Part 2 - Building an Enhanced DocGraph Dataset using Mortar (Hadoop) and Neo4J

In the last post, I talked about creating the enhanced DocGraph dataset using Mortar and Neo4J. Our data model looks like the following:

Nodes
Organizations
Specialties
Providers
Locations
CountiesZip
Census

Relationships
* Organizations -[:PARENT_OF] - Providers -[:SPECIALTY]- Specialties
* Providers -[:LOCATED_IN]-Locations
* Providers -[:REFERRED]-Providers
* Counties -[:INCOME_IN]- CountiesZip
* Locations - [:LOCATED_IN]-Locations

Each of the nodes will have several properties associated with them. For example, Organizations will have a name associated with it. Locations have a city, state and postal code associated with each location.

Data
The data we are going to use is the initial DocGraph set, the Health Care Provider Taxonomy Code (NUCC) set located here, the National Plan and Provider Enumeration System (NPPES) Downloadable File here, and a zipcode to state file and the income per zipcode downloaded from the US Census. These files were loaded to an Amazon S3 bucket for processing.

Mortar Project
To create the Neo4J graph database, we will need to create several files to be loaded into Neo4J. To create the files, we are going to create a Mortar Project and use the pig file that we created in the last post.

Create Mortar Project
In order to fully leverage the Mortar Project framework, I created a mortar project which makes it available in GitHub. This will create a new project skeleton and register it with Mortar. This project will have folders created for commonly used items, such as pigscripts, macros, and UDFs.

Pig Code
Any Pig code that you want to run with Mortar should be put in the pigscripts directory in your project. I replaced the example pigscript in that directory called my-sample-project.pig with my docGraphNeo4J.pig script.

Illustrate
Illustrate is the best tool to check what you've written so far. Illustrate will check your Pig syntax, and then show a small subset of data flowing through each alias in your pigscript.

To get the fastest results, use the local:illustrate command.

Once the illustrate result is ready, a web browser tab will open to show the results:

Mortar Watchtower
Mortar Watchtower is fastest way to develop with Pig. Rather than waiting for local or remote Pig run, you can validate that your scripts work simply by saving. Watchtower sits in the background analyzing your script, showing you your data flowing through the scripts instantly.

After installing Mortar Watchtower, I was able to do near realtime analysis of the data simply by typing in:

Once I type that into my console window, I see:

A browser window then pops up:

As you can see, the Watchtower Viewer redisplays your script with example data embedded inline with each alias. You can click on the header of this inline table to toggle between different numbers of example rows. You can also click on any given table cell to see the complete data, including any truncated.

Full Run on Mortar
Once the code was ready for running, it was time to run on a full Hadoop cluster. To specify cluster size for your run, use the --clustersize option:

When I ran these jobs on the full Hadoop cluster, it ran in about 16 minutes. It wrote the following records to my Amazon S3 buckets:

Summary
In summary, I was able to take the three raw data files, write a pig script to process the data, run the pig job on a Hadoop cluster and create the multiple files that I will need to populate the Neo4J instance.

Why did I choose Mortar?
Mortar is fast, open and free. As Mortar says, using a Mortar project provides you with the following advantages:

* Pig and Hadoop on Your Computer: When you create a Mortar Project, you get a local installation of Pig and Hadoop ready to use, without needing to install anything yourself. That means faster development, and better testing.
* Version Control and Code Sharing: Mortar Projects are backed by source control, either through Mortar or your own system, so you can collaborate with team members on a project.
* 1-Button Deployment: When you're ready to run your project on a Hadoop cluster, a single command is all that's needed to deploy and run in the cloud.

Using Mortar's Watchtower, I was able to get an instant sampling of my data, complete file watching, instant schema validation and instant error catching.

For me, Mortar was easy, fast and a great tool to get the data ready for loading into Neo4J.

Next Steps
In the next post, I'll write about how to move the data from the data files and load them into Neo4J.