the merge/purge problem.*Data mining and knowledge discover*. pp.9-37.
- JerryYao, D., Sarrah, M., and Aldabbas, H. (2012). Three Tier level Data Warehouse Architecture for Ghanaian Petroleum Industry . *International Journal of Database Management Systems (IJDMS)* Vol, 4.
- Jonathan, I. and Marcus, M-A.(2000). Data Cleaning: Beyond Integrity Analysis. *Division of Computer Science*.
- Karr, A-F., Sanil, A-P. and Banks, D-L. (2006). Data quality: A statistical perspective. *Statistical Methodology*. pp.137-173.
- Kimball, R. (1996). Dealing with dirty data. *DBMS*.

- Kim, W., Choi, B-J., Hong, E-K., Kim, S-K. and Lee, D.(2003). A taxonomy of dirty data. Data mining and knowledge discovery. pp.81-99.
- Lin, J-H. and Haug, P-J.(2008). Exploiting missing clinical data in Bayesian network modeling for predicting medical problems.Journal of biomedical informatics.pp.1-14.
- Mallach, E.(2000). Decision support and data warehouse systems. Irwin/McGraw-Hill.
- Mller, H. and Freytag, J -C. (2005). Problems, methods, and challenges in comprehensive data cleansing. Professoren des Inst. Fr Informatik..
- MLee, M.L., Ling, T.W. and Low, W.L. (2000). IntelliClean: a knowledge-based intelligent data cleaner. ACM. In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 290-294.
- Mohan, S., Willshire, M.J. and Schroeder, C.(1998). DataBryte: A Proposed Data Warehouse Cleansing Framework. In IQ. pp. 283-291.
- Nemani, R-R. and Konda, R.(2009).A framework for data quality in data warehousing. In International United Information Systems Conference. Springer Berlin Heidelberg.pp. 292-297.
- Naumann, F.(2002). Quality-driven query answering for integrated information systems (Vol. 2261). Springer Science and Business Media.
- Raman, V. and Hellerstein, J-M (2001). Potter's wheel: An interactive data cleaning system.VLDB. pp. 381-390.
- Rahm, E. and Do, H-H.(2000). Data cleaning: Problems and current approaches. IEEE Data Eng. Bull.pp.3-13.
- Vassiliadis, P., Vagena, Z., Skiadopoulos, S., Karayannidis, N. and Sellis, T. (2001). ARKTOS: towards the modeling, design, control and execution of ETL processes. Information Systems Volume 26, Issue 8. pp. 537-561.
- Wayne, W-E. (2004).Data quality and the bottom line: achieving business success through a commitment to high quality data. TDWI report.
- Wang, R-Y., Storey, V-C., and Firth, C-P.(1995). A framework for analysis of data quality research. IEEE transactions on knowledge and data engineering.pp.623-640.
- Yano, K. and Kon, M.(1994). Structures on Manifolds. World Scientific, Singapore, Series in pure mathematics, 3.