CS348 Project - Final Report

Abdullah Bin Assad Chandana Sathish Lukman Mohamed
Vikram Subramanian Dhvani Patel

August 13, 2020

https://watch-dog.azurewebsites.net/

Application description

Application User

We have created an interactive web app for people interested in exploring crime data in the Greater Toronto Area. People who wish to assess how safe a neighbourhood is before investing in a new property or novice drivers trying to avoid accident-prone roads or just people curious about crime rates around them can greatly benefit from our application.

Interaction with the app

A user would simply search for a query (as seen on the demo application) using the drop-down on certain words in the question to tailor the search to their needs. For example, "I want to explore bicycle theft crimes in 2019 (year) citywide on a bar chart" can be one of the many queries a user selects. Alternatively, a user can also pick from a list of predefined queries if they are unsure about where to start. Once the user clicks "OK", the page updates to display the requested query and allows them to change the representation/visualization of the data as well as further refine their search with more options. We also plan on adding additional features as follows.

Key features

1. Filter crime/traffic events (starter question with drop-down for filtering different columns as seen in demo) and show results on a table
    - Filter by crime indicator
    - Filter by year/month
    - Filter by neighbourhood
2. Display crime data
    - bar chart, line chart, and pie chart
    - map, heat map
    - summary/total count

3. Create predefined complex queries in the form of question and answer
    - Example - At what hour do most crimes occur, which neighbourhood has the most number of traffic accidents, is there a correlation between education/demographic and crime, etc.
4. Provide users with information on which police division they are situated closest to based on the address they provide us with
5. Report a crime
6. Interactive "How well do you know your city" feature
    - Let the user guess certain values from a query they select, then show them the actual results and tell them how close they were

Data

The tables we get from the Toronto Police Open Data:

- `https://data.torontopolice.on.ca/pages/open-data`

We get a few additional columns of the census from the Toronto City Open Data:

- `https://open.toronto.ca/dataset/neighbourhood-profiles/`

The tables themselves are a little difficult to work with. Therefore, we create a data parser (in code.zip) to convert it to .csv files that fulfill our requirements. Sample data and production data are part of the code.zip as csv files (ex. Neighbourhood.csv, CrimeEvent.csv).

System support

Our interface consists of an interactive web-app. Both the web application and the SQL database are hosted on Microsoft Azure. Azure AppServices and MySQL are modern, scalable, efficient and flexible. This setup is ideal and efficient for building a web app such as ours; plus Azure is cheaper than the alternatives.

The back end is made up Node.js + Express.js from which API end points are exposed. This is what directly talks with the database. The front end is a React application. The notalable NPM packages used include: the Semantic UI React library for UI components, the Mapbox API for the maps you see in the application, ChartJS and D3.js for the data presentation, and ag-grid for the table.Thus, our web application code mostly consists of JavaScript. Members of our group have experience with Node.js and React which gave us an excellent basis for our project.

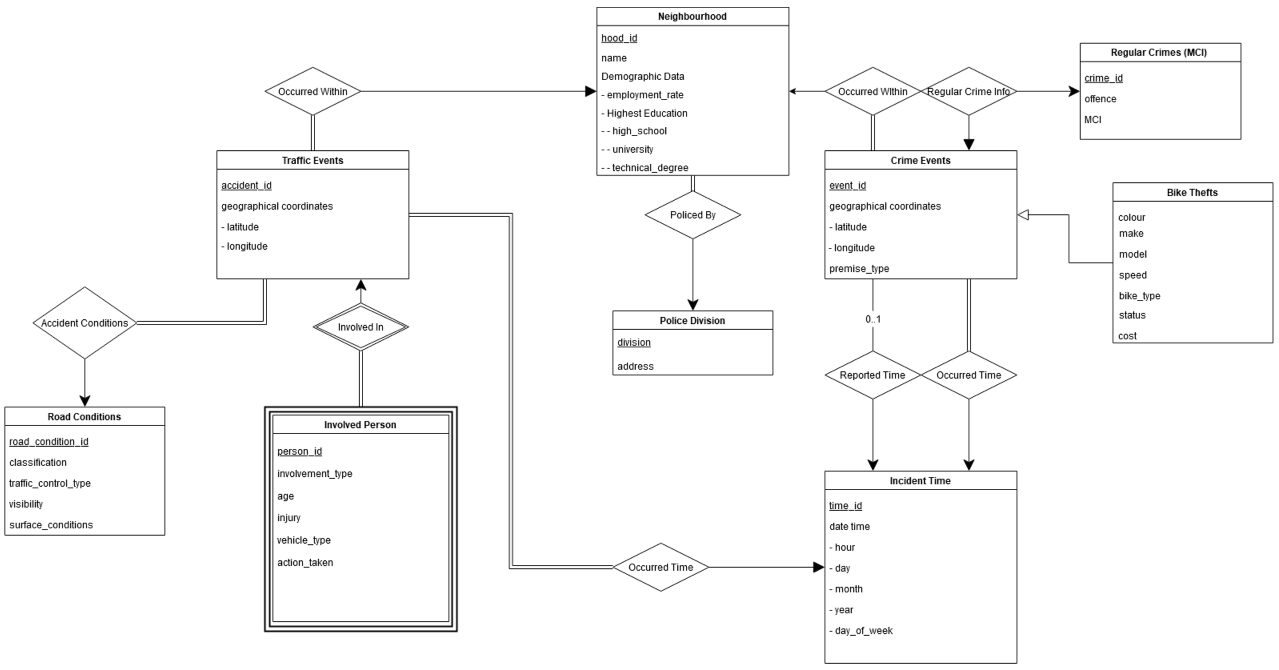
Design database schema

List of assumptions

- all reported times of incidents are either in or after 2014

- for regular crimes and traffic accidents, the time and place of the incident are recorded accurately and not left out (not NULL mostly)
- the level and standard of reporting crimes is consistent in all neighbourhoods (required for useful analysis and comparison among neighbourhoods)
- the cause of the traffic accident is taken from testimonies of all parties involved and is not one sided (although hard to avoid survivorship bias)
- the police arrive at the scene of an accident within reasonable time to be able to assess the road conditions leading up to the cause of the accident
- all parties involved in an accident are accounted for (driver, passengers, pedestrians)
- the status of stolen bikes is updated as best as possible (most bikes are never recovered)

E/R diagram



Note: Check the standalone image of the ER diagram if this one is too small.

Relational Data Model

- IncidentTime
  - <u>time_id</u> : INT
  - hour : INT
  - day : INT
  - month : INT
  - year : INT
  - day_of_week : INT

- PoliceDivision
  - division : INT
  - address : VARCHAR(50)
  - area : DECIMAL(12, 9)
  - shapeLeng : DECIMAL(12, 6)
  - shapeArea : DECIMAL(11, 3)
- Neighbourhood
  - hood_id : INT
  - name : VARCHAR(50)
  - employment_rate : DECIMAL(4, 2)
  - high_school : DECIMAL(4, 2)
  - university : DECIMAL(4, 2)
  - technical_degree : DECIMAL(4, 2)
  - division : INT (Foreign Key Referencing PoliceDivision(division))
- BikeTheft
  - event_id : INT (Foreign Key Referencing CrimeEvent(event_id))
  - colour : VARCHAR(50)
  - make : VARCHAR(50)
  - model : VARCHAR(50)
  - speed : VARCHAR(50)
  - bike_type : VARCHAR(50)
  - status : VARCHAR(50)
  - cost : DECIMAL(8, 2)
- RegularCrime
  - crime_id : INT
  - offence : VARCHAR(50)
  - MCI : VARCHAR(50)
- CrimeEvent
  - event_id : INT
  - occurrence_time_id : INT (Foreign Key Referencing IncidentTime(time_id))
  - reported_time_id : INT (Foreign Key Referencing IncidentTime(time_id))
  - crime_id : INT (Foreign Key Referencing RegularCrime(crime_id))
  - hood_id : INT (Foreign Key Referencing Neighbourhood(hood_id))
  - latitude : DECIMAL(10,8)

  - longitude : DECIMAL(11,8)
  - premise_type : VARCHAR(50)

- InvolvedPerson
  - <u>accident_id</u> : INT (Foreign Key Referencing TrafficEvent(accident_id))
  - <u>person_id</u> : INT
  - involvement_type : VARCHAR(50)
  - age : INT
  - injury : VARCHAR(50)
  - vehicle_type : VARCHAR(50)
  - action_taken : VARCHAR(50)

- RoadCondition
  - <u>road_condition_id</u> : INT
  - classification : VARCHAR(50)
  - traffic_control_type : VARCHAR(50)
  - visibility : VARCHAR(50)
  - surface_condition : VARCHAR(50)

- TrafficEvent
  - <u>accident_id</u> : INT
  - occurrence_time_id : INT (Foreign Key Referencing IncidentTime(time_id))
  - road_condition_id : INT (Foreign Key Referencing RoadCondition(road_condition_id))
  - hood_id : INT (Foreign Key Referencing Neighbourhood(hood_id))
  - latitude : DECIMAL(10,8)
  - longitude : DECIMAL(11,8)

Application Overview

- Starter Question
  - Located at very top of the application .
  - It filters the information in the cards below by crime type (regular crimes, bike thefts and traffic accidents).
  - For crimes we have filters for the major crime indicator (MCI), date (year or month) and location (citywide, neighbourhood and police division).
  - By default, it is set to show all crimes citywide (Toronto) that happened in 2019.
  - Click OK and the userr new search will update the cards below.
- Data Cards

  - All queries have similar cards with different data.
  - Table Card
    - Located right below the Starter Question.
    - Contains the raw data from the query.
    - Paginated and filterable.
  - Heat-map Card
    - Located right below the Table Card.
    - Shows the amount of crimes per time period.
    - Interact to refine the time range.
  - Summary Card
    - Shows the number of crimes per major crime indicator.
  - Cluster Map Card
    - Shows each individual crime location.
    - We can zoom in on the cluster to break it up and see individual locations.
  - Line Chart Card
    - Shows the number of crimes per month if our date type is set to year, or day if our date type is set to month.
  - There are a few other data cards besides the ones mentioned above: horizontal bar char, a doughnut chart, and a pie chart.
  - Note that, since it may be hard to view certain small numbers, the user can click on a chart label to remove it.
  - Bike Thefts and Traffic Incidents
    - Have the same cards.
    - As different types of data are collected for different types of crimes, different types of data are mentioned in the above data cards.

- Police Division Map
  - Located below everything mentioned above.
  - Shows the amount of crimes per police division and where the police divisions are located.
  - Provide an address and the application will output the closest police division.
- Additional Filters
  - A few filters are used in the application.
  - For example, we can look at the robberies that happened in rexdale-kiplin in March 2018.
  - We can also filter by a police division instead.
- Report Crime
  - Located at the bottom right of the application.

  - Fill this out to report a crime and update the database.
  - Please recall that the userr new crime will be added to 2020 (so update the filter before checking!).

- Batman Mode
  - The button to activate it is located at the top right of the application.
  - A game that tests how well the user know the city of Toronto and its crime.
  - Once activated, the user has to guess the total number of crimes that occurred and the crimes per month/day for the selected query.
  - Then at the end, users can see their score for guessing both the data.
  - This is a fun way for users to explore the data and challenge their friends.

- Predefined Queries
  - Located at the same spot as the Starter Question.
  - An alternative to the Starter Question if the user is not sure what they are looking for.
  - Select the toggle to the left of the Starter Sentence to switch to Predefined Queries.
  - Currently we have 21 queries, but more can be added quite easily.

Changes/Improvements Since Milestone 2

- Created additional indices to speed up queries for the predefined questions feature (2.4 seconds to 0.6 seconds!):
  - CREATE INDEX MCI ON RegularCrime(MCI);
  - CREATE INDEX BikeType ON BikeTheft(bike_type);
  - CREATE INDEX InvolvedPersonType ON InvolvedPerson(involvement_type);
- Implemented all remaining features and polished application