A Sustainable Path for Delivery

DataSci 205

Summer 2024, Section 7

By: Kenneth Hahn, Matthew Paterno, Victoria Brendel

Business Case

- As AGM continues to grow, we will eventually want to move away from PEAK delivery services to developing our own delivery system catered for our product.
  - No longer have to pay third party services or depend on third party data.
  - Customers have direct relationship with AGM and we have control over reliable delivery
- Because the Bay Area is our most popular location, we would like to develop a delivery service that is eco-friendly and speedy.
  - Use a system of BART and e-bikes.
- With NoSQL, we can develop the databases required to help find delivery routes and analyze key aspects of our routes.
  - Neo4j: Find shortest path between stores and customers and find new optimal store locations.
  - MongoDB: Store complex delivery and customer information
  - Redis: Compute and store real-time traffic data.

Neo4j

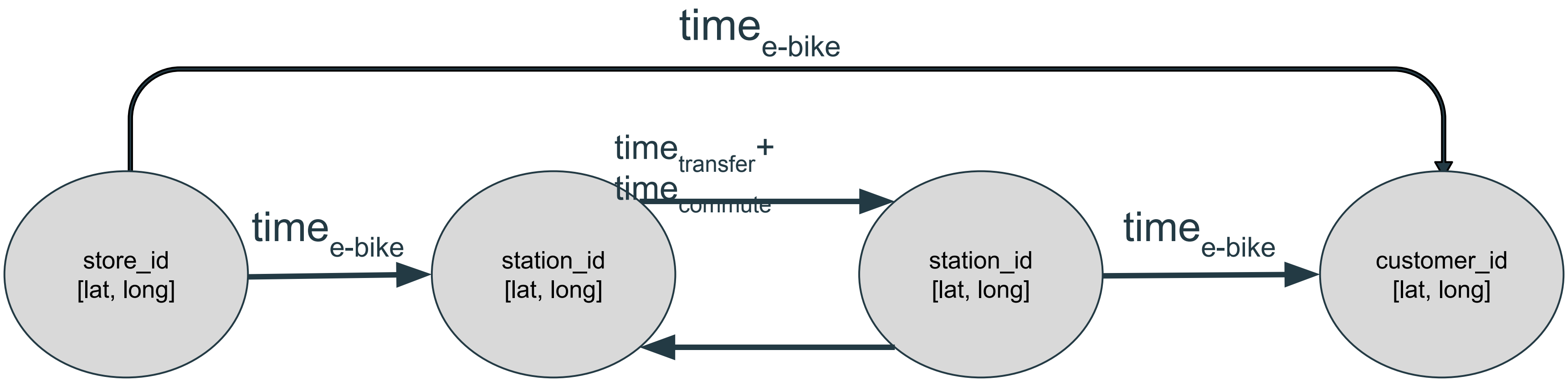
Overview

- Neo4j is a NoSQL graph database used to relate nodes and edges.
    - Node: A thing or object that you'd like to represent (e.g. a person, an airport, a router, etc.)
    - Edges: A relationship between the nodes (e.g. mutual friends, distance between airports, data flows, etc.)
- Much more useful over a relational database when it comes to handling relationships.
    - Relationships in a graph database are stored alongside nodes. No need for complex **joins** that will be slow and cumbersome.
    - A graph can be intuitive when viewing how things relate to one another as opposed to multiple tables.
    - Relationships and nodes can be updated independently and easily.

Application

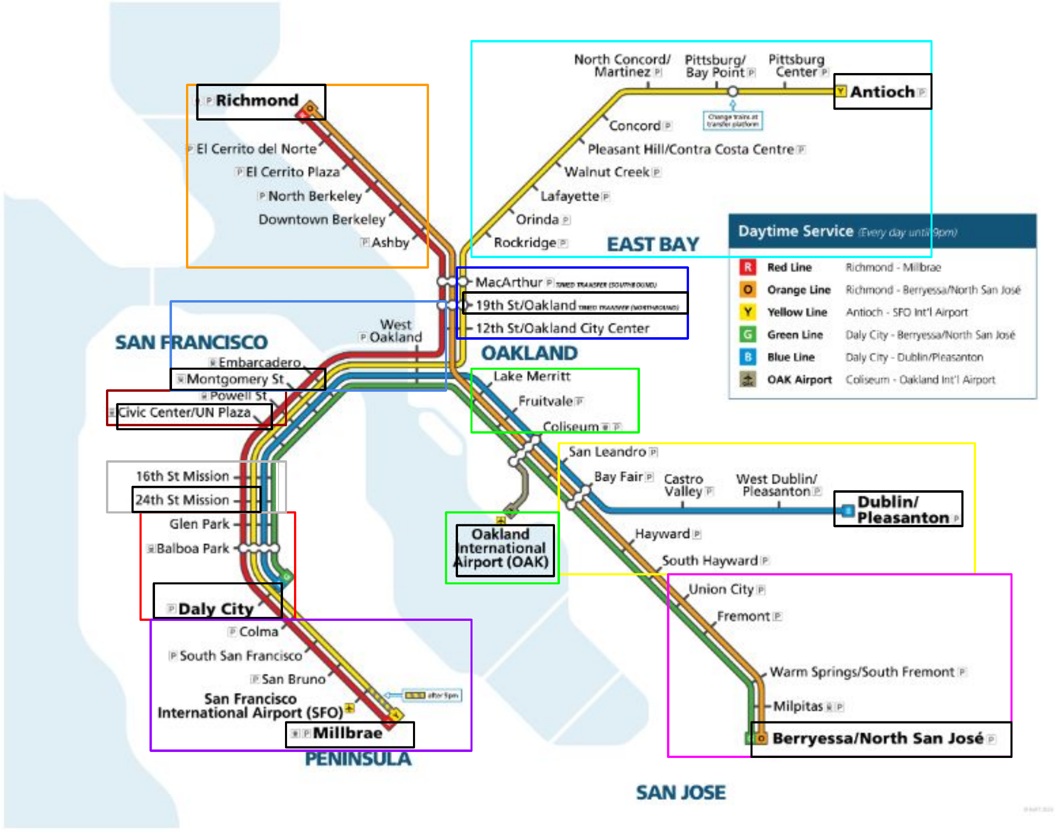
- We can use Neo4j to create a graph of all customers, stores, and BART stations, where they are related by travel times.
- We can update the relationships (travel times) easily to find shortest delivery path between a store and a customer either by e-bike, BART, or a combination of both.
- We can use this data to find optimal store locations if our customers are not getting their food quickly enough.

Neo4j General Graph Layout



- To find $time_{e\text{-}bike}$, we used Google Maps Direction API to find the *duration* from one [lat, long] coordinate to another, with the *biking* travel mode.
  - We used a correction factor of 1.5 to the *duration*, assuming that e-bikes are 50% faster than normal bikes.
- store_ids, station_ids, and customer_ids were taken from the stores, stations, and customers databases respectively in PostgreSQL.
- Customer [lat, long] coordinates were determined by choosing a random [lat, long] from a zip code "box" they were located in.
- Because this is a proof of concept, we only used 50 customers (randomly sampled from the list of 8138 customers) to reduce computation time and number of API calls.

Louvain Modularity and Betweenness Centrality



- Use Louvain Modularity on Stations to determine which stations can be clustered together.
    - Each cluster can be associated with one store.
- Within each cluster, use betweenness centrality to find the nodes with lowest betweenness (stores that are hard to get to).
- Result led to 11 communities.
    - The map on the left shows the communities where each colored box is a different community.
    - Black boxes are stations with the lowest betweenness in each community.
- What if we don't have the capital to have 11 store locations? Can we reduce the number?

Find New Store Locations

	finalnode	mean_travel_time_minutes	count_customers	percent_customers
22	arrive West Dublin	80.400000	1	2.0
17	arrive North Concord	80.272222	1	2.0
20	arrive Richmond	61.666667	1	2.0
13	arrive Hayward	59.933333	1	2.0
7	arrive Daly City	57.711111	1	2.0
12	arrive Glen Park	57.227778	2	4.0
21	arrive Walnut Creek	54.636111	4	8.0
3	arrive Balboa Park	52.861111	2	4.0
6	arrive Concord	50.561111	1	2.0
4	arrive Bay Fair	47.211111	1	2.0
18	arrive Pleasant Hill	42.550000	1	2.0

Around 32% of customers have delivery times that will be longer than 40 minutes with one store.

Legend

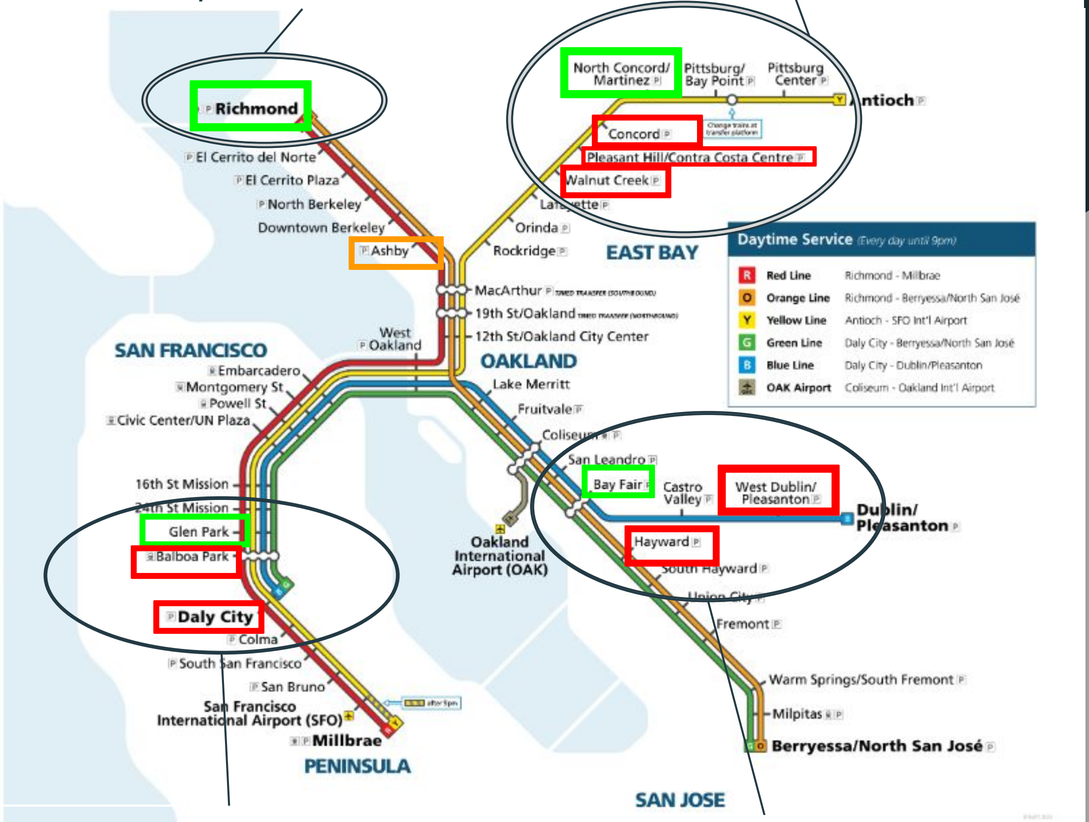
= New Store Location

= Existing Store Location

= Station that cannot reach in < 40 min

~2% of customers impacted

~14% of customers impacted



~10% of customers impacted

~6% of customers impacted

Recalculating with New Stores

~2% of customers impacted

	finalnode	mean_travel_time_minutes	count_customers	percent_customers
3	Store North Concord	41.322222	1	2.0

Now only one station has a mean travel time of > 40 min

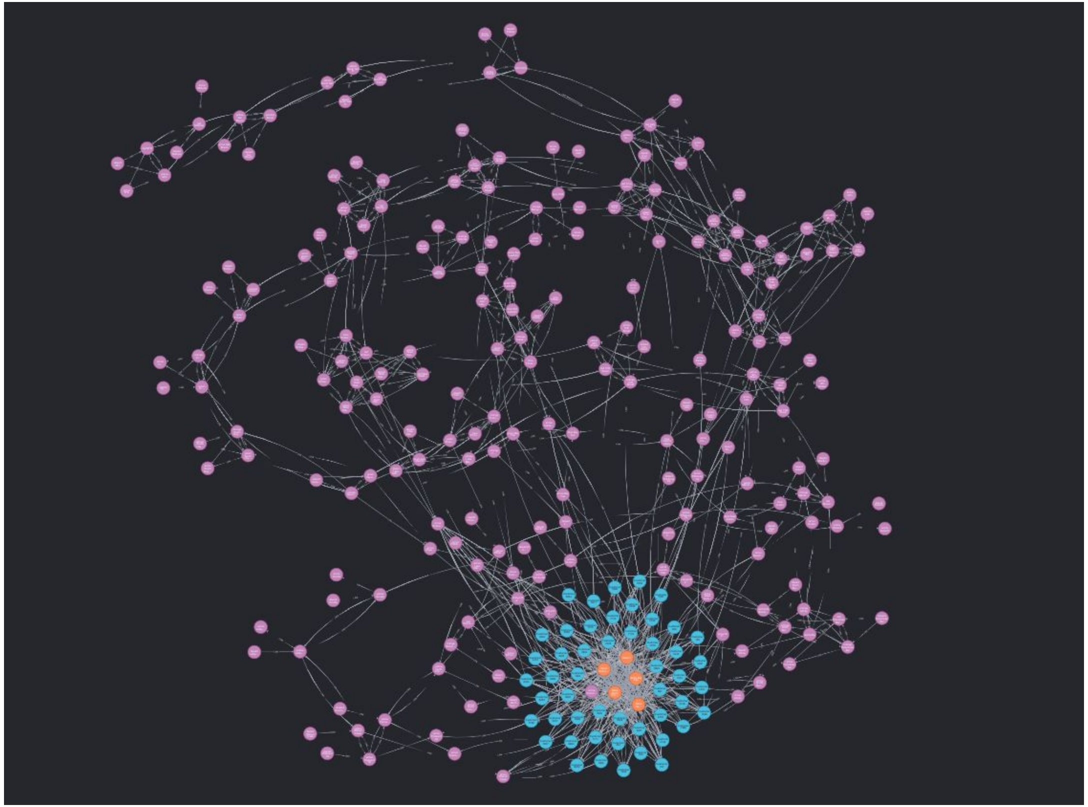
This customer lives very far from any BART station (takes 40 min by e-bike)

This is deemed acceptable for our delivery purposes

	store	customer	totalCost	nodes	costs	finalnode
167	Store North Concord	Customer 6697	41.322222	[Store North Concord, Customer 6697]	[0.0, 2479.3333333333335]	Store North Concord

Final View of Neo4j Graph

- Stores: 5 nodes
- Stations: 214 nodes
- Customers: 50 nodes
- Relationships: 1262



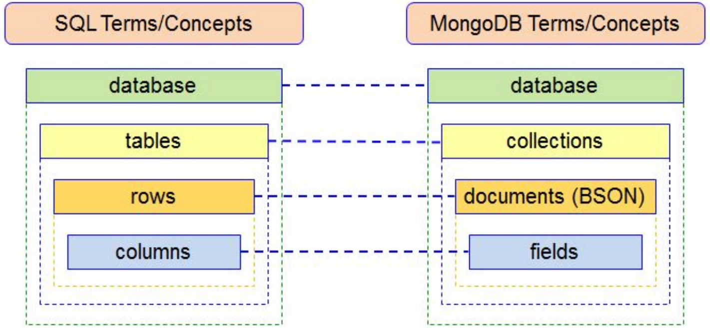
MongoDB Overview

- MongoDB: a NoSQL database designed for storing large volumes of diverse data
    - document-oriented, flexible schema, high scalability and performance
- Why MongoDB
    - Handles diverse data types (customer data, route information, etc.)
    - Allows for quick read and write operations, essential for real-time data access
    - Scales horizontally, supporting the growing data as AGM expands
- Potential disadvantages
    - Lack of fixed schema - can lead to inconsistencies in how route, customer, and delivery data are stored
    - Performance and scalability - MongoDB can consume significant resources, particularly memory

MongoDB application

Scenario: Storing main routes, customer information, and delivery history

- Since we have 8138 customers, MongoDB provides a flexible solution to store customer data - name, address, delivery preference, etc.
- With a combination of bart and E-bikes, the complex delivery method can be stored in MongoDB - detailed route information, coordinates, delivery times



Suggested structure

- Collection 1: Routes
  - Fields: route_id, start_point, end_point, waypoints, estimated_time
- Collection 2: Customers
  - Fields: customer_id, name, address, contact_info, delivery_preferences
- Collection 3: Delivery History
  - Fields: delivery_id, customer_id, route_id, delivery_time, status

Example queries in MongoDB

Code:

```
result = db.delivery_history.aggregate([

    { '$match': { 'route_type': 'E-Bike' } },

    { '$group': { '_id': None, 'totalDistance': { '$sum': "$distance" } } }

])

for doc in result:

    print(doc)
```

Purpose:

Calculates the total delivery distance covered by e-bikes by summing the distance field for all e-bike delivery records

Code:

```
result = db.delivery_history.aggregate([

    { '$group': { '_id': "$customer_id", 'averageDeliveryTime': { '$avg': 
"$delivery_time" } } },

    { '$sort': { 'averageDeliveryTime': -1 } }

])

for doc in result:

    print(doc)
```

Purpose:

Calculates the average delivery time for each customer and sorts the results to identify customers who experience longer delivery times

Redis

Redis is an in-memory data structure store that stores key-value pairs in RAM. This makes lookups extremely fast, ideal for real-time systems.

Use Redis for common, frequent lookups.

Notable products that leverage Redis: Uber, Pinterest, Airbnb.

Disadvantages:

- Cost of in-memory storage
- Volatility
- Design limitations makes it difficult for graph computations.

Possible applications

Real-time traffic monitoring

- Share data used for computations across customers can be stored in Redis for quick access.
- Example: Store real-time traffic data that for specific paths that we can use across our travel-time computations for determining the quickest real-time route.

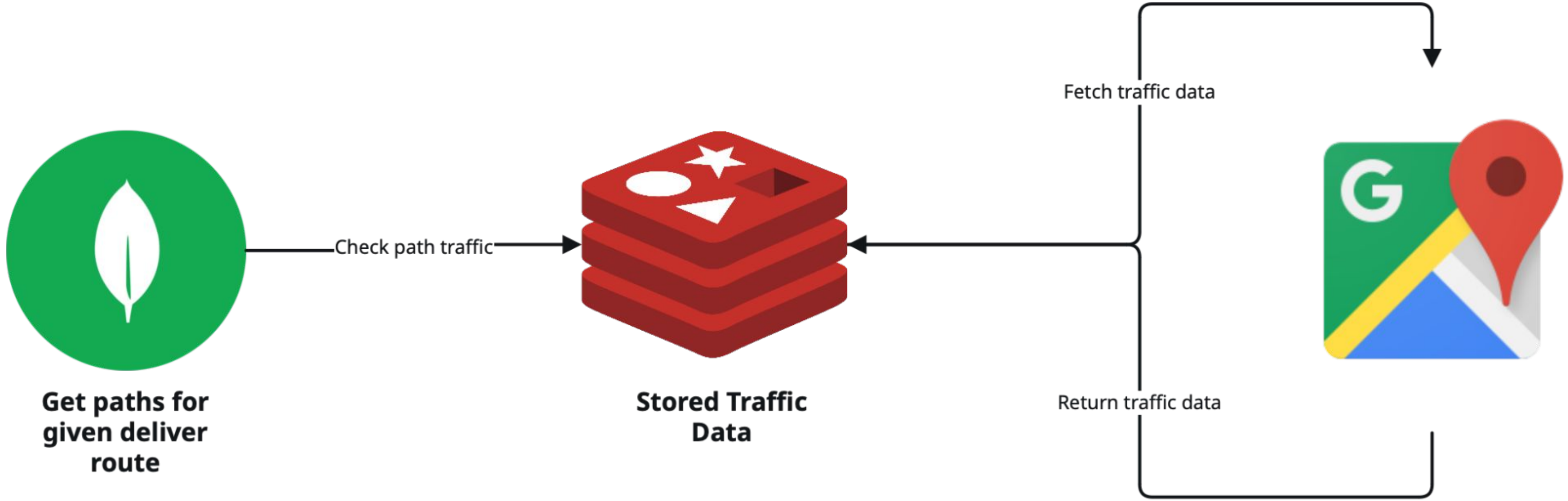
Speed up model training

- If we want to train on our customer data we may choose to have our models interface with our cache layer to speed up the training time.
- We can also store computed features in Redis for quicker access during model inference stages, allowing for fast computations even for a model with many features.

Example Query

```
SET traffic:location:12345
"{\"status\": \"congested\",
\"speed\": 20}"
```

Storing traffic data for stored delivery routes



Conclusion

Adopting modern DBs allows us to better meet customer and stakeholder needs.

- Neo4j empowers our team to easily perform graph based computations.
- MongoDB is a scalable NoSQL alternative that simplifies querying.
- Redis unlocks real-time computations like traffic monitoring that could directly benefit the customer experience.

x