Script – Jacob Dyer

Intro

Hello. My name is Jacob Dyer, and in this video Lukas Duball and myself are going to talk about our final project for CS 523.

Our project focuses on comparing three database systems: MySQL, MongoDB, and Neo4j. What we would like to determine is under what circumstances do these systems perform well in addition to when they perform poorly. To do this we have created a list of queries that we will run on each system and we will use the time it takes to return an answer as a performance benchmark. All the systems will be running locally on our machines so there should not be any network delay involved in our analysis; keeping the results confined to whatever the database is doing to complete the query.

Before I hand off to Luke, I would like to take some time to discuss the data we used for the project.

Discuss the game

To make sense of the data we need to first understand where it comes from. Our dataset was pulled from an online nightly data dump of Elite Dangerous. Elite Dangerous is an MMO game set in space where players are the commanders of space vessels. The goal of the game is to amass wealth and to become powerful within the factions that exist in this universe. These factions control many star systems within the Milky Way and are usually at constant war. Typically, everything in the game is decided on how many credits you have, credits are the primary currency. There are many ways to accumulate credits. To list a few there is: Trading, Asteroid Mining, Piracy, Private Faction Jobs, and Bounty Hunting. The game is made so that if a player jumps to another planetary system, and that system has yet to be visited by a player, it will procedurally generate a new system and all the planets within that system. This makes the play space somewhat infinite. The nightly data dump is publicly available but a bit limited. The owner of the data dump used to dump all the celestial body information but has stopped doing that because of how many celestial bodies are currently in the game. I think I read it was last about 3 gigs. The files we used cover less about the natural elements, such as celestial bodies, and more about the civilization that has been made around those bodies.

Discuss each file

Most of the people in this universe live on space stations. These stations orbit some celestial body or are on the surface of some planet and are central to the entire game. Stations are where the player will trade/barter, sell/buy ships, as well as interact with non-player factions and player factions alike. All this information is located within our first set of data: Stations. Most of the data we get from the dump is formatted in json so there are nested elements beyond properties like a name and location. Stations also contain services like repairing, refueling, and a marketplace. Important foreign keys here are: System ID, which denotes which planetary system the station is in, and imports/exports, which is an array of commodity IDs that show what this station might produce and what it needs to produce that commodity.

Next is the populated\_systems file contains information on the planetary systems that have people living in them. The real important data here is the absolute position of the system. The position is given as x y z coordinates and they are how we can determine the distances between two systems. This file also contains data on who controls the system and how many people live there.

The next file of interest is the factions file. Factions in Elite Dangerous can control both stations and systems. A faction has a certain type of government associated with it such as: Communism, Cooperative, and Patronage. These factions also have an allegiance to the games hardcoded ruling bodies such as: The Empire, Federation, or being Independent.

Next are Commodities which are outlined within the commodities file. This file acts as a reference to listings. This file contains the name of a commodity and some aggregated data from listings such as: max buy price or minimum sell price.

Next is the listings file, which, is by far the largest file in this project. It contains about 4 million listings from commodities being sold at each station. These listings contain the buy/sell price of a commodity and its demand. It also has a reference to both the station that listing is located and the commodity this listing represents.

Finally, we have modules. Modules contains information that is referenced by stations. Stations stores a list of modules that that station sells in a nested list within the json. Modules are like commodities in the fact that they are bought and sold at stations but differ by the fact that they are upgrades for player ships and don’t behave like commodities when they are resold as they are unable to retain their value.

LUKE GOES HAM HERE

* Covers schemas
* Covers queries up to #5

Query visualization

Alrighty, before we wrap up our presentation, I would like to take some time to show you all some interesting visuals we were able to generate. Because all our systems have positional data, I went ahead and found a python library called plotly that can make some nice 3D scatter plots. In addition to plotting the systems I was also able to adjust their point size and scale it to the population that lives there. I also managed to build some functions that will graph lines between systems to connect them if they are close enough. These lines can be turned off at any time just as the nodes can be.  
 The first visual I would like to show off is a graph consisting of all the populated system. As you can imagine it takes quite a bit of time to generate this graph given its 4 million components but, after it is generated it performs quite well. Something interesting to note is that not all populated systems are close by. Some are scattered out on the outer rim and very far away from any other populated systems. This means they are most likely populated by only player factions.

Next is a branching query that returns a bubble around two systems. In this case Nerthus and Sol. Sol being the system containing Earth and all the other planets familiar to us. In this graph you can see that Sol is quite large compared to its neighboring systems for its population is quite immense. Another thing to note is that all the nodes have colors corresponding to their respective major faction. Making it easy to see where these factions have control over the milky way.

Lastly, I graphed the results from a pathfinding operation. I graphed two of these: the first is of Nerthus and Sol. The next is the result of one of our main queries that resulted with the two systems that allow for the highest trading profit margin. Both performed really well in time and resulted in an easy to follow graph.

Conclusion

With that we are about out of time. All the data we used can be found on eddb.io/api and we put our project into a github repository if anyone is interested in looking at our files. I will put a link to both in the description. Anyways, thanks for watching.