Organizing ISS Experiments and Results with MongoDB and Neo4j

Alec Buchanan

**Project Overview:**

The website that holds all the data about the experiments performed on the ISS does a poor job showing the relationships between different data points. For example, what categories does each space agency do research in? Or, what keywords relate to other keywords or experiments? These relationships could be important to scientist and also make the data more readable for the public.

This project will focus on creating the relationships mentioned above. The way I am going about doing this is by using two different databases. I will be using MongoDB, a document database, and Neo4j, a graph database. Neo4j will be used to map relationships in the data, while MongoDB will be used to store, organize and retrieve information.

**Other Programs Used:**

A couple other programs will be used alongside MongoDB and Neo4j. The first program is mongo-connector. Mongo-connector is started while both MongoDB and Neo4j are running, and it is used to keep the two databases in sync. Whenever information is inserted into MongoDB, mongo-connector will push the information to Neo4j. The second program used is mongoimport. Mongoimport is used to upload documents, in the form of JSON files, onto the database.

**Scraper3.py:**

Scraper3.py is the program that I wrote, and it populates MongoDB with data and Neo4j with relationships. Scraper3.py starts out by requesting NASA’s webpage that holds a list of all the ISS Experiments. From there the webpage is broken down and creates a list of categories and sub categories to be imported to MongoDB. It also stores a list of relationships linking experiments to categories and so on. The program then loops through each individual experiment by passing the link to the *pasreExperiment(link)* function.

The *pasreExperiment(link)* function dissects each experiment. It recognizes the main topics and sub topics. It will separate the topics and turn them into individual properties in MongoDB. It will also identify the different agencies involved with the experiment. Finally, the function will create relationships between the different agencies and data points.

After the website has been scraped and the information has been parsed, it needs to be imported to the databases. The program will convert the data, which is stored as lists and dictionaries, into different JSON files. Each JSON file will then be passed to mongoimport. Here is where the information is inserted into MongoDB and pushed to Neo4j. This process can take a little while; that is why the program waits 60 seconds before adding the relationships. After the 60 second wait, the program connects to Neo4j and adds relationships.

The program has finished, but there is one more thing to note. Relationships in Neo4j can be created through mongoimport. There are two separate ways to do this, but neither give us the desired result. One way requires the use of nested documents which ruins the MongoDB schema. The other way requires an extra row to be added to the MongoDB document, but it limits the document to one relationship.