Alexandra Norman, Lauren Darr, Ramesh Simhambhatla , John Heinen

**I.** **INTRODUCTION**

Tableau Performance with Big Data Sets(Summer 2017)

**T**ableu is a data visualization software developed by U.S. company Tableau Software. It is popularly used as a business intelligence (BI) tool. In the words of the company, this software meets the need of “making data understandable to ordinary people” (Tableau, 2017). Tableau is a big player in the world of data analytics with 26,000 customers as of 2014 (Trefis Team, 2015). One reason Tableau is so useful is its ability to query relational databases to build visualizations. Group member, Alexandra Norman’s personal experience with Tableau at work inspired this exploration into the functional limitations of Tableau with big data. Norman reported that Tableau slowed down considerably when reading ‘large’ tables. But, what is too big for Tableau to handle? The primary goal of this project is to test the performance of Tableau when querying large databases.

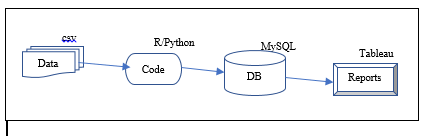
# **Project Plan**

**‘B**ig data’ is a popular term casually used in industry and academia alike. Unfortunately, there is no singular definition for how big ‘big data’ is. Forbes contributing author Lisa Arthur suggested that ‘big data’ doesn’t need to be limited to unstructured or exponentially growing data, but rather it is more important that it is well defined within every enterprise (Arthur, 2013). The aim of this project isn’t to cater to any prevailing definition of ‘big data’ but to focus on functionality. Therefore, the project’s initial plan is defined around a hypothesis from practical experience. It is surmised that a suitable starting database size to capture any decline in performance is approximately 1GB.

The general idea is to query a singular database at a series of ascending sizes to build visualizations and record the processing metrics. There are three major steps leading up to this final testing (Figure 1). Publicly available U.S. census data was acquired from the U.S. Census Bureau. More specifically, the American Community Survey Public Use Microdata Sample (PUMS) from 2011-2015. This data set is a sample of raw data intended to be used for custom analyses. The chosen records contain individual person responses as opposed to household responses. This data set consists of 298 variables.

An ideal platform for creating and using a large collaborative database is a dedicated server or cloud computing service. In theory, this would provide more efficiency than using personal desktops to upload large files to distinct, but identical databases in MySQL.

**Figure 1. Work**



# **Limitations**

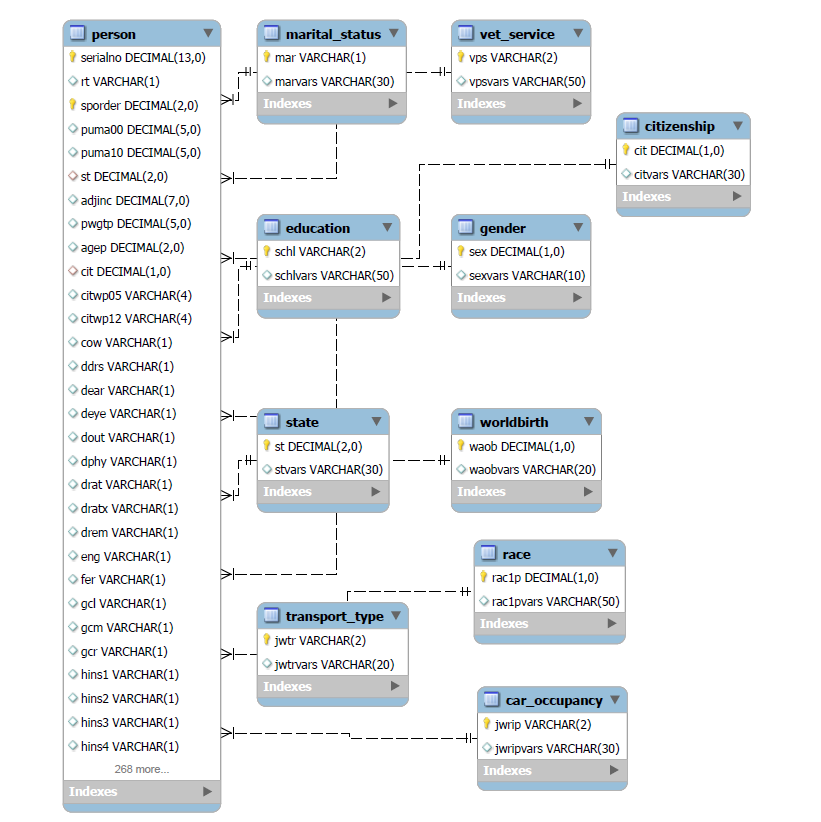
**T**he need for a hosting service proved to be a major limitation in the project plan. It was determined that a hosting service of adequate size to test a large database would not be available for free. This hindrance means it may not be possible to test Tableau with database sizes large enough to detect performance break-down.

# **Revised Methods**

**W**ith limited free hosting services available the project plan was revised to include a proof of concept and a new focus on capabilities for data scientists working remotely. A relational database was created using SQL (See appendix). This database consists of a ‘person’ table including all 298 variables available. Ten smaller tables with primary keys linked to foreign keys in the ‘person’ table were also created (Figure 2.).

Person table files were uploaded and tested at an increment of 500,000 records. Tableau visuals were created on a personal computer: system model-HP Spectre x360 Convertible 15-bl0XX, Processor: Intel(R) Core™ i7-7500CPU@ 2.70GHz, 2904 Mhz, 2 Core(s), 4 Logical Processor(s). Tableau connected to MySQL server (LIVE V EXTRACT) Fill in

**Figure 2. Database Schema**



# **V. Results**

# **VI. Lessons Learned**

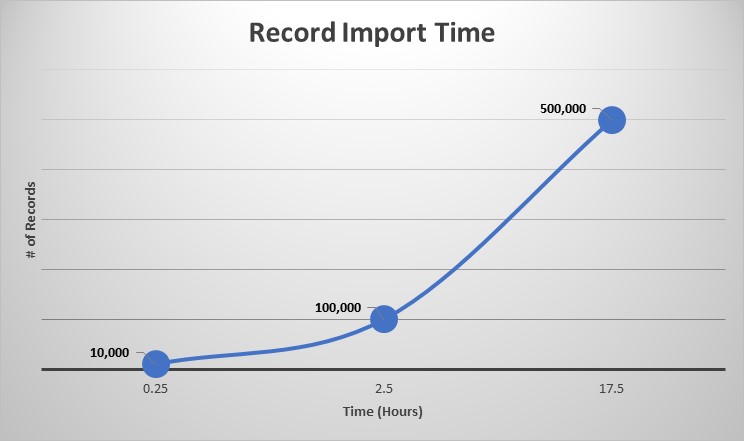
**T**he project scope at the start was to execute and identify Tableau performance bottlenecks with big data files (10M+ records). However, we have learned that, this effort will require infrastructure, knowledge and experience to complete (preferably on cloud S3 or NoSQL on AWS or Azure) that extend beyond this class.

Working with “Big Data” is difficult. With 298 variables and an infinite amount of records at our disposal, it took some time to get our hands around the data and how we were going to use it. This data could not be opened in MS Excel or Access since it had more than 255 columns. We found a csv splitter tool (<https://sourceforge.net/projects/splitcsv/>) that allowed us to split these huge files with more than a million of files each into smaller, easier to manage files to help us get started.

Not being able to work in a shared server environment forced us to be creative about how we shared information. We quickly found that loading data into our database became a time-consuming process. As shown in Figure 3., you can see that the larger the amount of records we loaded, the longer it took to load data at an exponential rate. We were forced to share the database by taking a “data dump” using the tool native to MySQL, share the “dump” file folder on Google Drive, then the other person would have to use the tool to import the “dump” into their database. Once the data was loaded into the MySQL database, this process did not take much time and allowed us to share in the responsibilities for completing our tests.

We found that when loading the data through the import tool there were some records that would be left out due to a constraint in our database. If we used a script, we could have developed a script to capture the records that were rejected. Without this, it would be very difficult to identify why these records were not loaded and take any steps to clean-up any data issues. This was another unexpected issue we found working with a “Big Data” set.

Several attempts to insert data with more efficiency were made with observed issues. *Using INSERT INTO <table name><fields> VALUES <data>.* This process is very cumbersome due to large file size and number of fields. Insertion of records will require looping the data as many times as number of records, and was abandoned after observing very poor insertion performance. Using Panda package *(df.to\_sql(name='mytest3', con=engine, if\_exists='append', index=False):* This method worked well for smaller data sets (this method can create a new table on the fly, if doesn’t exist or can append the data for the existing table). However, the laptop with 8MB RAM consumed 100% memory when 10K records file attempted. So, the conclusion was to not to use this method, though the method could logically work.



***Figure 3:*** *Load times of People Census Data by number of records to MySQL tables. The graph shows that the larger the number of records loaded at a time, the longer the load took to run. For 10k to 100k, there was a 1:1 ratio, for 100k to 500k, the ratio rose to 1.4:1.*

# **Conclusion**

## << update results, findings >>

**Appendix**

**Census Schema Script**

**How should we add it in?**

**Ten Tables Script**

**References**

1. Tableau, 2017. <https://www.tableau.com/about>

2. Trefis Team, 2015. *A Closer Look at Tableau’s Customer Base Growth.*

<https://www.forbes.com/sites/greatspeculations/2015/04/15/a-closer-look-at-tableaus-customer-base-growth/#4720765a7be2>

3. Arthur, Lisa. 2013. *What Is Big Data? https://www.forbes.com/sites/lisaarthur/2013/08/15/what-is-big-data/#16cb80f45c85*