10.01.15 – Capstone Project Report Two - TBTB

These past two weeks there has been substantial progress on our project and we have gotten to the point where the different parts of the application are starting to fit together. The database construction has been mostly finished and aside from some small tweaks it is essentially done. The database has two spatial indexes based on the pickup and dropoff locations and also has indexing on the two dates so those queries are also faster. The data has been trimmed down as well to make the size of the database smaller and at the moment, we have one month stored. The stored procedures have also been created for a region query and a nearest point query.

Entity framework has been set up within the web application and linked to the SQL database on a home server that all of the members may access. Code has been written on the server side to interface with the stored procedures and has been tested and linked to the client side. On the client side some more work has been done as well, the map is being displayed and the group now knows how to get points from Leaflet Draw and give them to the server side to get the data from the database. Drawing points has not been completely figured out yet, but we are close and we feel very solid progress has been made, and because the database is completely set up and in an accessible location, progress will increase very rapidly.

Tyler:

In these past two weeks I have continued my job of guiding the group through the necessary processes in order to create this project. I have taught and explained how an MVC project is setup in Visual Studio and how to do certain things such as Bundling scripts and stylesheets together, how to organize the file structure, how Entity Framework works, and how the server side code and client side code will interface with each other. This has been a large part of my work because none of the group has worked with any .NET/C#/MVC or actual web applications before. I have also tried to make the group as organized as possible by creating specific Tasks and Changelog files that the group can access with proposed completion dates so no one has any issues with finding things to complete.

I have also taken the responsibility of creating the server for the group to host our project on, and use as a webserver for actually presenting the project in class and to anyone else who may use it, the server is also hosted to a URL that I have rented for the year. Because of this I will be the one as the administrator for the server, updating the code when necessary and making any final changes to the database as it is the master copy that the entire group uses when developing on their local machines.

As Tom neared the completion of the database I was able to start more concentrated development in the web application which involved configuring Entity Framework with the proper

credentials so it would not break when updated and pulled from the git repository. Also I was able to link the two together and call the stored procedures from within the C# server side code in the application, as well as test the routing by calling it from a client side Ajax call and configuring all of the data types and constructors we need to make that all possible.

Brendan Gray:

Throughout the last two weeks, progress on the map has been substantial. I was able to successfully integrate Leaflet.Draw, which is a critical asset in our project. Leaflet.Draw will be used for drawing arbitrary points, squares, and other polygons inside of our map, which was integrated in the last iteration of our report. Integration with our last iteration has gone smoothly, with the map with just the way we want it for now.

The code was easy to understand and was easily used with our existing code. Because of this, Bill and I have been able to comprehend it's multiple included files, such as the JavaScript files. The examples it has come with have also been very handy, and we have been able to glean much from these tutorials. So far, I can safely say I have already learned much from this class. Bill will be talking about how he integrated an extremely important function that is used with our database.

Bill:

Throughout the past two week my main goal has been to help Tyler and Tom integrate with Leaflet Draw with the backend. So far we have Leaflet draw drawing a rectangle and outputting the points in GeoJSON format. Now we need to be able to take that and plot all the points located in the rectangle box and show them to the screen. Our next goal after we get the link between GeoJSON and leaflet draw is to be able to go from GeoJSON and be able to take the coordinates and draw a polygon on the map. Once we get those two features working we will have a small link between the backend and frontend to this project giving us the able to integrate more features.

Tom Taylor:

The first thing I needed to investigate was primary keys, clustered indexes, and indexes in general. I knew I needed to work on this because on my first attempt my index files ended up bigger than my database files! After that I worked on understanding the two types of geospatial indexes, geography and geometry. I actually created two different tables, each indexed by one of those two spatial index types. I did this so I could actually do query comparisons between the two types. The last thing I worked on in this cycle was converting some of my sample queries (region and nearest point) into stored procedures that we will be able to use from our calling application (.NET MVC) to actually generate result sets into the calling controller.

As far as indexing goes I learned that for geospatial indexing you must have a clustered index and that each table can only have a single clustered index. I then investigated what the term clustered index actually means. Basically, it is directly related to how the data is physically stored in the database. That is why there can only be one of them as you can only physically store the data in a single way. There are four properties that make for a good clustered index: narrow, unique, static, and ever increasing. Now the original primary key that I created was unique and static, but it was not narrow (smallest byte size possible) or ever increasing. To take care of this I decided to change my primary key from medallion, hack_license, vendor_id, and pickup_datetime to an auto generated integer identity column. That then allowed this new clustered index to meet all four of the above criteria. When I went back and recreated the tables in this new way. The size of the table indexes were about a quarter of the size they were originally and the two spatial index columns generated in about half of the time.

I used this new knowledge to then create two tables, each one using one of the different types of geospatial indexes. The primary difference between the two types is that geography understands the curvature of the earth related to longitude and latitude while geometry only understands a flat plane. As far as indexing goes the geometry spatial requires that you create a bounding box as part of the index structure. This then “bounds” the plane to be just a specific rectangle related to longitudes and latitudes. Since geometry is “simpler” and the index is bounded these queries should be quicker than geography queries and I was able to prove that.

The last part was working on two actual queries, a region query and a nearest point query. I figured out how to create both of them and then ran them against both the geography and geometry type spatial indexes. The nearest point query for the geometry index was very slow compared to all of the others. I figured out that the problem with this was the scale between geography and geometry. The actual scale for geography is in meters and the scale for geometry is really unknown as it is based on the size of the bounding box used for indexing. I was able to find a way to take the geometry distance and convert it into a geography distance. This then allowed me to correlate my query results and understand why there were magnitudes of difference related to distance. Once I used an appropriate distance in the geometry nearest point query the time for the query came back into line and was then a little faster than the same nearest point geography query. Converting these testing queries into stored procedures was actually a pretty straight forward procedure. All that was really necessary was to create the variables that would be passed into the stored procedure and then build the queries using those variables.

Reference:
https://www.simple-talk.com/sql/learn-sql-server/effective-clustered-indexes/
As a guideline, clustered Indexes should be Narrow, Unique, Static and Ever Increasing (NUSE).