Introduction

FamilySearch is a company that collects genealogical records with the intent of creating a family tree containing every human that has ever lived.

The goal of *Big Taco Mining* was to analyze a dump of FamilySearch's database, and determine what percentage of an area's population at any given period of time does FamilySearch have record of.

This is important to know because knowing the amount of data FamilySearch owns can influence what records they try to acquire in the future and how to involve Church members in their work.

Data Provided By FamilySearch

FamilySearch provided us with a dump of their database in the form of millions of JSON blobs where each JSON blob contained the information for one person.

Each JSON blob contained data of the following format:

```
{
    id: <id of the root person>,
    persons: [
        {
            id: <id of the person>,
            facts: [
                {
                    type: <fact type, i.e. birth | death | census…>,
                    date: <date when fact occured>,
                    place: <place where fact occured>
                }
            ]
        }, ...
    ],
    places: [ <list of places referenced to by the facts>],
    relationShips: [ {
        type: <relationship type, i.e. sibling | spouse | parentChild >,
        person1: <id of person1>,
        person2: <id of person2>
    },...],
}
```

Difficulties With Data

- For each person in Family Search's database we needed to find the range the person was alive. This was difficult because many the dates associated with people did not have a normalized format.
- We also needed the location where a person lived, this was difficult because the locations associated with people did not have a normalized format

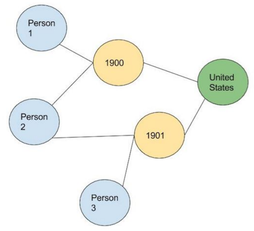
Big Taco Mining

Consulting Services for FamilySearch

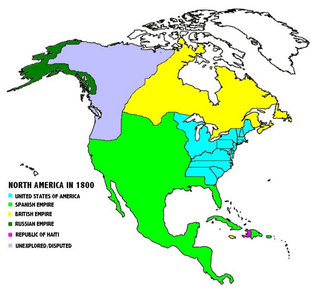
Isai Mercado, Jeffrey Young

Original Solution

- Write a spark mapreduce job that pumps data into a graph database called Neo4j. The graph database schema would have the following format:



- Have visualization query neo4j to display results
- Have different maps for every year displaying country border changes



Problems With Original Solution

After research and lots of effort we found out that our original solution was not implementable primarily because:

- Neo4j does not scale well horizontally. FamilySearch has records of billions of people. We couldn't get Neo4j to handle that many relationships and nodes
- Open source historical maps are very difficult to find. When we did find open source historical maps, it was very difficult to create a client that could interact with the maps to visualize data.

Final Solution

- Write a spark mapreduce that takes the json blobs from FamilySearch as input, and outputs the number of people that were alive in a country during a 50 year time span.

  example results:

  ('belarus::1500', 6681)

  ('belarus::1550', 62186)

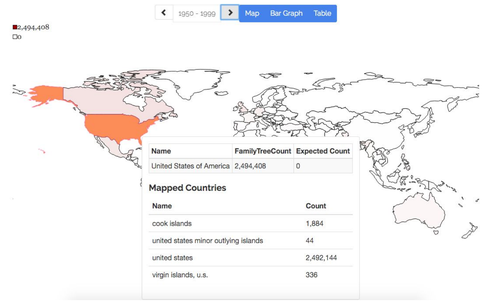
  ('belarus::1600', 120823)

  ('belarus::1650', 174538)

  ('belarus::1700', 266806)

  ('belarus::1750', 430718)

- Import mapreduce results into a mysql database
- Map each country from the mapreduce results to a modern-day country.
- Visualization queries mysql database to visualize results



How Our Final Solution Handled Data Difficulties

Our final solution worked, however we had some difficulties handling inconsistencies in the JSON blobs provided by FamiyTree.

- In order to handle unusual location names in the data, we used a library. Our counts only look for:
  - country names with good spellings in native language unicode
  - state names with good spellings in native language unicode
  - 2 or 3 character country abbreviations (US, USA, MX, MEX)

  Any person in FamilySearch's database that did not have a location associated with them fitting the above criteria was not included in the results.

Results

The results of our project can be viewed at: https://big-data-family-tree.herokuapp.com/