Trace Levinson

Peyton Norder

Project 01: PostgreSQL vs. MongoDB

PostgreSQL:

The Data we used for this project was connected at several levels making PostgreSQL work very well for the implementation. Foreign keys allowed for the recipe table to reference the ingredients allowing for easy access to the specific ingredients needed for each recipe. Foreign keys also helped keep the database semi regulated as the compiler would not allow for improper use of the rows due to the foreign key constraints. Postgres tables are meant to reference one item per row, meaning an array of items cannot be inputted to a table and be referenced later easily. This became a major problem when attempting to insert multiple ingredients into one recipe. We created another table with a many to many relationship to deal with this problem, we called ours cooked\_with. By using this cooked with table we were able to avoid the issue of adding arrays to our tables and instead queried the cooked\_with table with the recipe\_id to determine what ingredients are needed for each recipe. This was a drawback to using PostgreSQL. Another drawback was the way we handled the add\_order which included a CREATE RULE to update the inventory when an insert to order was done, each value had to be individually inserted into the order table, if one insert statement with multiple values was used the inventory was not correctly updated. Generating the data for PostgreSQL was done with a Python script that created csv files to be imported. Generating data was easy and inserting it was simple adding to the positive for PostgreSQL.

MongoDB:

Mongo built in functions really made the implementation of the database easy, when adding an order the ability to use db.recipes.findOne to determine the correct recipe via id. A document was created and assigned the values. Mongo allows for arrays and for specific elements of the arrays to be pulled out making the implementation of the addOrder easier than PostgreSQL. The ability to find and delete all documents belonging to a collection made it easy to rerun seed scripts while creating the database and not have to worry about dealing with old collections of documents. Using Javascript with Mongo was a major upside to the database and allowed for the creation of scripts to be simple and intuitive. Seeding our data for Mongo was very easy as all that was needed was the data created from the Python function in JSON format and then a javascript file to simply insert the data. Mongo allows for the running of scripts in the command line which made testing Javascript files much easier. This however caused Mongo to delete all then documents in the database and then reseed them, which was fine, it just meant the ability to dynamically change schemas did not have a major impact on the project. All documents in Mongo being in JSON format helped with the implementation of the add\_order function and although Mongo does not have a set schema this really helped us keep the information in a logical manner and helped us visualize what parts of the data needed to interact, like the ingredients needed for a specific recipe that changes the amount of ingredients you have overall.

MongoDB was our preferred language to maintain the database. The ability to not have a set schema along with the ease of interacting and creating data made it outperform PostgreSQL.