**Data Wrangling: Development of a customized ETL tool**

**By: Vijay Shanker Karingula**

A creative component submitted to the graduate faculty

in the partial fulfillment of the requirements for the degree of

**Master of Science**

Major: Information Systems

Program of Study Committee:

Dr. Brian Mennecke

                                                              Dr. Sree Nilakanta

 Dr. Steven B Cannon

**CONTENTS**

**Abstract……………………………………………………………………………………………………..3**

**Introduction……………………………………………………………………………………………….3**

**Approach…………………………………………………………………………………………………….4**

* **Tech stack used………………………………………………………………………………..4**
* **Setting the environment…………………………………………………………………..5**
* **Data source and Extraction……………………………………………………………….8**
* **Database Schema Design………………………………………………………………….9**
* **Data Cleaning and Transformation……………………………………………………10**
* **Data Loading…………………………………………………………………………………….12**

**Challenges………………………………………………………………….………………………………..12**

**Discussion…………………………………………………………………………………………………….14**

**Limitations……………………………………………………………………………………………………18**

**Future Research…………………………………………………………………………………………….18**

**Conclusion…………………………………………………………………………………………………….19**

**References…………………………………………………………………………………………………….20**

**Abstract**

No matter how advanced today’s technology is, still lots of efforts and resources are being spent on fixing data quality issues. While it is understood that data by itself is not important but what insights and visualizations the data is forecasting is important. For data analysis, making the data ready is crucial so that it can be consumed for analysis and most of the times this requires cleaning and reformatting the data to prepare it for subsequent analysis. Through this project we propose the development of an ETL (Extract-Transform-Load) tool with user interface and logical optimization of traditional ETL process. Then we discuss enhancements that can be made to the current model in future through research. Finally we conclude by discussing how the data analyst community can leverage the current application for data analysis.

**Introduction**

In the today’s fast paced Internet world, petabytes of data is being generated at a faster rate than it can be processed and understood. The sources of data generation are spanned over diverse fields and industries like social media, agriculture, finance, real estate, oil and natural gas, health care, technology, education, transportation, airlines, government and many more. Real-time data is being generated for sure in every activity by businesses and researches. This data is very important to analyze for further improvements and advancements in businesses and researches. Analyzing data appropriately will enable data visualization. Data visualization will in turn help understand the patterns and trends.  But unfortunately, the thus generated data is in a very crude form and hence analyzing the data is not possible right away in a single step. So even before analyzing the data, analysts need to take this raw data and perform data diagnosis and data cleaning for data quality issues and then manipulate and transform data into usable form. This is where the job of data scientists comes into picture. Data Scientists spend 80% of their time to convert data into usable form [1] and as an intermediate step lot of data wrangling is needed. In this project one such method of data wrangling is demonstrated where the raw data is processed, cleaned, verified and loaded into a database so that the meaningful data is ready in the database to be consumed for analysis and visualization.

**What is Data Wrangling?**

*Data Wrangling is loosely the process of manually converting or mapping data from one raw form into another format that allows more convenient consumption of the data with the help of semi-automated tools [2].*

**Approach**

**Tech Stack used:**

In this project, the technologies used are as follows—

Perl is used as a primary programming language. HTML and JavaScript for creating front-end web pages. PostgreSQL, the database for storing appropriate end data. Initial raw data is extracted into Microsoft Excel sheets. So, the data in Excel sheets is the primary source of data for data wrangling in this project. Komodo Edit is a powerful editor which is used for writing the source code easily. Apache HTTP server is used as a web server software, Mac OSX is the operating system platform used to develop this project. Command line interpreter is used because this project can be executed not only through a web interface but also through the terminal. Shell Scripting (bash) to pipeline the execution of different perl files through a single bash script. Git is used for version control.

**Setting the Environment:**

Before even proceeding with the development, setting an environment is important and often can be tricky. For this project, following are the steps done to set an environment-

1. Mac OS X ‘Yosemite’ (10.10) by default has an apache installed in it. Just it needs to be started and used.
2. When a url “*http://localhost”* is given in a browser in OS X, it should display the message “It Works!”. This is an indication that apache is running in the system appropriately. But how is this message displayed? By default when the url is given, a default html script is executed. “It Works!” message is displayed because it is written so in default html script. So we need to make the control point to our custom prepared html page instead of default html script.
3. Generally, the default html location in the system can be known by executing the command in the terminal --  *ls /Library/WebServer/Documents/*
4. How to make the control point to our custom prepared html page and execute when an url *http://localhost* is given in the browser? Go to /etc/apache2 folder and open the configuration file httpd.conf. This file needs to be edited but cannot be edited through normal editors because httpd.conf is a root-owned file. A command-line editor such as nano can be used as -- *sudo nano /etc/apache2/httpd.conf*.
5. Now when the httpd.conf file opens, change the line “*Document Root “/Library/WebServer/Documents* to “*Document Root /Users/<your\_user>/Sites*. If Sites folder is not present, then one should be created and custom prepared html file should be present in Sites folder.
6. Next, the <Directory> tag should also be changed to point to new html file --

*<Directory “/Users/<your\_user>/Sites”>*

   In the same <Directory> block, the ‘AllowOverride’ setting should be changed as *AllowOverride All*

g. After the above steps, the Apache configuration is pointing to Sites folder in home directory. But one problem still exists. Apache by default runs as user *‘\_www’* and group *‘\_www’* which causes permission problems when trying to access files in the home directory. To fix this problem, the default values for user and group must be changed in httpd.conf configuration file. Replace the user value from *‘\_www’* to *‘<your\_user>’* and group value from *\_’www’* to *‘staff’*.

h. Next step would be creating a custom html file in Sites folder which would be the starting page of the process in this project. Now pointing the browser to *http://localhost* should display the output of custom defined html file in Sites folder.

i. If for some reason the contents of html file are displayed instead of output, then some more settings are needed to be done in httpd.conf file. AddHandler in httpd.conf to support perl and cgi files as shown below -

*AddHandler cgi-script cgi pl*

*<Directory /path/to/cgi/files>*

*options +ExecCGI*

*</Directory>*

j. Always make sure that Common Gateway Interface (CGI) scripts are loading. This can be ensured by simply uncommenting the following line in httpd.conf file -

*LoadModule cgi\_module modules/mod\_cgi.so*

                               (or)

*LoadModule cgi\_module modules/mod\_cgid.so*

k. If all the above steps worked well, then CGI scripts can be written in Sites folder and then user can see the execution(output) in the browser by pointing to *http://localhost/<example.html>.* Here <example.html> can be any user interface code file related to the project. Usually it should be the starting page(like welcome page or the starting step for data wrangling).

l. Next, PostgreSQL need to be installed and path must be set and configured. It can be done as follows-

*export PATH=$PATH:/Applications/Postgres.app/Contents/Versions/9.4/bin*

Apart from these steps, the operating system and PATH is expected to have perl, a text editor, Git and Microsoft Excel.

**Data Source and Extraction:**

In the world now, data is superabundant in every field, every industry and every sector. World already contains unimaginably vast amount of digital information which is increasing even more vast at a rapid rate. It means data is everywhere[2]. Often this vast data is available in some raw format. This is the chief source of input data considered for any data wrangling or big data projects. For research purposes many organizations provide access to free data resources. Here are some of the numerous free data sources for research studies and experiments-Berkeley Data Lab[3], Wall Street Journal[4], Yahoo’s Flickr Shapefiles[5], World Health Organization[6], UNdata[7] from United Nations, Census Bureau[8] and data.gov[9] (the home of the U.S. Government’s open data).

But for the demonstration of this project, a real-time super-store sales data set is taken where the data is already in the spreadsheets. Usually, to perform data wrangling, initial raw data is easy to be taken from text files or spreadsheets, which would be an easy way to jump-start the wrangling process. So, a small effort to extract the real world raw data to some convenient file formats like spreadsheets or tab-separated text files will definitely save a considerable amount of time and effort in the initial stages of data wrangling process. Some smart data extraction and text mining tools are available through which data can be extracted to spreadsheets or tab-separated text files. For example, Textpresso is one such tool which is a text mining package software for biological and biomedical literature which helps in text classification and mining for further data curation[10][11].

**Database Schema Design:**

Based on the data available in the input file and goal of the project, the database tables are identified. Data is logically divided and is intended to sit in appropriate database tables. Normalization should be sufficient to ensure data is not repeated but efforts are also made to make the database schema design as simple as possible. For data integrity, constraints are enforced on the database tables after the data is loaded into respective database tables. Application of constraints following loading is done to make data loading faster and more efficient. In the current project of demonstration, the database used for data loading is postgreSQL. PostgreSQL is an advanced, open source relational database. Perl scripts are written where SQL queries are embedded within the perl code to load the legitimate data into the database.

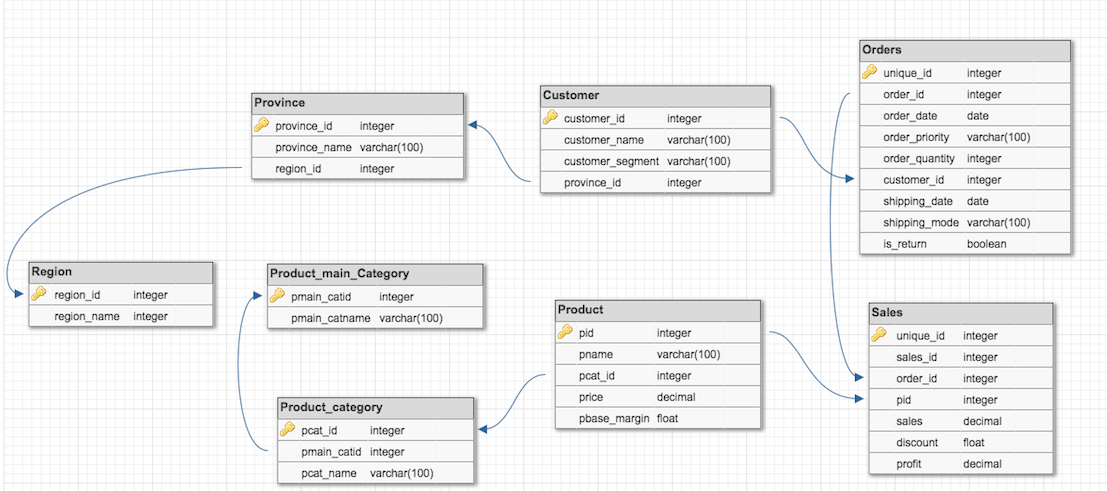


Fig 1. The database schema used in the current project

**Data Cleaning and Transformation:**

Usually raw data is “dirty” in some sense (contains errors or inconsistencies) and cannot be used as it is. So, before using the data for analysis, data analysts and curators need to perform lots of data cleaning and data transformation. Data transformation is all about transforming the data for storing it in proper format or structure for querying and analysis purpose [12].

Data cleaning and transformation is very important step in data wrangling because any traces of uncleaned data will result in misleading insights into data. So in the house of big data, data wrangling is considered as a ‘janitorial work’ because 80 percent of the time and efforts are spent on data cleaning and transformation [1].

In this project, the raw data initially is collected into excel sheet tabs. Next step would be writing perl scripts to parse the raw data into some convenient format like tab-separated text files. When raw data sets in excel sheets are given as input to perl scripts, tab-separated text files are generated. As a part of data cleaning, perl scripts are written which throws notification to curators when the data is not in appropriate format or not valid or violating any rules of data schema. For example: In a date field instead of date, if there is name of person, perl code should be smart enough to notify the same to data curator or data analyst or data scientist. Also sometimes while creating the intermediate tab-separated text files, data might lose originality and will probably appear as garbage values. For example: If the data sets contains certain characters with diacritical marks (ä, é, õ, etc.), they must be converted to appropriate ASCII equivalent characters. During data transformation, such special characters may turn into garbage values. In the real world of research and business if such inconsistencies are overlooked and loaded into the database, it could create a huge negative impact on the research or business. It would also be a costly process to rectify at the later stages. To avoid such losses, data cleaning and transformation should be performed thoroughly. The perl programming language provides good modules and packages to handle such complicated data transformation. It is often called as ‘Encoding’ in perl programming terminology. However, in this demonstration project, data sets have such special characters that need special attention during data cleaning and transformation. After performing data cleaning, data validation, proper encoding of special characters, data becomes legitimate and can be loaded into database. Before loading into databases, this legitimate data can be stored in tab-separated files or excel sheets from which data can be consumed easily and stored into database.

**Data Loading:**

Data loading phase is making the data sit into the target. Usually the target could be a very simple delimited file or a database or a data warehouse. Loading process should be done using as little resources as possible to be efficient. Apparently, as the data loading phase has interaction with the database, it is the responsibility of the data curator or data analyst to ensure that loaded datasets holds the rules of database schema.

Fortunately, perl has a very good programming support for interacting with the databases. This interaction with databases can be done using a module in perl called as Perl DBI (Database Interface). Perl DBI offers a standardized way of communication with the databases where the programmers can embed that specific interaction code within their programs[13].

**Challenges**

In this project there are some interesting challenges which one would definitely face while following the proposed method of data wrangling. One of such challenges is to persist the values set in a hashtable. In other words, the key-value pairs set in one session will not persist in a different session. During the data cleaning process, to validate the data sets, one such schema rules is followed where the data value in one excel sheet is valid if it exists in another particular sheet of the same Excel file. So, there is interdependency between the spreadsheets and validation of data in each sheet is handled in a different session. So, the values set in one session cannot be seen while in a different session. A Solution to this problem comes in the form of BerkeleyDB. BerkeleyDB is a perl module that is available in CPAN where it provides an interface to provide a temporary data structure [14].

In this temporary data structure the key-value pairs can be set while performing data validation. BerkeleyDB automatically creates a new berkeley file withe key-value pairs set inside. So being in any session of validating data belonging to any spreadsheet, the berkeley file can be accessed and the key-value pairs persists once they are set. After the purpose is served, if data analysts wanted to save some memory, the berkeley file can also be explicitly deleted.

One more major challenge faced during the demonstration of project is very long data loading times. This can happen due to the implementation of the underlying database schema. During the data loading process, if all the constraints on database tables are enabled before loading data, then it would certainly take more time to load the data into the appropriate database tables. So, to make the data loading process more efficient, it is helpful to disable the constraints like primary keys, foreign keys, auto increment, indexes before actually loading the data and later enable them. This would give the same effect as that of the former method but with reduced loading time.

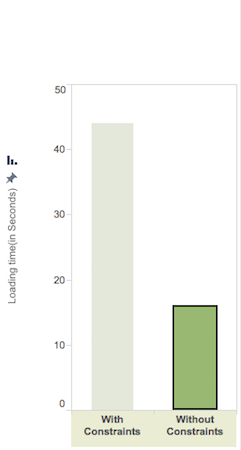


Fig 2

The above figure (Fig 2) shows the time taken to load the data into the database when all the constraints on tables are enabled is 44 seconds while it is 15.8 seconds when the constraints are disabled while loading the data. However, in the latter case all the constraints are enabled after the load completes.

**Discussion**

In this project we have performed data wrangling technique according to self-driven protocol with intent to convert raw data format into another data format so that the data finally can be easily consumed for further analysis, visualization and various statistical deductions. In this attempt when keenly observed, we have also developed a customized, semi-automated ETL tool. The current ETL tool in this project has a user-interface design and server interaction. Though the complete Extraction, Transformation and Loading steps are being performed at the back-end, the user at least is aware of what is happening at the back-end during any phase of the process. Command Gateway Interface (CGI) scripts are written for this purpose, to throw errors to the front-end web pages to notify user who is using application. Additionally, the entire process can also be executed on a command-line terminal without using web interface.

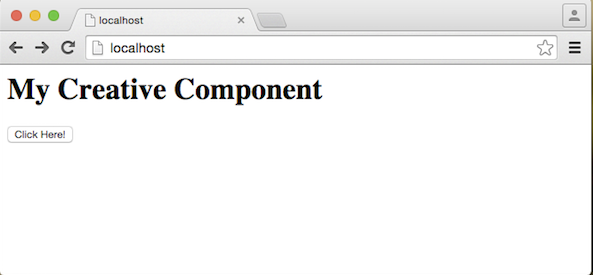


Fig 3. Showing the starting point of application

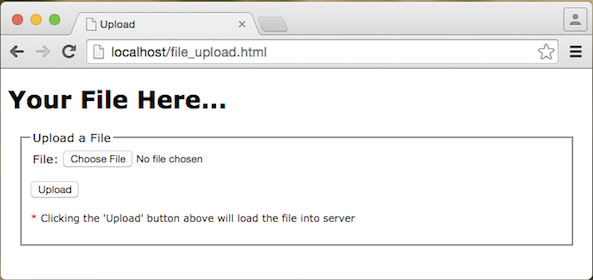


Fig 4

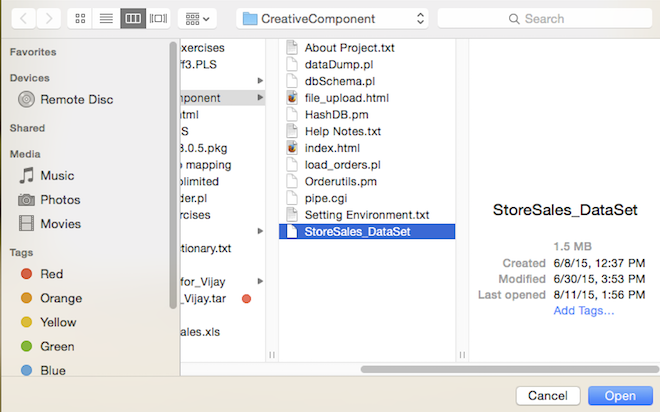


Fig 5

Fig 4 and Fig 5 shows the option in application where the user can upload/input the data file easily through browser interface.

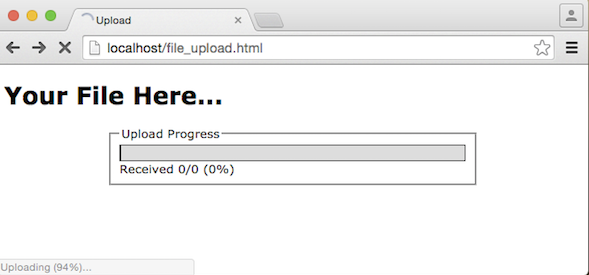


Fig 6. Showing the upload progress of data file that user gives as input

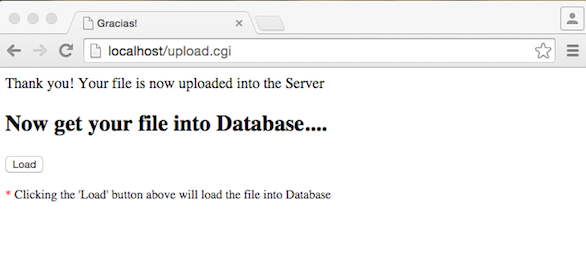


Fig 7. Shows that the data file is uploaded into server and user has an option to load the data into database with load button



Fig 8. Showing the process how legitimate data is loaded into database finally

**Limitations**

Though this project offers a user-friendly data wrangling/ETL tool, there are certainly some limitations. The proposed method or application takes the input raw data from spreadsheets and excel files only. An enhancement would be to accept input data in additional formats. Secondly, the current project does not make good use of metadata. Also there could be additional automation in the steps like data cleansing and data validation.

**Future Research**

This project is a preliminary attempt to propose a data wrangling method of that kind through a customized ETL development that can have a user interface. However, there is certainly a lot of scope for further development through research. For example, a significant room for development exists in the process of data transformation phase. Efforts can be made to automate the tasks like data cleansing, data validation. It is tough to automate data verifying because most of the times this step in data transformation needs specific rules to verify data and the rules could be volatile from project to project. However, streamlining and accelerating data validation is possible and that in turn will decrease the time to test data, compare and report. The current ETL process and data wrangling can be improved further if the process offers flexible and better ways to handle metadata development of application. There should be some process or an architecture or framework that uses metadata as much as possible. But what is metadata? Metadata is data that describes other data [15]. Metadata summarizes basic information about data and this will make the development process, findings and working with data easier. Metadata driven data wrangling will bring down the cost estimates of development, increase performance, less number of errors. It will be so standard architecture that in future the current project if enhanced to be metadata driven, then a new framework for ETL process and data wrangling would be developed. Yet another interesting area for further research is making the process cloud based. In the today’s social computing world, it would be a drastic improvement if the data wrangling can be achieved and done in the cloud.

**Conclusion**

In this project, we have developed a data wrangling/ETL tool with a desirable architecture and user-interface features. Throughout the project the data used is stored in a structured format. In the current process, there has been a lot of manual work to get the data from actual data sources to intermediate source files like Microsoft Excel. So, a great deal of efforts are wasted because of the lack of some automation. Significant efforts and researches must be made in this direction of improving and automating data extraction. Future developments of ETL tools must leverage the power of big data right from the flow of data to storage of data. Finally, it would be of tremendous help to data scientists and data analyst community if these data wrangling and ETL tools are integrated with data analytics and visualization tools.

**References**

1. Dasu T and Johnson T. *Exploratory Data Mining and Data Cleaning*.

New York: John Wiley & Sons, Inc, 2003.

1. https://en.wikipedia.org/wiki/Data\_wrangling
2. http://www.economist.com/node/15557443
3. http://dlab.berkeley.edu/data-resources
4. http://online.wsj.com/mdcapp/public/page/marketsdata.html
5. http://code.flickr.net/2011/01/08/flickr-shapefiles-public-dataset-2-0/
6. http://www.who.int/gho/database/en/
7. http://data.un.org/DataMartInfo.aspx
8. http://www.census.gov/data.html
9. http://catalog.data.gov/dataset
10. http://www.textpresso.org/
11. http://gmod.org/wiki/Textpresso
12. https://en.wikipedia.org/wiki/Extract,\_transform,\_load
13. https://en.wikipedia.org/wiki/Perl\_DBI
14. http://search.cpan.org/~pmqs/BerkeleyDB-0.55/BerkeleyDB.pod
15. http://whatis.techtarget.com/definition/metadata