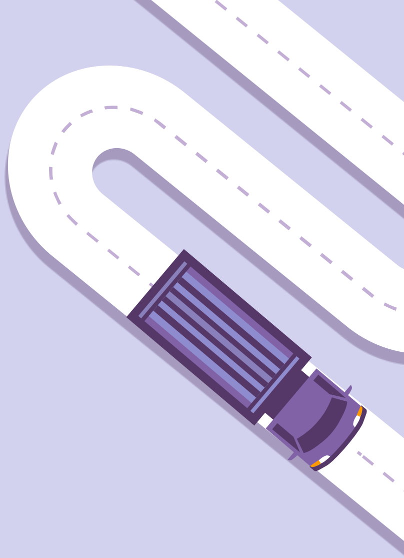
Implementation of NoSQL for Acme Gourmet Meals

Project 3 - Aaron Shulkin, Mohammad Kanawati, Nayan Ganguli, Nicholas Luong



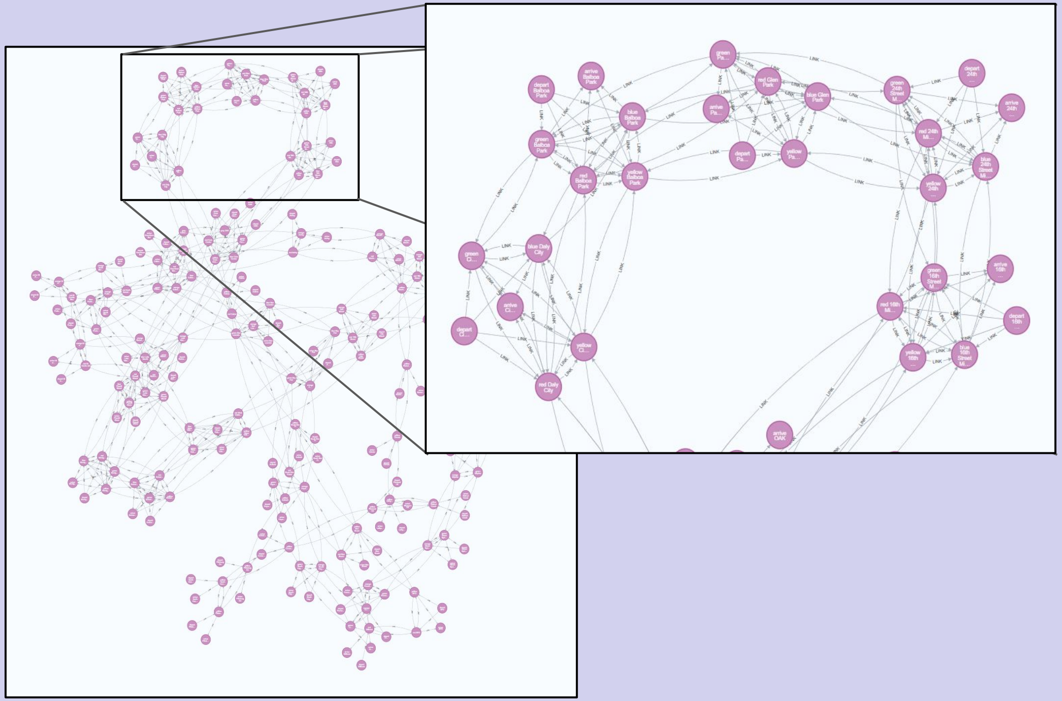
Agenda

Pre-Delivery – Neo4j (*Dijkstra's Single-Source Shortest Path*)

During Delivery – Redis

Post-Delivery – MongoDB | Neo4j (*Harmonic Centrality | Louvain Modularity*)

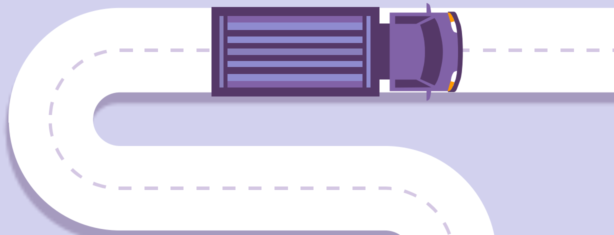
Pre-Delivery - Neo4j


Neo4j Graph representing our delivery map and routes

Using Neo4j, we can implement Dijkstra's Single-Source Shortest Path to optimize our routes and find the quickest paths

This allows us to develop a path on how they should best utilize BART to arrive on time

Traditional databases lack the nuance to capture the complex relationships for each destination (nodes) and their various potential paths



During Delivery - Redis

Real-time Tracking and Caching:

Redis excels in real-time data processing and caching.

It can be employed to store and quickly retrieve real-time information about delivery vehicle locations, status, and other dynamic data.

This allows for instant updates and optimizations in response to changing conditions, ensuring timely deliveries.

Compared to Traditional DB:

Redis operates primarily in-memory, allowing for extremely fast data access and retrieval.

This is crucial for real-time scenarios, where quick access to dynamic data, such as vehicle locations and delivery statuses, is essential for efficient route planning and optimization.

Post-Delivery - Neo4j

We can utilize the harmonic centrality algorithm with Neo4j to identify the importance of nodes based on their harmonic mean distance to other nodes in the network

Below are examples of how we can implement this for reviewing future deliveries:

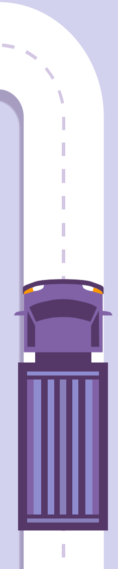
Used to identify the most strategic pickup locations at BART stations

↓

Key pickup locations become central hubs, attracting a steady flow of customers

↓

This strategic placement contributes to customer satisfaction, potentially increasing business



Post-Delivery - Neo4j

We can also implement the Louvain Modularity algorithm with Neo4j to detect communities in networks using travel time as weight.

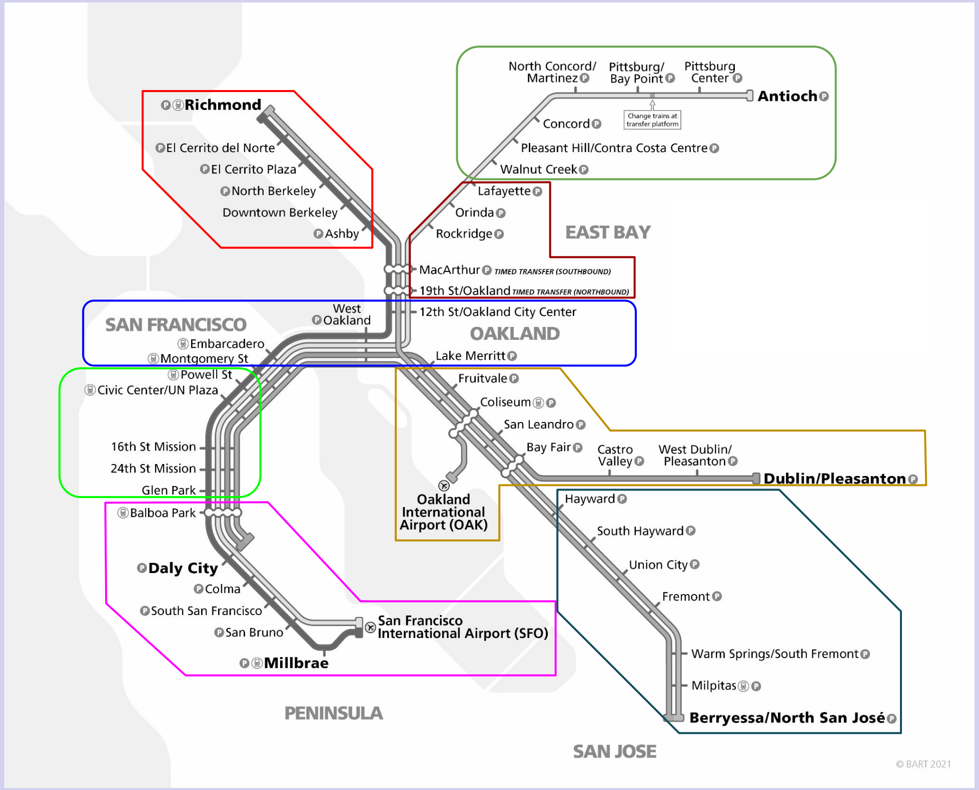
We apply a modularity score to evaluate the density of connected nodes within a community and we can recursively merge these communities into nodes

With the new nodes, we can do the following:

Reduce redundant stations to optimize pick-up locations

Set up a single docking station for our drone and robot fleets inside these communities, as travel times between stations are minimal.

Further run centrality algorithms to identify a central station inside each community



Post-Delivery - MongoDB

Documenting and Updating Locations:

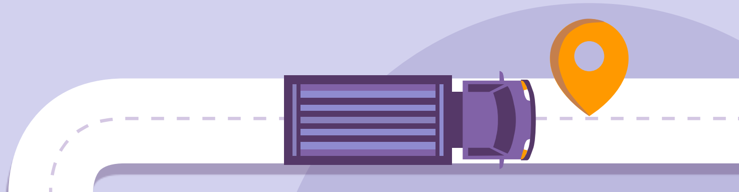
MongoDB can store information about various pickup locations, including BART stations and other strategic points.

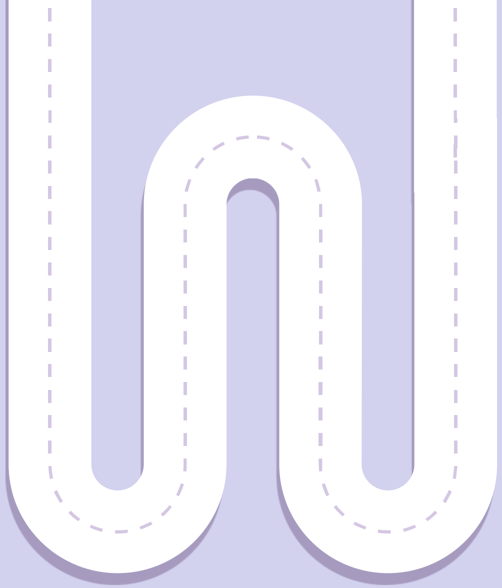
The document-based structure allows flexibility in capturing details such as location coordinates, pickup schedules, and available transportation modes (e.g., traditional trucks, drones, public transportation).

Compared to Traditional DB:

As the delivery network evolves with the addition of new pickup locations and transportation modes, MongoDB's dynamic schema accommodates changes without requiring a predefined structure.

This flexibility is crucial in adapting to the dynamic nature of the delivery business.





THANK YOU!