11.19.15 – Capstone Project Report Five - TBTB

In this iteration we concentrated on adding more functionality in the stored procedures stored in the database and adding visualizations to the application itself as well. Included in this update we included the ability to query for a circular region instead of just a polygon or showing an error message and added a special type of line query that Tom will explain in his section of this report. Which means that the regular line query should be completed in the next iteration as well, as Tom was working on that in tandem.

Those new queries have been completely implemented in the server side code and linked to the client side so that all of those will function as expected and now all of the buttons on the leaflet draw plugin menu will work. To go along with this we changed some of the UI to try and make it perform better on smaller screens as it did not seem to be as responsive as expected, more tweaks to be worked through on that front, and we also added some visualizations with the current data. These visualizations were created using d3 and added within kendoUI windows to be moved around the screen as needed. A parallel coordinates chart was built to be interactive and display trends between all of the data returned in the most recent query. The pie chart shows passenger count across all of those returned trips to show what the most common taxi size is for the time period and the bar chart shows more information for the current data. Right now the charts only display the current data as they are loaded but in the next sprint we will add the ability for it to show all current data displayed on the map, and be dynamic as more queries are done.

Tyler:

For this iteration I concentrated mainly on integrating the new queries that Tom created into our application on the server and the client side. This involved converting the points into the proper SQL geography type to kick off in the stored procedure and properly converting that into

the object to be serialized. It is worth nothing that it seems all of the different Geo JSON shape types will require a different sequence of tasks to get it in the format we need. Where the polygon has all of the points listed under the first element in a ‘coordinates’ array the linestring needed for the line query as them in the ‘coordinates’ array itself. Simple if statements take care of this but it is worth noting. It is also worth noting that we seemed to run into the problem again where Entity Framework will send NULLs for all of the parameter types to try and determine what the return set of a stored procedure is and then it will generate the result model in the application based off of that. But if your stored procedure dies when NULLs are passed then it will think the return value is just an integer and I had to work with Tom to change around what the arguments were to get it to work in that format.

Also I added all of the window templates for the charts to be displayed in and gave that code to Brendan and Bill to place their charts into. As described above it will only use the data currently in a currentData Javascript object and this is just temporary as it will soon be looking at a live collection of data currently displayed on the page. I developed a visualization myself and decided to use a Parallel Coordinates chart that shows trends and relationships between all of the data fields returned includeing the Number of Passengers, Total Cost, Total Distance, and Total Trip Time. You can also interact with the chart and only look at certain values on the chart lines to get a more concise view of the data you wish to see instead of everything that was returned. Getting this chart and the others into the kendo window required some interesting code to make it work, as you need to just put the actual HTML string as the contents of the window through the functions of the Kendo UI library, but we got it all figured out and working as it should.

<u>Brendan Gray:</u>

This iteration was a giant learning curve for me. I had many troubles attempting to implement the bar chart. This was largely due to my negligence in paying attention to detail. I typed a couple of errors in my code and had to undergo rigorous code tracing inside of my chrome browser in order to find it. Unfortunately when creating my bar graph I mispelled “height” as “heigh” and so my bar graphs appeared as if they were not there. However, they were there, but they had no height to them.

Also, I mismatched the ID that I had in my CSS file. So even if I hadn’t misspelled height, they still weren’t appearing because they didn’t have any set styles. This is how I learned much about D3 and CSS. I also mixed in a little bit of JQuery thanks to Tyler’s recommendation. All in all, I was able to plot the data in a bar graph using hard coded data. Implementing data from a query is the next step in creating the bar graph. Databases are not my forte, so obtaining and applying data that has been queried will be my next learning curve. B the next iteration, I should be adept when applying D3, and I should have a small example of experience with data returned from a query and how to manipulate that data.

Bill:

During this iteration i was focusing on getting the Pie Chart implemented to show the number of passengers for each slice of the pie chart. The main issue i was having for the data was getting the currentpoint object for the data being shown. I don’t have that feature implemented yet but my plan is to get it implemented in the next few days. The main things I got done was the Pie chart and understanding an implementation for getting the data to be outputted. Another issue i found was with the queries on my laptop was the trying to get them to run at a reasonable speed since it was taking three minutes to find 1300 rows. So i moved my project over to my desktop computer at home and no issues have been found so far with the hard drive speed in that computer. Also i started on getting a basic user guide and a programming guide for the project it is not in the project yet but will be in a few days. Overall i

didn't get a lot done with this iteration but have the ground work to get a lot done in this next sprint and final push to get the project done.

Tom Taylor:

During this iteration I primarily worked on three different things. The first one was still continuing work on data validity. My goal is to have a "production ready" database when we come back from Thanksgiving break. To accomplish this I need to have dealt with and removed all data that I consider invalid. At least from my perspective of what is an actual valid taxi trip. The second one was more work on the queries/ stored procedures that I need to provide to the front end. I finished up what is now known as the linesIntersection stored procedure and started working on figuring out how to give some of the geography shapes we are generating some volume. I need this to be able to produce queries of the topographical type. The last and newest thing I started working on was trying to create views and looking into query performance in general. As the stored procedures get more complex, related to analyzing geographical data types on the fly, the result set return time does begin to suffer at times

Regarding data validation I was able to turn the medallion and hack_license columns into integer data types thus reducing the sizes of those fields significantly. I have also removed all of the rows that have start and stop positions that are exactly the same as well as trip times that are of duration zero seconds. I have removed other anomalies as well, such as trips with zero passengers, etc.

I have been able to figure out how to give "volume" to geographies if it is necessary. I use this concept to "move" the points towards the roads. As we know many of the points do not actually relate to roads on the map. One option would be to do preprocessing of the data to actually move the points to correctly position them on the nearest corresponding road. My approach to this is a little different, I emulate the point move by give the start/stop location some volume of a few meters. In this way when we draw the line of a road on the map I will return points even if they are offset from the geography of the given line by a few meters. This extended method is built in SQL as well and is known as .STBuffer (distance). The distance is what gives the geography volume and is in meters. This can also be done on the line geography that I am

creating from the two points provided when the drawn line data is provided to me. This is an area where I am starting to notice some return set performance issues so I have been trying different combinations of things looking for speed improvements.

As far a performance goes, one of the things I have been trying to do is set up some views so that I can store precomputed buffered geography regarding the start and stop points. Unfortunately I have been struggling to get this accomplished. I keep getting errors when trying to build the view, mostly related to memory. I know that computing geographies is calculation intensive, however I am not physically running out of memory so I am troubleshooting through this. I am really interested if I can increase performance at the cost of disk space so will continue down this avenue in the next iteration.