MostFunGroup! Ahmed Boukhousse; Zach Gharst; Anna Johnson; Matt Miller

Introduction:

In order to study the relationship between earthquakes, regions, and insurance policies, we have decided to use the relational database model to fulfill the requirements of storing, retrieving, organizing, and deriving data. The target end-user for this product would be someone in the insurance field looking to craft new insurance policies for clients based on their location as well as adjust or remove current policies. A separate front-end model will also be available for prospective policy holders to determine the factors that might affect the pricing of an insurance policy.

Requirements:

The project requires information that must be collected about:

- There is certain data collected about earthquake events. An earthquake occurs in a location (possibly composite: latitude and longitude), has a magnitude, and happens at a timestamp. The earthquake event could come with a tsunami depending on the source. As well, there will be a link for more information from the USGS agency. An earthquake will have a many-to-may (took place / happens) relationship with the region.
- An insurance policy has data for its location and price. It's important to observe the effects that earthquakes have on insurance policies, so there will be derived data such as the number of people likely to be on an insurance policy. The insurance policy will have a "Prices" relationship (many to many) with the earthquake.
- Region (or some other type of location) has area bounds that earthquake events happen in them and insurance policies that reflect the region the properties are included in. The region will also have a many-to-many "Covers" relationship with the POLICY.
- Clusters are groups of earthquakes bundled together by some sort of physical phenomenon (a single source, for example). Earthquakes will have the relationship that they are a part of a cluster.

Functional & Non-Functional Requirements:

- Queries on earthquakes, clusters, regions, and insurance policies for a non-technical end user.
- Examining the relationship on earthquakes and insurance policy price/prevalence.
- Deriving data such as number of earthquakes in a region, the cost of an insurance company due to damages, etc.
- Ability to insert, update, and delete from the database: earthquakes, insurance policies, clusters.

Architecture:

- Presentation tier: This is what a typical end user sees. We are going to create two separate custom HTML/CSS framework from scratch that is sent to the end user (either an insurance company or an individual looking to purchase insurance) via an HTTP server.
- API tier: A custom built PHP API to handle requests from the MySQL database in the backend. The user will request a page, such as “Earthquakes” or “Cities”, and then the HTTP server will send a request to the MySQL database for the data to be presented.
- Data tier: We will use the relation database MySQL to store, retrieve, and organize the data for this project.

Platform:

The server will be running on Ubuntu 20.04 with the following necessary services/programming languages: Apache HTTP Server 2.4.46, MySQL 8.0.21, SQL:2016, PHP 7.4.10, HTML 5.2, and CSS 2.1.

Scope & Ideas:

- Have the user be able to enter in a location (GPS or postal code?) and return the information related to the earthquakes near them.

- We could potentially look at factors such as prevalence of earthquake insurance, amount of damage, number of injuries, and how this data relates to the frequency and magnitude of earthquakes as well as insurance rates by region.
- Look at how much insurance companies have had to pay out as a result of past major earthquake events to help inform insurance companies on how they should set their rates for different regions.
- Observe shifts in earthquake behavior over time to possibly predict future trends.

Team:

Rather than have distinct and separate roles, each team member will be involved in all parts of the project. This way, every member feels involved in the process and adds the skills of front-end, back-end, database administration, data collection, and data architecture to their arsenal. Team members will work individually on data collection and idea development and will share this information in weekly meetings where all team members will collaborate on product development.