A Hybrid Database Solution to Support Transportation Analytics with Case Studies on Improving Emergency Medical Service Response Times

by

Bryan Wodi

A thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada

August 26, 2022

Thesis advisor Author

Dr. Carson K. Leung **Bryan Wodi**

A Hybrid Database Solution to Support Transportation Analytics with Case Studies on Improving Emergency Medical Service Response Times

Abstract

Using a single type of database solution to support real-world applications is becoming more and more challenging because of the volume and variety of data. For instance, the data collected for the transportation industry include both structured and unstructured data. Using solely a single type of database solution—relational database system-only or graph database-only—to store and manage data can be challenging. As real-world applications ask even more complex questions related to data, the database solution should be able to facilitate answering these questions in a reasonable time. Hence, for my MSc thesis, I present a hybrid database solution to support transportation analytics. The hybrid solution consists of relational and non-relational databases, pooling their strengths to support the demands of the modern application. I also demonstrate this as a practical solution with two case studies on improving emergency medical services (EMS) response times by having the support of the presented hybrid database solution. My key contributions to this thesis include (a) the design and implementation of the hybrid database for supporting transportation analytics, (b) algorithms to extract and convert connected linear shapefiles into a graph network, and (c) two case studies on improving EMS response times.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I want to begin by thanking my advisor, Prof. Dr. Carson K. Leung, for supporting me throughout my journey as a graduate student. I also thank my thesis advisory and examination committee members, Dr. Noman Mohammed (Computer Science) and Dr. Jonathan D. Regehr, P.Eng. (Civil Engineering), for your support and time in reading and examining my thesis and defence. Thanks Dr. Shaowei Wang for chairing my defence.

I want to thank Mitacs and TRAINFO for their financial support for this research. I would also like to thank Neil Ternowetsky, Chief Technology Officer of TRAINFO Corp., for his support and mentorship on this thesis and in the railway safety transportation industry. Alongside Neil are the rest of my colleagues at TRAINFO Corp. Another important colleague I would like to thank is Maryam Ghaffari-Dolama, whom I worked closely with on this project. My contribution seeks to support her alongside TRAINFO's research.

Finally, I thank my family and friends for their support and encouragement throughout this journey. Most significantly are Nosa Clifford, Anna Oppa, and Gift Amadi.

Bryan Wodi

B.Sc., The University of Manitoba, Canada, 2018

The University of Manitoba
August 26, 2022

This thesis is dedicated to my mothers, Vivian and Hilda; they are the reason I have come this far, and continue to push myself.

Chapter 1

Introduction

Data mining aims to extract "implicit, previously unknown, and potentially useful information" from large amounts of data [FPS96]. In recent years, the advances in data storage and communication technologies have led to increased availability and diversity of data [CHB$^+$20]. In the transportation industry alone, we have multiple data sources coming from, but not limited to, global positioning systems (GPS), Wi-Fi/Bluetooth sensors, road condition assessment reports, and data associated with land use. These many forms of data underscore the need to extract previously unknown and potentially useful information. Therefore, we need new capabilities for data extraction, storage/management, and dissemination systems [GTH$^+$17].

Knowledge discovered from these rich sources of information can significantly impact the results of any analysis. For example, when dealing with Emergency Medical Services (EMS) transportation, knowing of a new factor that might affect emergency responders' travel time can be crucial when making decisions. Hence, it will be very beneficial to researchers and other stakeholders for managing traffic systems and conducting traffic analysis [CHB+20].

Unsurprisingly, the process of data mining strives towards deriving useful information from raw data. The gained information help to make better predictions and identify patterns to facilitate the decision-making process [KSA16]. Such facilitation in decision-making could yield substantial benefits in EMS. EMS administrators seek methods to minimize response time (RT) (i.e., the time between reporting an incident and when first responders arrive at the scene [BK02]). According to some studies [BK02; hea12; PA20], the risk of fatality remains very low if affected individuals receive medical attention within 4-8 minutes of an emergency. Hence, following the National Fire Protection Association (NFPA) 1710 standard [Nat21], the Winnipeg Fire and Paramedics Services (WFPS) set their aim that the first EMS unit responding to an emergency must arrive at the scene within four minutes.

In reality, there are several challenges [BK02; PJY+16; MPS+19; PA20] that could hinder an EMS vehicle from arriving at the scene within this established time frame. For the WFPS, the most challenging factor is delay or blockages at active grade-level crossings (subsequently referred to as crossings) when responding to calls [Dul15]. An *active crossing* is a crossing blocked by a moving or stopped train. At any rate, emergency responders are delayed daily by trains at active crossings, costing them precious minutes in a time-critical situation. Delays at crossings are usually caused by freight trains stopping or moving slowly at those crossings [CR17; US 19]. That being the case, one would think a trivial solution is to avoid active crossings if there were predictions for the blockages. However, predicting the arrival times of freight trains at crossings is also really difficult [APR+16; PPP+19], hence, very difficult to avoid.

As a result, these blockages are a significant cause for concern, especially for Winnipeg, which, according to the Government of Canada crossing inventory [Gov18], has the highest number of crossings within any Canadian city. While being delayed at a crossing could be nothing more than an inconvenience for many, emergency cases (as it usually is for EMS) could mean life or death. According to Deputy Chief Jo Zambito from the Niagara Falls Fire Department [Zam20], they are aware of the risk of being delayed at crossings. Because of this risk, their default response navigation strategy is to use an alternative route that does not interact with crossings. Although the alternative route may add an *extra two or three minutes to their travel time*, they consider it acceptable. However, in cardiac arrest emergencies, two minutes could be the difference between life and death [hea12].

Therefore, the WFPS increasingly faces the need for a personalized mobility solution that ensures lower risks of delays at crossings when responding to emergencies. So, having a model to assess their risks of being exposed to such delays on their route would immensely reduce travel time — saving lives. Knowing the risk of exposure would benefit their decision-making when navigating to call locations. As a result, a data management system that can support this type of analysis would be crucial in helping the WFPS mitigate their risks of exposure. This data management system would allow for a quick and efficient generation and storage of these models.

We contend that using one kind of database is not a satisfactory solution. While it is possible to use a non-relational database to perform transportation analysis, this thesis shows that it is inefficient. However, by leveraging the strengths of relational and non-relational databases, we demonstrate this as a model solution to support modern-day use for EMS transportation analytics.

1.1 Thesis Statement

There are some bodies of work [ZBP^{+}18; SGB20; VAOA22] that have used relational and non-relational databases to support one another, popularly called the *hybrid* approach. Although, few have shown real-world or large-scale use. The approaches that show real-world applications use difficult-to-retrieve data sources [ZBP^{+}18; Bes22] – making these approaches non-reproducible. The others [SGB20; VAOA22] are not in the transportation industry.

For this MSc research, the goal is to design a hybrid database management system (DBMS) that combines the strengths of relational and non-relational databases to support transportation analytics. Our key contributions include (a) the design and implementation of the hybrid database for supporting transportation analytics, (b) algorithms to extract and convert connected linear shapefiles into a graph network and (c) two case studies on improving emergency medical service response times. Questions to be answered in this thesis:

Q1. How well can the hybrid DBMS handle traditional structured data (e.g., non-geo-spatial data)?

Q2. How well can the hybrid DBMS handle semi-structured and unstructured data (e.g., geo-spatial data)?

Q3. How well can the hybrid DBMS handle some real-life applications with structured, semi-structured, and unstructured data?

1.2 Thesis Organization

This thesis is structured the following way. We provide more background information on hybrid databases and various technologies used throughout this thesis in Chapter 2. We further describe the current dispatch process of units during an emergency by the WFPS.

Chapter 3 contains a review of related work in DBMS in general and the transportation industry. We highlight the desired features in both kinds and databases and summarize how this thesis research compares to existing work.

In Chapter 4, we describe the hybrid database design. We go into detail about the datasets that we used throughout the system. We also show the system architecture and how the different components interact for optimal performance.

Chapter 5 describes two real-life case studies on how we used the hybrid database to improve emergency first responders' response times. We showed how the results from these case studies could help dispatchers make better routing decisions.

Chapter 6 contains the evaluation outcomes from two single-style database approaches compared to our hybrid database solution. This chapter also discusses the conclusions we draw about using hybrid databases in supporting transportation analytics.

Finally, in Chapter 7, we give our research conclusion and discuss future work that one could do to extend this project on leveraging hybrid databases.

Chapter 2

Background

In this chapter, we describe and explain some essential terms, techniques and tools used in our thesis work.

2.1 Relational Databases

Relational databases have been around since the 1970s and are a common choice for most data-intensive storage and retrieval applications [VMZ$^+$10; VKT18]. They usually store data in rows with column names defining each cell's representation, similar to spreadsheets. We perform data retrieval using Structured Query Language (SQL) — a declarative query language.

Relational databases comprise relations, loosely described as a collection of tables containing rows of data with column headings [Leu22]. The rows (tuples) represent unique relationships between the column headings. Relational databases use the relational model, a mathematical way of structuring data and using data [Dar12; VKT18].

Table 2.1: A tabular depiction of the relation: Trips

incidentID	**timeStarted**	**timeArrived**	**Origin**	**Destination**
3666	2022-01-01 00:07:29	2022-01-01 00:11:34	123 Apple Way	555 New St.
3667	2022-01-01 14:00:38	2022-01-01 14:12:03	123 Apple Way	333 Old St.

Table 2.1 is a tabular representation of a relation. Here, we have a relation called Trips, with five attributes (incidentID, timeStarted, timeArrived, Origin, Destination) and two tuples — each row in the table.

Relational databases range from small, personal databases like Microsoft Access to large-scale database servers like Microsoft SQL Server and MySQL. The most common is MySQL [DE20] which is why we used it for our research.

2.1.1 MySQL

Because of its practical use in the industry and academic world, we chose MySQL for this research. Not to mention, there is also extensive documentation and support both from the manufacturers and the user community. MySQL performs well on storing and processing spatial data for Geographic Information System (GIS) [Lak18] applications, which is very useful when we need to perform specific spatial-related queries. The hybrid design used in this thesis leverages the speed of retrieval of structured data and specific spatial data.

2.2 Non-relational Databases

Non-relational databases do not conform to the relational model of storing and accessing data as a collection of rows and columns in tables [VMZ$^+$10; RS20].

The major criticism of relational databases is that they do not efficiently handle large volumes of data, especially when it is not entirely (or at all) structured. As a result, Not-Only SQL (NoSQL) emerged [VMZ$^+$10; AC18; MMR18] — the fundamental difference being that it does not have to conform to the constraints that we will further discuss in Chapter 3.

More so, there are four categories [Lak18] of storing data using the NoSQL approach, namely:

1. key-value storage, where we store data as a pair of keys and values. An example is Amazon's DynamoDB [Ama21].

2. column storage, we store data in columns associated with row keys. An example is Google's Bigtable [Goo21].

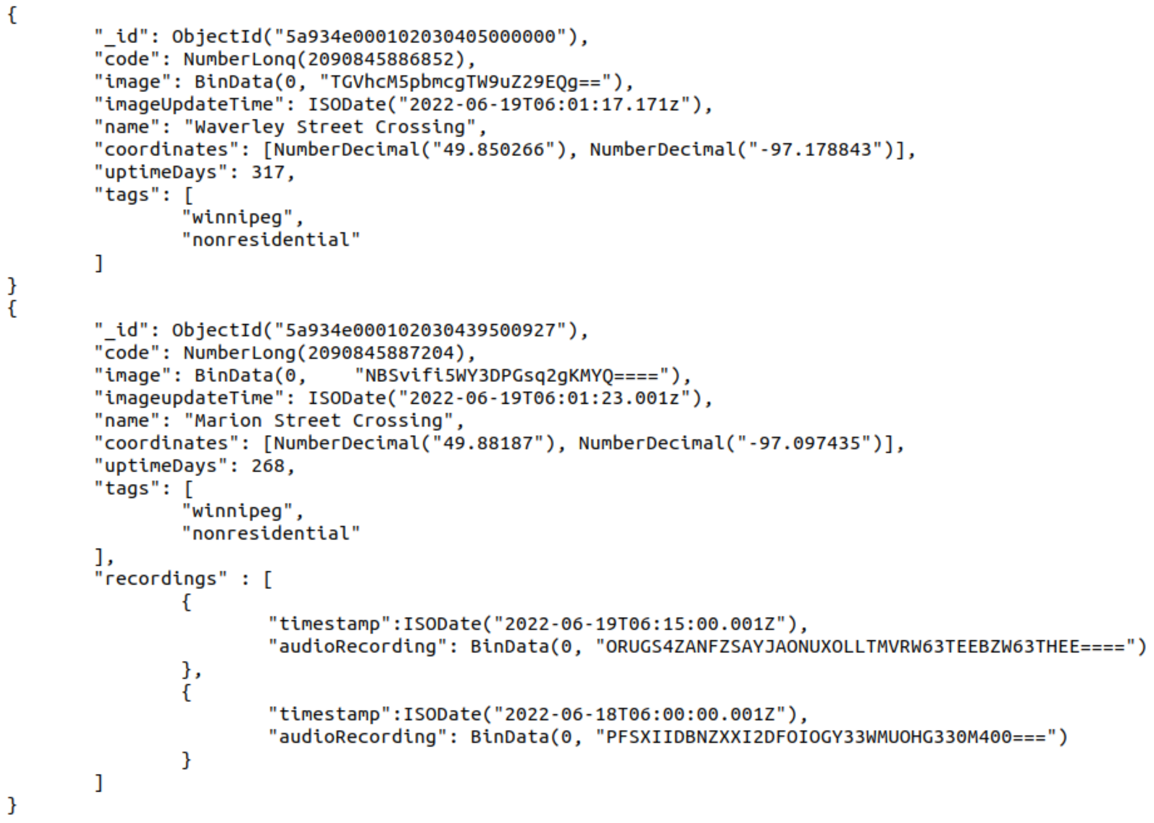
3. *document storage*, where we store data in a single document. An example is MongoDB [Mon20].

4. *graph database*, which uses a relation of nodes to form a network of nodes. An example is Neo4j [Neo20].

In this thesis, the categories that we use in our research are the document and graph databases.

2.2.1 Document-Oriented Databases

The document-oriented database model is based on key-values [Mon20; HELD11]. The data stored are documents in which they encode the data in a semi-structured format, the most popular being JavaScript Object Notation (JSON), Binary Javascript Object Notation (BSON), Extensible Markup Language (XML) or Portable Document Format (PDF) [Mon20; HELD11]. More so, a document stores information about an object, its primary identifiers and any of its related metadata [TRBB15]. A document database comprises a collection of documents [HELD11]. A document comprises fields and associated values, as seen in Figure 2.1.



```
{
        "_id": ObjectId("5a934e000102030405000000"),
        "code": NumberLonq(2090845886852),
        "image": BinData(0, "TGVhcM5pbmcgTW9uZ29EQg=="),
        "imageUpdateTime": ISODate("2022-06-19T06:01:17.171z"),
        "name": "Waverley Street Crossing",
        "coordinates": [NumberDecimal("49.850266"), NumberDecimal("-97.178843")],
        "uptimeDays": 317,
        "tags": [
                "winnipeg",
                "nonresidential"
        ]
}
{
        "_id": ObjectId("5a934e000102030439500927"),
        "code": NumberLong(2090845887204),
        "image": BinData(0,    "NBSvifi5WY3DPGsq2gKMYQ===="),
        "imageupdateTime": ISODate("2022-06-19T06:01:23.001z"),
        "name": "Marion Street Crossing",
        "coordinates": [NumberDecimal("49.88187"), NumberDecimal("-97.097435")],
        "uptimeDays": 268,
        "tags": [
                "winnipeg",
                "nonresidential"
        ],
        "recordings" : [
                {
                        "timestamp":ISODate("2022-06-19T06:15:00.001Z"),
                        "audioRecording": BinData(0, "ORUGS4ZANFZSAYJAONUXOLLTMVRW63TEEBZW63THEE====")
                },
                {
                        "timestamp":ISODate("2022-06-18T06:00:00.001Z"),
                        "audioRecording": BinData(0, "PFSXIIDBNZXXI2DFOIOGY33WMUOHG330M400===")
                }
        ]
}
```

Figure 2.1: Structure of a document-oriented database

A key advantage of document databases is that the schema is flexible because a document can contain other nested documents. For example, in Figure 2.1, the second document contains two nested documents in "recordings". In this example, the "recordings" documents hold audio recordings of a train at a crossing captured by a roadside sensor. This nesting structure makes it possible for documents of unique structures to be stored as part of the same set [TRBB15]. This advantage makes reading performance superior to relational databases, as one can, via a single key, retrieve a set of structured information hierarchically.

Although several implementations of document databases exist, the most popular implementations are Apache's CouchDB, RavenDB and MongoDB [DE20]. For our research, we go with MongoDB.

2.2.2 Graph Model

The graph model is a collection of two elements: a node (also called a vertex) and an edge [VKT18]. Each node represents an entity (student, railway crossing, or another piece of data), and each edge represents a relationship or connection between the two nodes.

Graphs have become one of the most valuable structures for developing data provenance systems [VMZ$^+$10]. The term provenance means lineage—how something came to be. Hence, graphs have become an increasingly popular choice for modelling objects and interactions—as seen with social networks and gene clusters in biology. More specifically, in the transportation industry, graph models are a popular choice for representing networks when conducting analyses [ZJH$^+$21].

For this thesis, we use a property graph as our data store. A **property graph** is a type of graph model that comprises nodes, edges and other properties [Ang18]. We model the data structures for a property graph schema as graphs. Hence, property graphs facilitate the modelling of entities, their relationships and properties because one does not need to develop a complete graph data model on the first attempt [PVSZ19].

Structurally, the nodes of the property graph have a label and a set of properties as key-value pairs. A relationship between two nodes is a directed edge between them. Similarly, edges have labels and can hold a set of properties in the same form as nodes [Ang18; PVSZ19]. We depict this in Figure 2.2.

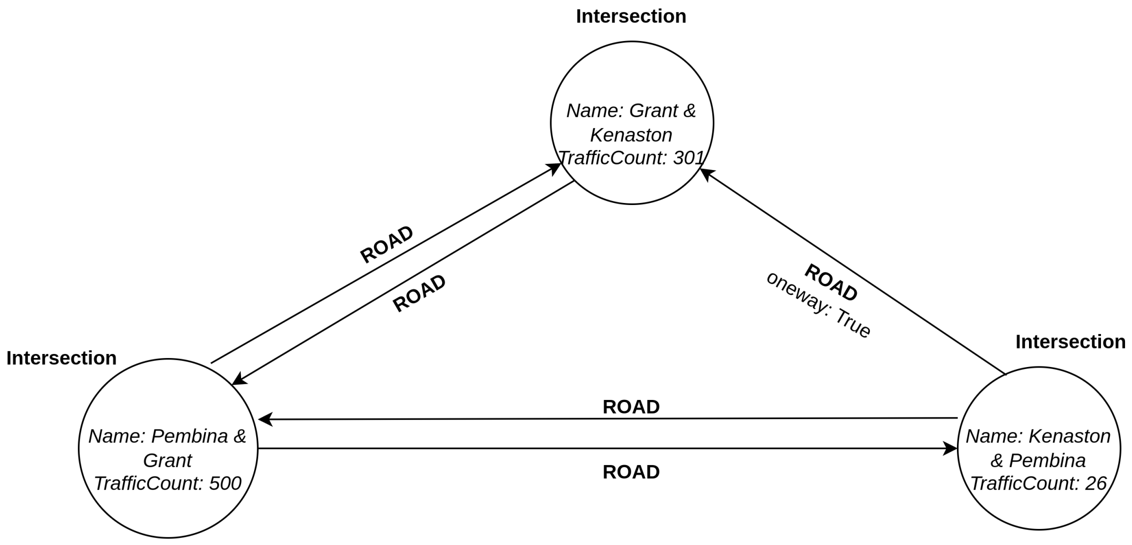


Figure 2.2: Example of a property graph representing the relationship between three intersections on a road transportation network

2.3 Emergency Response Process

The emergency response-time process involves call processing time (the time from receiving the 9-1-1 call to alerting emergency response stations or units), turnout time (the time from receiving the emergency alert to boarding the response vehicle), and travel time (the time that the vehicle begins travelling to the call location to the arrival time) [Nat21]. Based on the case studies to be discussed in Chapter 5, Figure 2.3 explains the overall process of responding to an emergency by the WFPS, which is like other emergency response processes in North America. Three parties are involved: call takers, dispatchers and responders [vKvB17]. Call takers are in charge of inbound requests and communicate with an emergency caller. Dispatchers are in charge of outbound calls to the ambulance teams, fire services and hospitals. It is important to note that dispatchers support an emergency event from when they are assigned to them until the case is closed.

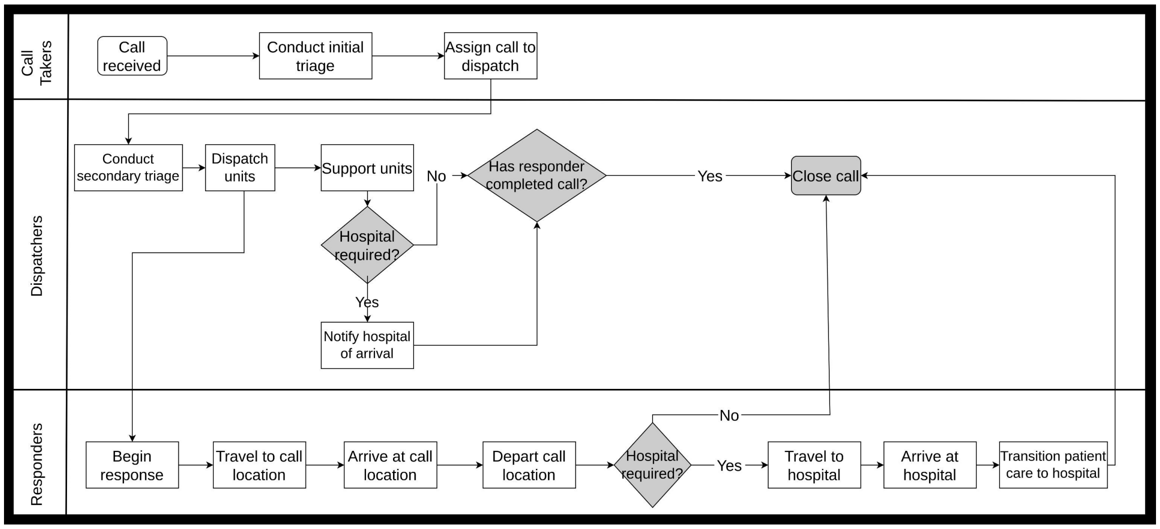


Figure 2.3: General overview of WFPS' emergency response process

The WFPS faces a huge complexity when dispatching units because of limited resources. There are several reasons for limited WFPS resources. The two most pressing reasons are:

1. EMS specialization: Different first responder units may have different licenses, allowing them to administer certain medications in emergencies. Certain specialized units must remain available for deployment.

2. Equipment access: Some fire trucks are more equipped for certain emergency scenarios than others. For example, WFPS possesses only five ladder-equipped fire trucks servicing the entire city, making their availability equally important.

Dispatchers aim to maintain reasonable response times and fleet capacity while responding to emergencies. The presence of an active crossing presents a challenging dilemma for dispatchers. For example, if a responder is stuck at a crossing, the dispatchers can instruct the unit to either:

1. turn around and go another way, or

2. stay put while the dispatcher sends another unit from the opposite side of the train to respond to the event

In the first scenario, the responder would have increased their response time. Even worse, in the second scenario, we have two units responding to a call that only needed one unit, reducing fleet capacity by two. For crossings notorious for causing delays, WFPS sends multiple units from different directions to respond just in case the crossing is active upon the unit's arrival.

2.3.1 Composition of an EMS Trip

To further the understanding of the analysis that our proposed database will support in Chapter 5, let us give a high-level description of WFPS' composition of responder trips which we referenced a lot in that chapter. We can further break an EMS trip down into multiple legs. A leg is a physical trip of a responder from one point to another. Depending on the call, a trip can compose two or more legs.

In order to understand the concept of legs, let us first review the spatial components of completing a call. An emergency responder completing a call can comprise the following spatial components:

1. Emergency response station: The station they assign the responder unit.
2. Call location: The location of the emergency response event.
3. Hospital: A facility a responder can deliver a patient to for transferring care.

Having reviewed the spatial components in Figures 2.4, 2.5 and 2.6, there are three primary trip compositions a responder would follow when responding to emergencies:

1. Trip originates from station: When a trip originates from a station, there are three possibilities:
   (a) the unit arrives at the location and then returns to the station if the call does not require advanced medical care and they do not dispatch the unit to another call
   (b) the call does not require advanced medical care, but when the call is closed at the call location, they dispatch the unit to another call.

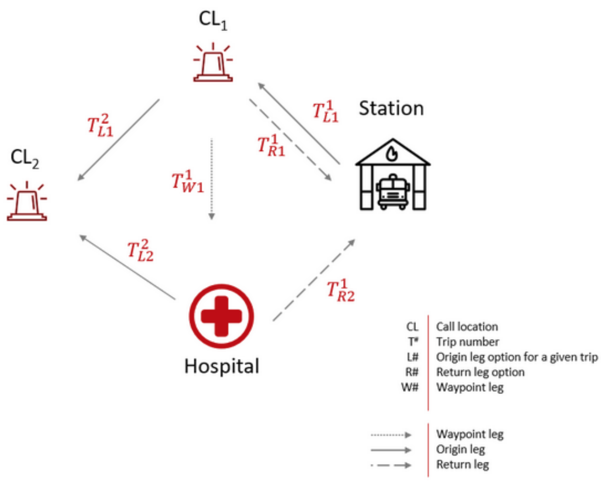


Figure 2.4: Response trip path originating from station

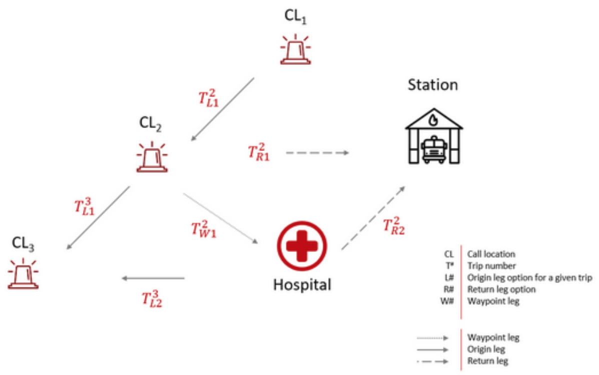


Figure 2.5: Response trip path originating from previous call location

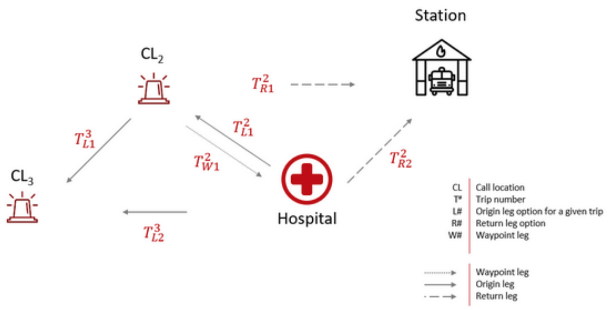


Figure 2.6: Response trip path originating from hospital

   (c) the call requires advanced medical care, so they transport patient(s) to the hospital. When responders transfer care over to the hospital, they return to the station; or are assigned to a new call and begin a new trip.

2. Trip originates from a previous call location: Here, the trip composition will have the responder dispatched after the completion of the previous call. The trip composition also has one of three possibilities mentioned above.

3. Trip originates from hospital: Here, the trip composition will have the responder dispatched after completing a previous call at the hospital. Again, the rest of the trip will follow one of the three possibilities mentioned above.

As outlined, three leg types exist for emergency responder trips:

1. Origin leg: the first leg of a trip associated with a call

2. Waypoint leg: the leg of the trip connecting the call location to a hospital

3. Return leg: the leg of the trip returning the responder to a station. In the study conducted in Chapter 5, we ignore return legs to stations as they do not directly affect response time.

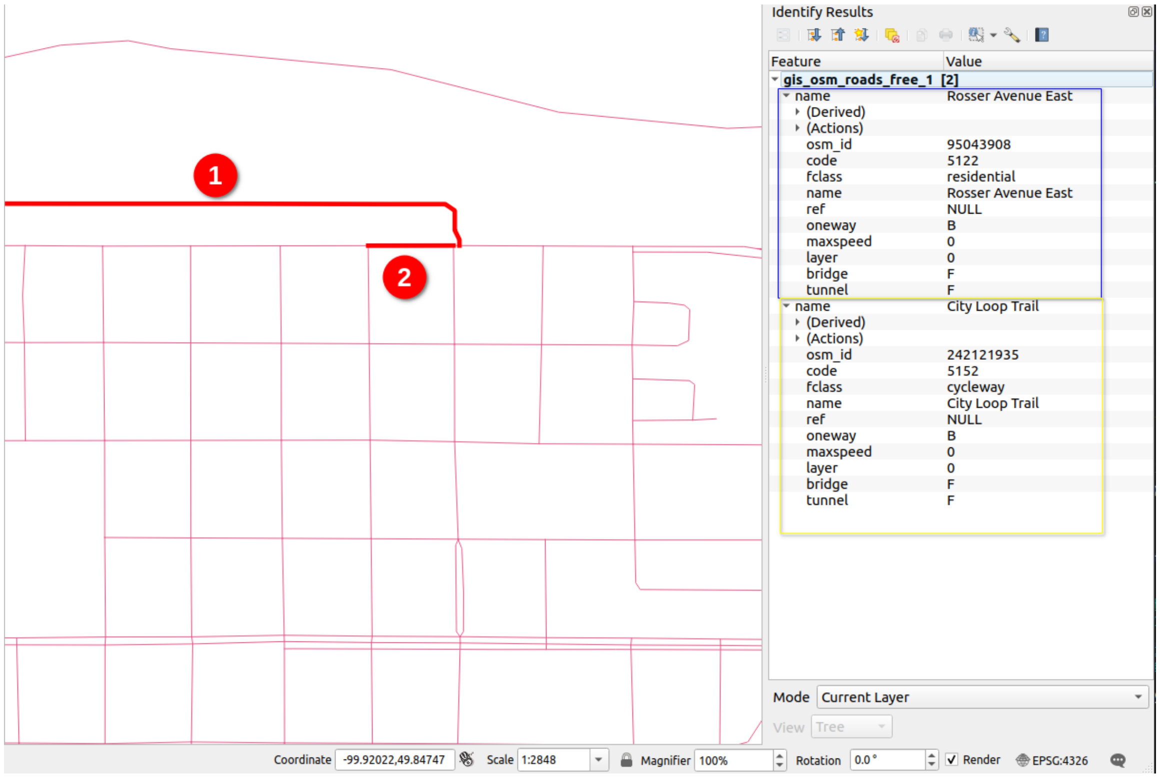
2.4 Shapefiles

Classically, spatial data can be considered as raster or vector data. There are three types of vector data: points, lines, and polygons. Note that a school could be represented as a polygon at larger scales, whereas it is likely represented as a point at smaller scales.

Shapefiles are referred to as "data layers". They are files that include geographical boundaries and spatial features in a format that software can use [RCPR02]. They store geometric locations and attribute information of geographic features. Shapefiles may include polygonal features. When representing vector data at larger scales, cities or postal code regions could be described using polygons. They could also have linear features such as streets, highways, or railway tracks. Lastly, shapefiles could include point features such as schools and hospitals when representing vector data at a small scale, intersections, or railway crossing [Env98; RCPR02].

A shapefile contains structured data; the data is in rows and columns, and one could store it in a relational database. A shapefile has a required column called "geometry"—where we store the geometric data as a Binary Large Object commonly referred to as a "blob". A shapefile can only store one type of geometric feature per file. Hence, if one were to represent a road network, they may need one shapefile to store all the roads as a linear feature. Then, another shapefile to store all intersections, bus stops, and other points of interest as point features.

Figure 2.7 is a screen capture of the data contained in a lines shapefile visualized using QGIS, a GIS software. This screen capture shows the layout of roads in a small town in Manitoba. We can see that the street labelled 1 and 2 are selected. Street 1 is a cycle trail with its properties shown in the yellow box on the right. Street 2 is a residential street and its properties are displayed in the blue box. By looking at both boxes one would notice the absence of the geometry property. This data is not a single text value but a number of points used to describe the curvature of a line. This geometry field is therefore unstructured. So, we classify shapefiles as semi-structured



Feature	Value
gis_osm_roads_free_1 [2]	
name	Rosser Avenue East
(Derived)	
(Actions)	
osm_id	95043908
code	5122
fclass	residential
name	Rosser Avenue East
ref	NULL
oneway	B
maxspeed	0
layer	0
bridge	F
tunnel	F
name	City Loop Trail
(Derived)	
(Actions)	
osm_id	242121935
code	5152
fclass	cycleway
name	City Loop Trail
ref	NULL
oneway	B
maxspeed	0
layer	0
bridge	F
tunnel	F

Mode Current Layer
View Tree

Coordinate -99.92022,49.84747 Scale 1:2848 Magnifier 100% Rotation 0.0 ° ✓ Render EPSG:4326

Figure 2.7: Visualized lines shapefile using QGIS

data.

2.5 Summary

In this chapter, we provided some relevant background information for the rest of this thesis. For example, we talked about relational databases, mentioning that MySQL was the most commonly used relational database. We also discussed two kinds of non-relational databases: document-oriented and graph databases. Graph databases are best used in storing well-connected data like transportation or social networks. We also discussed how the emergency response process works, using WFPS as an example. We showed how an active rail crossing could limit fleet capacity and increase response times. Finally, we discussed shapefiles, highlighting their importance in transportation studies.

Chapter 3

Related Works

Typically, the fundamental decisions involved in the design and development of big data management systems are choosing appropriate infrastructures for storage and computation [KLAA20]. As a result, this field has been subject to several contributions.

With massive volumes of transportation data collected at high velocity from different sources, integrating these valuable data sources for transportation analysis has become challenging. Because of their variety, many studies sought to find systems that accommodate heterogeneous data of various levels of structuredness [Bje18]. The term "structuredness" describes the ability of data to follow a particular style of representation (or schema) [ADF$^+$14]. As reported, Big data can be in structured, semi-structured or unstructured form [FPS96; ZCW$^+$16; CS17; DCL18; CHB$^+$20; KLAA20; VML20].

Most of the world's previously unknown information is in unstructured form. Even though Relational Database Management Systems (RDBMS) remain a leading database solution [DE20], their use as a “one size fits all” solution has become antiquated [SÇ18]. Additionally, RDBMS are difficult to expand once created; and requires a lot of computing power to return results in a reasonable time [SMA^{+}18].

Since the introduction of relational databases, there has been the introduction of model solutions that can house and process large amounts of data, regardless of their structuredness. The most prominent of them is NoSQL databases [ACP17]. NoSQL databases prove to be a cheaper solution for large volumes of data, as they can scale horizontally by adding more servers [SSPM16]. One cannot say the same about relational databases, as they need physical hardware upgrades to scale. There are many applications of non-related databases as DBMS solutions; most of them are used in complex domains, as shown by some studies [ACP17; AC18; DCL18; MNL20].

One cannot ignore the strengths of using relational databases for structured data. They fully support the requirements for ACID properties:

- Atomicity: The entire transaction is executed completely or it does not happen at all.
- Consistency: The database must remain consistent before and after a transaction takes place.
- Isolation: Multiple interactions occur independently of each other.
- Durability: The data is permanent in the database once the execution is completed.

In contrast, not all NoSQL databases fully support the requirements for ACID. Not to mention, access to information is fast, and relational databases provide efficient storage of data [RS20]. Some [GOR+15] have proposed completely migrating to NoSQL DBMS. In contrast, others [VCGF16; JMM+18] proposed techniques to integrate structured and unstructured data for data analysis.

Jankovic et al. [JMM+18], whose contribution was in the transportation domain, proposed a schema-on-read modelling approach and data virtualization. The contribution of this study was to display heterogeneous data sources as one integrated data source. Here, the data producer provides data in its raw form, and it will be up to the data consumer to apply their schema. Therefore, this approach works best when exploring data lakes or dealing with data from multiple sources [Del20]. While their approach showed outstanding results, the steps taken to integrate these data sources and leverage them for querying seem cumbersome and unpractical as a general solution. They do not represent the data graphically, which can be crucial for knowledge discovery analyses [ACP17].

Bessa [Bes22] proposed methods to get knowledge from transportation data by analyzing urban mobility patterns. Specifically, they analyze how traffic counts, meteorological conditions and public transportation demand affect the movement of people within a city. The approach collects and stores both structured and unstructured data into a NoSQL database. Our concern with this approach is that they use EasticSearch which has a steep learning curve to use. Our approach uses popular database solutions like MySQL and Neptune for managing structured, semi-structured and unstructured data.

Villari et al. [VCGF16] proposed a data-intensive approach. They employed Model-Driven Engineering (MDE) methods to generate data models from unstructured data. Notably, it forces the unstructured data to behave in a structured manner. The methods also leverage background knowledge in SQL-like databases when querying the data. However, their research only focused on the conceptual modelling of the data. The most prominent challenge with this school of thought is the lack of an explicit model to describe (big) data requirements, as seen in relational databases.

Villalobos et al. [VAOA22] proposed a hybrid of a relational and document database for evaluating the response times of a geoservice. Their work focused on the faster retrieval of geospatial data by combining the strengths of both database types. Their approach strengthens our case for a hybrid approach when storing and managing transportation, hence spatial data. Still, we think a graph database in their hybrid approach would be better suited to manage and retrieve spatial data.

Bjeladinović et al. [Bje18] proposed another line of thought. They insisted that relational and non-relational databases be used simultaneously instead of integrating them. Similarly, more studies [ZCW+16; CS17; ZBP+18; CHB+20; KLAA20; SGB20; VML20] have proposed this novel approach of utilizing *hybrid* databases comprising both relational and non-relational characteristics. They claim hybrid databases are a sound solution for handling big data while the ongoing transition from relational to non-relational database solutions. Even though some [KP17] believe that both database types are rather complements than substitutes to each other, only few [ZBP+18; SGB20; VML20] studies demonstrated this transition with real case solutions.

Zečević et al's [ZBP+18] approach was to extract data structures from data streams and then use these structures as a formal specification for a target model.

An issue with this approach (as with similar MDE approaches) was that they used one model in each stage of the design process. As we stated with Bessa [Bes22], the data sources make it difficult to easily adopt this approach.

We think it is important to mention that several studies [CR17; PPP+19; PPK+19] have shown how computationally demanding it is when a system performs data analysis from multiple data sources. While the framework proposed by Cui et al. [CHB+20] is very sound for transportation data analysis, it only considers road transportation using data from one source. Since Cui et al.'s solution [CHB+20] only uses one data source, it is fair to assume that it may not be thorough enough to be used as a standard.

Sokolova et al.'s [SGB20] proposal stand out as the most promising approach to a hybrid database solution for Big Data analysis. They use topology to migrate from SQL databases to NoSQL databases while maintaining the original SQL databases. This migration offers different modes of storing the same information and taking advantage of the strengths of both types of databases. Not to mention the effort it takes to perform such a transformation, Sokolova et al.'s [SGB20] approach reproduces the same data in different forms, hence inefficient from a data storage perspective. The data replication also introduces an additional concern, consistency, which the database administrator must ensure between databases. It is also important to note that their contribution is to the management of retail business data, whereas our contribution is to the management of transportation data.

Most of the data collected in the transportation domain are spatial. Mobile devices like smartphones, remote sensors and Internet-of-Things (IoT) produce location-based data, which has become increasingly available because of technological advancement [Kan19]. We refer to them as *geospatial data*. Geospatial data contains specific geographic information, such as longitude and latitude, representing a point on the map. They can also contain a series of longitude and latitude points, showing lines representing roads and areas (to represent a region) [WE18]. Geospatial data can also contain temporal information [EHG$^+$21]; that is, the data holds information about a location at a specific time.

Some immediate challenges are the storage solutions to use in housing and querying temporal data [Lak18]. In their study, relational database systems cannot satisfy the requirements for Spatio-temporal analysis [EHG$^+$21]. One primary requirement is visualization—their lack of a visual-based interface makes relational databases unfavourable for users when performing geospatial analysis [Kan19; EHG$^+$21]. In like manner, the choice of data storage can affect the results from the geospatial analysis [WE18]. For example, applying traditional data mining techniques to geospatial data could yield biased results when performing spatial auto-correlation studies. The biased results could be because the data needs to be stored in a way that shows relationships between nearby objects [WE18]. Having such data stored in a graph data structure can make performing techniques like running Spatio-temporal clustering algorithms on the data feasible.

Table 3.1 shows a summarized table of the related work. We highlight the preferred component (relational database vs. non-relational database) for each criterion in green.

Table 3.1: Comparison of relational and non-relational databases

Criteria	**Relational databases**	**Non-relational databases**
Handling structured data	excels at handling structured data [SÇ18]	inefficient at handling structured data [SGB20]
Consistency (all database transactions must change affected data according to all defined rules)	strict schema enforce consistency across the database [KL18]	low consistency due to a lack of schema [ACP17]
Handling wide variety (i.e., different categories) of data	cannot handle unstructured data [SÇ18]	can handle structured data and excels at handling unstructured data [Del20]
Handling large volume of data	handles limited amounts of data	excels at handling large volumes of data [ACP17]
Scalability (in response to increasing demand size)	scales up and down by modifying hardware [SSPM16]	scales horizontally by the use of commodity servers [SSPM16]
Flexibility (how well they perform outside their environments)	schema manipulation is a significant change once we deploy the database [SMA$^+$18]	due to an easily mutable schema, it is more flexible [ACP17]
Handling geo-spatial data	slower response times when performing queries on large amounts of spatial data [WE18]	excels at handling queries on large amounts of spatial data [WE18]
Handling spatio-temporal data	will yield biased results when performing spatio-temporal analysis [Kan19; WE18; EHG$^+$21]	strongly equipped to perform spatio-temporal analysis [WE18]
Support for performing machine learning tasks	more effort in schema design needed to support predictive outcomes algorithms [ACP17; WE18]	graph databases are more equipped for supporting predictive outcomes algorithms [WE18]

3.1 Summary

In this chapter, we outlined some existing work on relational, non-relational, and a mix of both databases. The consensus among the research seems to be that just one kind of database cannot support the modern-day application because of the variety of data. Many existing works directed our approach to implement a hybrid database. In particular, Sokolova et al.'s work inspired us to maintain the structured data in SQL databases while extracting unstructured features into a NoSQL database. We saw that there is a large body of work on database design. However, most of the literature does not show *practical*, real-world use cases of the hybrid approach on significant volumes of transportation data. In contrast, our research shows how we can use a hybrid database to support real-world transportation studies. We did not find any existing work directly comparable to our own. Our solution combines relational and non-relational databases to support a real-world data set.

Chapter 4

Hybrid Database Management System for Transportation

In this chapter, we answer the following questions: Q1. How well can the hybrid DBMS handle traditional structured data? Q2. How well can the hybrid DBMS handle semi-structured and unstructured data?

4.1 Overview

We provide scenarios using a single-style database approach to support emergency transportation analysis. We then compare them to a hybrid database approach. We conclude that a hybrid-style database approach is the better way to go. We provide an overview of the several stages we took to arrive at this conclusion as follows:

1. We identify essential data features/attributes that one needs to capture for a generic emergency responder transportation study.

2. We model the data attributes using a single style database approach. We take turns modelling the data sets captured in a relational-database-only system and then a non-relational (graph-database-only) system.

3. Observing that both approaches thrive in specific scenarios and do poorly in others, we explore a composite (hybrid) database style comprising both databases, which appears to be the best of both worlds.

4.2 Essential Features Needed to Perform Generic Transportation Analysis

This study intends to build a model to support analyzing railway and road transportation. On the most basic level, if we intend to analyze any form of transportation, we need to build that transportation network model to proceed. Hence, we will build railway and road transportation networks, which will be the skeleton of our analysis process. The building blocks of the model for our analysis will be as follows.

4.2.1 Transportation Network

Road and rail have geospatial features. Hence, to build a network for these components, we need to ingest geospatial data, hence, **shapefiles**. It is important to note that, although shapefiles are available as open data in many cities, the specific features and attributes vary between cities. Hence, to make our approach as easily adaptable as possible, we decided to use shapefiles from OpenStreetMap (OSM) because they are the most common providers worldwide. Note that, unfortunately, many jurisdictions still maintain and use their own spatial datasets, with their own unique fields and structures. Shapefiles combine both structured and unstructured data. Other than the geometry column, they strictly conform to a structure. Hence, we consider shapefiles **semi-structured data**.

In addition, all layers from the OSM data set have the following common attributes:

- id: the identifier for the feature, a unique identifier
- osm_id: an OSM identifier for the feature
- name: the name of the feature
- code: a four-digit code defining the feature class
- fclass: the class name of the feature
- geometry: the geospatial record of this feature (e.g., POINT, POLYGON, LINESTRING)

Let us describe the shapefiles for road and rail.

- Roads shapefile holds linear features of all kinds of roads, from streets and avenues to gravel tracks and lanes. For example, depending on the size of the location, the data set can contain between 200,000 to 500,000 rows of data. In addition to the attributes mentioned earlier, this shapefile also contains the following attributes:
  - ref: A reference number for the road segment, expressed as a character string.

  - oneway: A single-character string containing ‘T’ or ‘F’ or ‘B’. Which answers the question: “Is this a one-way road?” The ‘B’ indicates that it is bi-directional. When it is ‘F’ (false), only driving in the direction of the line-string is allowed.
  - maxspeed: The maximum allowed speed in km/h, expressed as an integer.
  - layer: The relative layering of roads, expressed as an integer.
  - bridge: A single-character string containing ‘T’ or ‘F’. Which answers the question: “Is this road on a bridge?”.
  - tunnel: A single-character string containing ‘T’ or ‘F’. Which answers the question: “Is this road in a tunnel?”.

- Railways shapefile holds linear features of the railway tracks. For example, depending on the size of the location, the data set can contain between 3,000 to 5,000 rows of data.

  In addition to the attributes mentioned earlier, this shapefile also contains the following attributes:

  - layer: The relative layering of railways, expressed as an integer.
  - bridge: A single-character string containing ‘T’ or ‘F’. Which answers the question: “Is this rail track on a bridge?”.
  - tunnel: A single-character string containing ‘T’ or ‘F’. Which answers the question: “Is this rail track in a tunnel?”.

- Traffic shapefile holds traffic-related points information from stops and crossings to traffic signals and speed cameras. The feature class of interest in this data set is “crossings”. Although OSM does not differentiate between railway or pedestrian crossings, we will have to filter out pedestrian crossings as it is not a feature of interest for our analysis. Again, for example, depending on the size of the location, the data set can contain between 2,000 to 50,000 rows of data.

4.2.2 Supplemental Data Sets

Shapefiles on their own may not be enough source of information upon which to build our analysis. More often than none, supplemental data sets rich in information may be required to complement the data in the shapefiles. For example, say one intends to group their analysis by postal code or other regions of interest, one would need to ingest those data sources. We can include other grade crossing information as a structured **CSV file** for this case. It is standard practice worldwide for transportation authorities to generate and maintain statistics on transportation infrastructure. We have used this database approach to perform analysis in the United States and Canada. Our findings reveal that the US Federal Railway Authority (FRA) and Transport Canada provide important statistical information about grade crossings in the United States and Canada, respectively.

We will discuss some essential attributes in the data set with similar data points. *Grade Crossing Inventory* is a CSV file containing all the railway grade crossings in Canada provided for free by Transport Canada. In addition to the name, latitude and longitude, the inventory also contains statistical information. Core information like the number of accidents, fatalities, and trains per day will be important for analysis.

For example, there are 26 attributes contained in the Transport Canada data set

Table 4.1: Data sources, their formats, and structure

Data	**Owner**	**Format**	**Structure**
Bidirectional road network	OSM	Shapefile	semi-structured
Train tracks inventory	OSM	Shapefile	semi-structured
Traffic related points	OSM	Shapefile	semi-structured
Grade crossing inventory	Transport Canada/ FRA	CSV	structured

and 255 in FRA's. In order to conserve space in this document, we will include just four important attributes. These four attributes are:

- Protection: Type of warning at the crossing, expressed as a character string.
- Vehicles Daily: An estimate of the number of road vehicles per day over the crossing, expressed as an integer.
- Total Trains Daily: An estimate of the number of freight (and passenger) train movements per day over the crossing, expressed as an integer.
- Train Max Speed (mph): An estimate of the maximum train speed over the grade crossing in miles per hour, expressed as an integer.

For the most generic transportation analysis, Table 4.1 is a list of our data set, the data owners, their format and structure.

4.3 Using Only a Relational Database Model

After analyzing the data requirements, the next step is to attempt to design a model to handle these data sets using only a relational database. To perform an analysis, let us design a high-level description of the data to be stored and derived.

4.3.1 Road Segment

It describes a single linear road object. Based on the attributes mentioned in Section 4.2, it would be logical to have the following conceptual schema for the road segment.

Listing 4.1: Road Segment schema

```
CREATE TABLE `roads` (
  `id` varchar(10) NOT NULL,
  `osm_id` varchar(10) NOT NULL,
  `code` int DEFAULT NULL,
  `fclass` varchar(40) DEFAULT NULL,
  `name` varchar(100) DEFAULT NULL,
  `ref` varchar(20) DEFAULT NULL,
  `oneway` varchar(1) DEFAULT NULL,
  `maxspeed` int DEFAULT NULL,
  `layer` int DEFAULT NULL,
  `bridge` varchar(1) DEFAULT NULL,
  `tunnel` varchar(1) DEFAULT NULL,
  `geometry` linestring NOT NULL,
 PRIMARY KEY (`id`)
);
```

4.3.2 Railway

It describes a single rail track. Its conceptual schema would be:

Listing 4.2: Rail Track schema

```
CREATE TABLE `rail_tracks` (
   `id` varchar(10) NOT NULL,
   `osm_id` varchar(10) NOT NULL,
   `code` int DEFAULT NULL,
   `fclass` varchar(40) DEFAULT NULL,
   `name` varchar(100) DEFAULT NULL,
   `layer` int DEFAULT NULL,
   `bridge` varchar(1) DEFAULT NULL,
   `tunnel` varchar(1) DEFAULT NULL,
   `geometry` linestring NOT NULL,
  PRIMARY KEY (`id`)
);
```

4.3.3 Crossing

It describes an intersection between a road segment and a rail track. Its conceptual schema would be:

Listing 4.3: Crossing schema

```
CREATE TABLE `crossings` (
   `id` varchar(10) NOT NULL,
   `osm_id` varchar(10) NOT NULL,
   `code` int DEFAULT NULL,
   `fclass` varchar(40) DEFAULT NULL,
   `name` varchar(100) DEFAULT NULL,
   `protection` varchar(40) DEFAULT NULL,
   `num_vehicles_daily` int DEFAULT NULL,
   `num_trains_daily` int DEFAULT NULL,
   `train_max_spd` int DEFAULT NULL,
   `geometry` point NOT NULL,
  PRIMARY KEY (`id`)
);
```

4.3.4 Relationships among Railway, Crossings, and Roads

As defined earlier, crossings are an intersection between rail and road segments. Thus, we will have to design that relationship in a table. We can look up the relationship between these three components by the (rail track ID, road ID, crossing ID,)-tuple. These describe where these three data sources align in the real world. Hence the conceptual schema will be as follows:

Listing 4.4: Railway × Crossing × Roads Relationship schema

```
CREATE TABLE `roads_railway_xing_interaction` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `cd_crossing` varchar(10) NOT NULL,
  `cd_road` varchar(10) NOT NULL,
  `cd_rail_track` varchar(10) NOT NULL,
  `start_time` timestamp NOT NULL,
  `end_time` timestamp NOT NULL,
 PRIMARY KEY (`id`),
 KEY `ri_fk_cr` (`cd_crossing`),
 KEY `ri_fk_rd` (`cd_road`),
 KEY `ri_fk_rt` (`cd_rail_track`),
 CONSTRAINT `ri_fk_cr` FOREIGN KEY (`cd_crossing`)
 REFERENCES `crossings` (`id`)
 ON DELETE NO ACTION ON UPDATE NO ACTION,
 CONSTRAINT `ri_fk_rd` FOREIGN KEY (`cd_road`)
 REFERENCES `roads` (`id`)
 ON DELETE NO ACTION ON UPDATE NO ACTION,
 CONSTRAINT `ri_fk_rt` FOREIGN KEY (`cd_rail_track`)
 REFERENCES `rail_tracks` (`id`)
 ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

4.3.5 Potential Issues with Using Just the Relational Model

Having seen the data and subsequently generated tables, let us explore some potential difficulties when attempting to perform analysis using this database design alone:

- Relationships between roads: In this relational model, we have extracted the contents of the shapefile and put them into a table. The caveat of having this data stored in a table form is that we cannot infer certain spatial information with the way we stored the data. For example, without performing several queries, one cannot look at a row in the road segment table and infer the road segments that are spatially next to it.

- Navigation: Similarly, if one intends to find a route from one road segment to another on the road network, it would prove cumbersome. We first must build a relationship between each road segment and all other road segments. This process is computationally expensive, hence inefficient, not to mention impractical.
- Multiple knowledge bases: With the relational database model, we must join several tables to pull in all those properties to see the characteristics of a feature, say, a road segment. For example, if we needed to determine the number of casualties/accidents on a road segment. We would need to create a join with the (crossings-roads-rail) table and then use that to look at the grade crossing inventory table where that record lies.
- Some aggregation will be cumbersome: Imagine the effort it would take to aggregate the number of trips interacting with each crossing. We will need to generate another table. Essentially, we will need to generate another table or view to analyze each new metric of interest.
- Ease of adoption: At the very generic level, the number of tables is already growing. Conveying these multiple tables might be a steep learning curve.

Notwithstanding, the relational database has a decisive advantage in data visualization. Many widely used industry-standard data visualization tools work best with relational data. They make aggregation and information conveyance very easy. Among the notable solutions are PowerBi, AWS Quicksight, Tableau and IBM Cognos. Moreover, although standard geographic information system (GIS) software may handle all these issues (e.g., buffer analysis, building of a "connected" network to facilitate routing and other spatial analyses), it can be expensive. Furthermore, we are aiming for a more "first principles" solution.

4.4 Using Only a Graph Database model

We mentioned in Section 4.3 that we could present everything in a relational model. However, it did not seem like a practical approach for analysis and management. Since we are dealing with highly connected data, storing it in a graph database may be more effective.

As the basis on which we will be conducting our analysis, the first step would be to extract the content and features of the shapefiles. The elements in the shapefile already store connected information. We will then add other data sources that complement the attributes of the shapefiles. These will become additional properties of the edges and nodes in the graph.

If we revisit Table 4.1, we have three shapefiles from the same source. The benefit of building the framework of the graph from shapefiles of the same source is that the data points will match across the files as they do geographically, since they are from the same source.

The purpose of this database will be to support studies that analyze the interaction

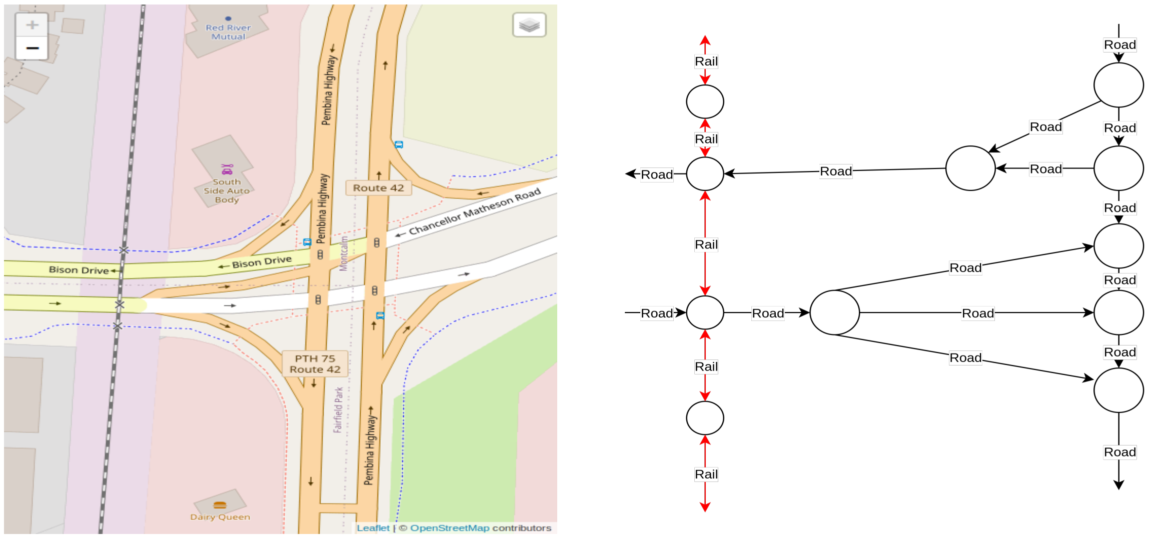


Figure 4.1: Fused graph network of rail, road, and traffic shapefiles

between road and railway transportation. Our graph will also need to be composed of a network of road and railway edges and nodes; that is, nodes and edges can be road or rail components. To build such a graph, we will first need to extract the road and rail features and build them as individual layers. Then, in another step, we will join both layers into a fused network of nodes and edges. Remember, the traffic-related points shapefile, among other things, holds records of grade crossings, that is, locations where the rail lines interact with the road network. Those points will be the fusing locations to merge both layers into a single graph layer. As seen in Figure 4.1, the image on the left shows an intersection of a rail line across Bison drive. The graphical representation on the right shows the composite graph of the fused rail network layer with the road network layer once we have merged both layers into a single layer. Note that the graphical representation only captures the south-bound Pembina Highway road segment.

We find it pertinent to mention that a standard GIS software like QGIS is capable of retrieving information from shapefiles, as we propose here for the graph model. Also, GIS facilitates routing from one point to another. However, the real value of using a graph model is integrating multiple data sets. Observed from Table 4.1, we have a mix of geospatial and non-geospatial data sources. One must manipulate the underlying program to merge these data sets to use in a GIS environment. Even so, with a graph, we can run various graph algorithms, for example, Monte Carlo or Dijkstra's Shortest Path—these are areas where standard GIS software falls short. Hence, using a graph model appears more desirable due to this utility.

4.4.1 Forming of the Individual Graphs

As mentioned earlier, we will create two separate graphs; one for roads and the other for the railway. We will then fuse both graphs at their geographical meeting points. Table 4.2 shows the node and edge classes for our graphs.

The first step is to create the individual graphs from their shapefiles before joining both layers. Algorithm 1 converts a lines shapefile (rail or road) to a simple graph.

4.4.2 Joining the Layers of the Graph

The goal of this step of the design is to generate a composite graph of edges and nodes from the road and rail network. The composite graph will be generated from two graph meta sources formed in Section 4.4.1. With the input of both graphs, we will then go ahead to fuse them together at joining points as seen in Algorithm 2 as follows:

After fusing the road and rail network graphs, we proceed to the last stage of

Algorithm 1 Convert lines shapefile to graph

```
Input: $shapefile$
Output: $edges$, $nodes$
 1: $nodes : Dict[str, Point] \leftarrow \{\}$
 2: $edges : Dict[str, List[LineSegment]] \leftarrow \{\}$
 3: createOneEdgeAndTwoNodesFromLines($shapefile.rows, nodes, edges$)
 4: for $line$ : $edges$ do
 5:     $intersectingLines \leftarrow$ getAllLinesThatIntersectsWith($line$)
 6:     for $intersectingLine$ : $intersectingLines$ do
 7:         $point \leftarrow$ findIntersectionPoint($line, intersectingLine$)
 8:         if $point = NULL$ then
 9:             continue
10:         if isPointAtEitherEndOfLine(intersectingLine, point) then
11:             continue                ▷ No need to split intersectingLine into two at point
          ▷ This line needs to be split into two segments at $point$. $point$ becomes a node in the graph
12:         createNewNodeAndAddToNodes(point, nodes)
13:         $subSegment_1, subSegment_2 \leftarrow$ splitLineAtPoint($intersectingLine, point$)
14:         addEdgeToEdges($subSegment_1$)
15:         addEdgeToEdges($subSegment_2$)
16:         removeEdgeFromEdges($intersectingLine$)
```

Algorithm 2 Fuse road and rail graphs

```
Input: $edgesRoad$, $edgesRail$, $nodesRoad$, $nodesRail$
Output: $compositeEdges$, $compositeNodes$
 1: $compositeNodes \leftarrow nodesRoad$
 2: $compositeEdges \leftarrow edgesRoad$
 3: $compositeEdges.addAll(edgesRail)$
      ▷ Insert nodesRail into compositeNodes. Update edge references IF the node already exists
 4: for $nodeRail$ : $nodesRail$ do
 5:     if $nodeRail.coordinates \notin nodes$ then
 6:         addToNodes($nodeRail$)
 7:     else                                        ▷ We have found a grade crossing
 8:         $commonNode \leftarrow nodes[nodeRail.coordinates]$
 9:         $validCrossing \leftarrow$ doesNodeAppearInTrafficDataset($commonNode.coordinates$)
10:         if $validCrossing \neq TRUE$ then
11:             $continue$                          ▷ Manually investigate odd occurrence.
12:         $railEdges \leftarrow$ findEdgesWithNodeCoordinates($nodeRail.coordinates$)     ▷ Update
    rail edges to now reference commonNode instead of nodeRail.
13:         for $railEdge$ : $railEdges$ do
14:             if $railEdge.startNode.coordinates = nodeRail.coordinates$ then
15:                 $railEdge.startNode \leftarrow commonNode$
16:             else
17:                 $railEdge.endNode \leftarrow commonNode$
18:         copyOverNodeProperties($commonNode, nodeRail$)
19:         $commonNode.type \leftarrow NodeType.RAILCROSSING$
```

Table 4.2: Node and Edge Classes for the Graph

Component	**Node or Edge**	**Description**
Road segment	Edge	A road or way that a vehicle travels on
Rail segment	Edge	A train track
Road Intersection	Node	A point where two road segments overlap
Road Junction	Node	A point where two road segments touch, but do not overlap
Rail Crossing	Node	A point where a road segment crosses a rail segment
Rail Junction	Node	A point where two rail segments touch

building the property graph. This stage involves adding supplemental data containing crucial statistical information from the grade crossing inventory file. Our concern was that, as both data sets were from different sources, they did not align as a 1:1 match in several spots. We needed to find a way of guaranteeing that we matched one row of data in the additional data set with the correct, corresponding node generated from the OSM shapefiles.

We evaluated some approaches, including searching for grade crossings by street names and subdivisions. The best approach we found was to use the coordinates from both data sets to try and find their match. With the coordinates approach, the challenge was to match crossings with their corresponding values in the other data set. With some crossings close to each other, we explored using the precision of decimal points in both sets to see if we could find a match. Our resulting matches included some false positive matches. Ultimately, we achieved the best results by making a simple circle around the coordinates of both data sets with a diameter of eight to ten metres. Then, we considered intersecting circles from both data sets as matches. We could confirm we were achieving over 95% accuracy in aligning both data sets than a mere 68% when using coordinates precision matching. Algorithm 3 shows this process.

Algorithm 3 Align supplemental grade crossings data source

```
Input: $nodesRail$, $gradeCrossingCSV$
1: for $gradeCrossing$ : $gradeCrossingCSV$ do
2:     $crossingBoundary \leftarrow$ GEN10METREGEOFENCE($gradeCrossing.coordinates$)
3:     $node \leftarrow$ FINDNODETHATINTERSECTS($crossingBoundary$)
4:     if $node = NULL$ then
5:         continue
6:     COPYOVERATTRIBUTES($gradeCrossing, node$)
```

Having the road and rail network graphs fused, we can start analysis to understand the relationships between the activities on a node in the graph and how it impacts the rest of the network. For instance, we can now better understand how an active crossing node impacts road traffic flow.

4.4.3 Potential Issues with Using Just the Graph Model

Nevertheless, the graph database still has its shortcomings. Let us explore some potential difficulties when we attempt to use this database type for our analysis:

- Visualization: There is currently very little support for visualizing big data on graph databases, especially geospatial data. As a result, conveying results of analysis to end users would be difficult to do as some data are best explained visually.

- Building the graph: We had mentioned in Section 4.4 that we need to store the composite graph's meta sources somewhere. We store the meta sources to reproduce the graph in the future. We also keep the meta sources if we need to isolate a graph layer and perform an analysis only on one layer. Building these individual layers, then combining them, may be cumbersome.

- Query time: When we run analysis on the graph, the question that comes to mind is how we store the results from that analysis. With the graph database only approach, we would have to store the results from graph operations directly on the graph. A typical operation includes pathfinding routes between two nodes. Over time, storing those results on the graph would increase the size of the nodes and edges and may cause searches to become slower.

4.5 Observations

By looking at the database information without actually considering the data involved in the analysis, the relational database seems to have some advantages. The relational database has an index which makes it more efficient to search through some related data, something we cannot achieve with the graph database. The relational database also has mass support of visualization tools, making it an easy choice for relaying results from analysis to users.

Since relational databases are best at handling transactional records, they will be the preferred storage choice to hold and process transactional transportation data. These data include travel logs, commuter schedules, and terminal information.

Storing abstract data types such as lines, polygons, and bit maps allow relational databases to store and retrieve spatial information efficiently.

The structure of the transportation network is inherent to a graph structure. Geographical information in shapefiles makes a strong case for storing it on the graph as the data is highly connected. Whereas, in just the tabular model, the network's complexities have been represented using multiple relationship tables and joins. The graph structure would eliminate this overhead if we were to store the data in the structure of the road and rail networks.

We also discussed the challenges of solely using a graph database approach. One of those challenges was slower query times when analysis data are stored directly on the graph. We will explore the possibility of performing analyses on the geospatial data on the graph database, then storing the results and statistical summarizations on the relational database.

One thing is inevitable. We must retrieve the data from shapefiles and store them in tables or a graph database. With tables, we can hold transactional records that run on the graph or hold results retrieved from the graph. Whereas with a graph, we maintain the same class structure, the extraction becomes cleaner, and the data features are all in one place.

Ultimately, we need to use a relational database for some data types, while we need the graph database for others. As a result, there is a trade-off between the relational model and the graph database model—we do not have a clear winner. In Chapter 6, we will assess the desired features of a database solution to support our kind of analysis to determine if a single database approach satisfies all criteria.

4.6 Using a Hybrid Model

We present a hybrid model design comprising a relational database and a property graph database. A two-component hybrid model will help us take full advantage of the strengths of each component database. We discuss the roles of the individual components as follows:

1. Graph database: We will use a property graph for this component. For all the reasons mentioned in Section 2.2.2, we can have all the key/essential data attributes in one place, at the node or edge. Since the structure works best for connected data, we use this database to perform most of the analysis. The graph aids us in discovering connections and patterns between nodes and edges. It will also help us answer questions we did not know to ask at the beginning of the database design. Hence, the graph database is essential to the hybrid database design. Then, the relational and document databases exist to support the capabilities of the graph. Since the graph database is a property graph, the contents of the shapefiles and crossing inventory listed in Table 4.1, once extracted by Algorithms 1, 2 and 3, will be stored as key-value attributes in the graph.

2. Relational database: This will be the second of the two components of the Data Warehouse. Table 4.1 shows us that the initial data set we will need for building our transportation network structure is primarily structured. It is crucial to take advantage of the relational database's quick look-up functionality, especially when building the graphs in Algorithms 1 and 2. With the contents of

the shapefiles loaded into a relational database, we can save time when building the graph. Without calling the `ST_* (Intersects, Touches)` functions for MySQL (or similar), finding these corresponding geometries would be cumbersome. Those built-in functions take constant time to yield results, whereas the alternative would be to loop through the data.

Additionally, the relational database can hold summarization statistics about our analyses, that is, the output of the analyses that run on the graph. We had mentioned in Section 4.5 that a strong reason for using relational databases over graphs is their strong support for visualization. Holding summary data from analyses in the relational database makes using readily available visualization tools seamless. In the future, when graph databases develop strong support for visualization, this may not be a required feature.

More so, to improve the efficiency of the graph database, we can store graph traversals between two nodes in the relational database. That way, instead of repeating a traversal, we easily retrieve the path from a table.

4.6.1 Database System Architecture

We had mentioned in Section 4.5 that the meta-sources that form the composite graph might need to be stored somewhere. Our reason for this decision is to give us the ability to quickly regenerate the graph and/or perform isolated analysis on layers if desired. Additionally, we need a place to store the shapefiles and CSV files before they are transformed and loaded into the relational database. It would be more efficient to have a document-oriented database to serve as a data lake to facilitate this particular hybrid model.

Following the database roles described, we will have the following resulting database architecture:

- Data Lake: We propose facilitating the hybrid model with a Data Lake. A NoSQL document database instance will store the data in their raw forms, available on demand. It will primarily hold the contents of Table 4.1 before they can be extracted (transformed) and loaded into more usable forms. Also, to extract and transform the data in the lake into the warehouse, we use Python scripts. They are represented by the boxes "Algorithm x" in Figure 4.2. We can also use the document-oriented database to store larger components used in characterizing edges and nodes of the graph database. For example, it may be crucial for an analyst to have grade crossing nodes of the graph reference all the unique train bells encountered at that location over time. They may also need road edges in the graph to reference historical images to show road degradation over time.

  These additional data characteristics can be easily stored directly in the graph. However, storing all this data on the graph may be inefficient as it may have adverse effects such as more network latency or increased traversal cost. Our opinion is to have an attribute of the edge or node reference an external location. The reference could be the collection's name in the document-oriented database, where the data can be easily accessed when needed. This strategy may help keep the load on the graph light.

- Data Warehouse: Our Data Warehouse contains the proposed hybrid model. It will house the data ready to be used by the analysis tool. The data warehouse will comprise a property graph for representing geospatial features. Our research uses the graph to model the road and railway network in the city/province/state. We will load data from the lake and transform that data into a format ready for use by the analysis tool. Additionally, the warehouse will include a relational database best used for storing tabular data. We will use it in storing CSV data as they are already structured. To summarize, these are the components of the warehouse:

– Property Graph: for modelling the rail and road network of the city

– Relational Database: for holding statistical data records in the form of fact and dimension tables, trip traversal records, and references of data located in the Data Lake.

Consequently, Figure 4.2 shows how these components interact with one another. We should mention that the analysis tool does not contribute to the hybrid database design. We will look at it as a "black box" that queries our databases for data. As a result, we will not discuss its contents and functionality in this thesis.

4.7 Discussion

Our hybrid design is a very generic design of a database architecture that can support an EMS transportation study. There are undoubtedly several concerns one might have. Some of those concerns include:

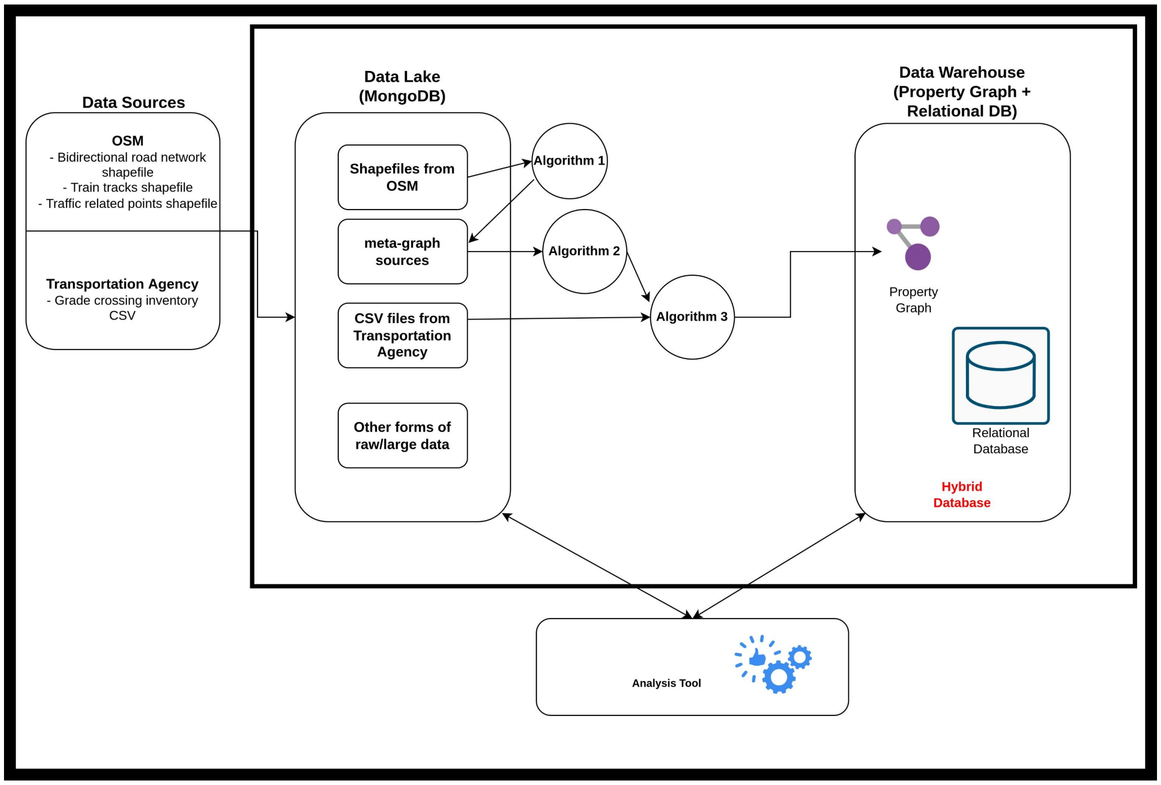

Data Sources
OSM
- Bidirectional road network shapefile
- Train tracks shapefile
- Traffic related points shapefile
Transportation Agency
- Grade crossing inventory CSV
Data Lake (MongoDB)
Shapefiles from OSM
meta-graph sources
CSV files from Transportation Agency
Other forms of raw/large data
Algorithm 1
Algorithm 2
Algorithm 3
Data Warehouse (Property Graph + Relational DB)
Property Graph
Relational Database
Hybrid Database
Analysis Tool

Figure 4.2: Architecture design for our hybrid model

1. *Is this hybrid approach indeed faster than the single-style approach?*

   We will assess this in Chapter 6.

2. *Are we introducing new problems in communication lags and connectivity between databases?*

   We will also assess this in Chapter 6.

3. *How about data consistency among databases?*

   Data consistency is not an issue as the information captured is disjoint between databases.

4.8 Summary

This chapter presented a hybrid database architecture for supporting EMS transportation analysis studies. We outlined the most crucial data attributes essential to any transportation study. In most cases, these data attributes are present in shapefiles. Additionally, we used the most commonly used shapefile source in the industry in OSM.

We analyzed the pros and cons of using a relational-only database to support this analysis and a graph-only database. The analysis led us to conclude that there is no clear winner between both databases. We then presented our hybrid database approach comprising a graph database as the primary database for running our analysis, with support from a relational database. In particular, the relational database supports the graph by helping us visualize results from our analyses and reducing the number of graph traversals. We also suggest facilitating our hybrid model with a document database for long-term data storage. We propose that this is the better approach as we enjoy each database's strengths without suffering the drawbacks of a single-style approach. The graph best models the nature of our transportation network; it allows us to look at data attributes and their interaction with one another in a very simplistic, intuitive manner. The hybrid approach helps us to relate transactional records to segmentation of the graph. We hold certain summary aspects that require more complexity in a different location when we use the hybrid approach instead of it being fused directly into the graph.

Finally, we also mentioned that there might be challenges with this approach. We mentioned the issue of latency and communication between databases. These concerns will need to be experimented upon in Chapter 6 to determine if those concerns are significant or worth the design decision.

Chapter 5

Case Studies: Using the Hybrid Database for EMS Response Times Study

In this chapter, we answer the following question: Q3. How well can the hybrid DBMS handle some real-life applications with structured, semi-structured and unstructured data?

5.1 Overview

We introduced a generic architectural design for supporting EMS transportation analysis in Chapter 4. Here, we go one step further to describe how we used this hybrid database approach to support two real-world studies on emergency response times. Using two real-world first responder transportation logs, we show that we can apply the approach to any first responder-specific transportation analysis.

These case studies are a research effort between the Federal Railroad Authority (FRA) in the United States and TRAINFO Corp. TRAINFO conducted an extensive study in Winnipeg, Manitoba, Canada, measuring first responder delays caused by trains in the year 2018. While writing this thesis, TRAINFO is also conducting another study on first responder trips in Charleston, South Carolina, USA, for 2021, with preliminary results only available now. Both studies aim to determine the number of first responder trips affected by the blockage of crossings. They also determine if responders could have saved time if they used another route during said trips.

It is crucial to note that this hybrid approach supported two studies on EMS response times from different cities in North America, highlighting its adaptability. There is no significant difference in the data set requirements from city to city. The data in Table 4.1 will still be required to build the database regardless of this city. Likewise, any data set on trips regardless of the city will have the following attributes in common: origin, destination, an identifier of the travelling unit (e.g. bus number), and optionally, a route travelled. By supporting both studies, we show that our hybrid design approach is not limited to any geography.

The result of this study highlights two key contributions of this hybrid database approach:

1. It can be used for real-world applications, and
2. It enhances simple data mining through information retrieval (from a historical archive), which can help advise non-real-time decision-makers.

We will discuss both case studies supported using the hybrid database approach individually. However, certain things remain the same in both studies.

In both studies, first responders provided TRAINFO with a one-year history of EMS and Fire trips completed. TRAINFO developed an EMS Risk Model to:

1. Quantify the frequency and duration of first responder delays at rail crossings in the city.
2. Identify which areas of the city were most affected by these delays.
3. Determine if real-time and predictive train information could reduce these delays.

The process involved:

1. Importing call log and Automatic vehicle locator (AVL) data (if provided) into the model. An AVL is a device that makes use of the Global Positioning System (GPS) to enable agencies to track the location of its vehicle fleet remotely by using the internet. AVL pings are the intermittent GPS locations for the vehicles as they travel.
2. Running the model to determine the first responder trips delayed by trains and
3. Simulating re-route options to calculate possible travel time savings.

Figure 5.1 shows a map of call origins and destinations, hospitals, and rail crossings. The model generated responder trip paths and identified units that were potentially delayed by trains when a trip satisfied three conditions:

1. the unit's route crossed railway tracks,

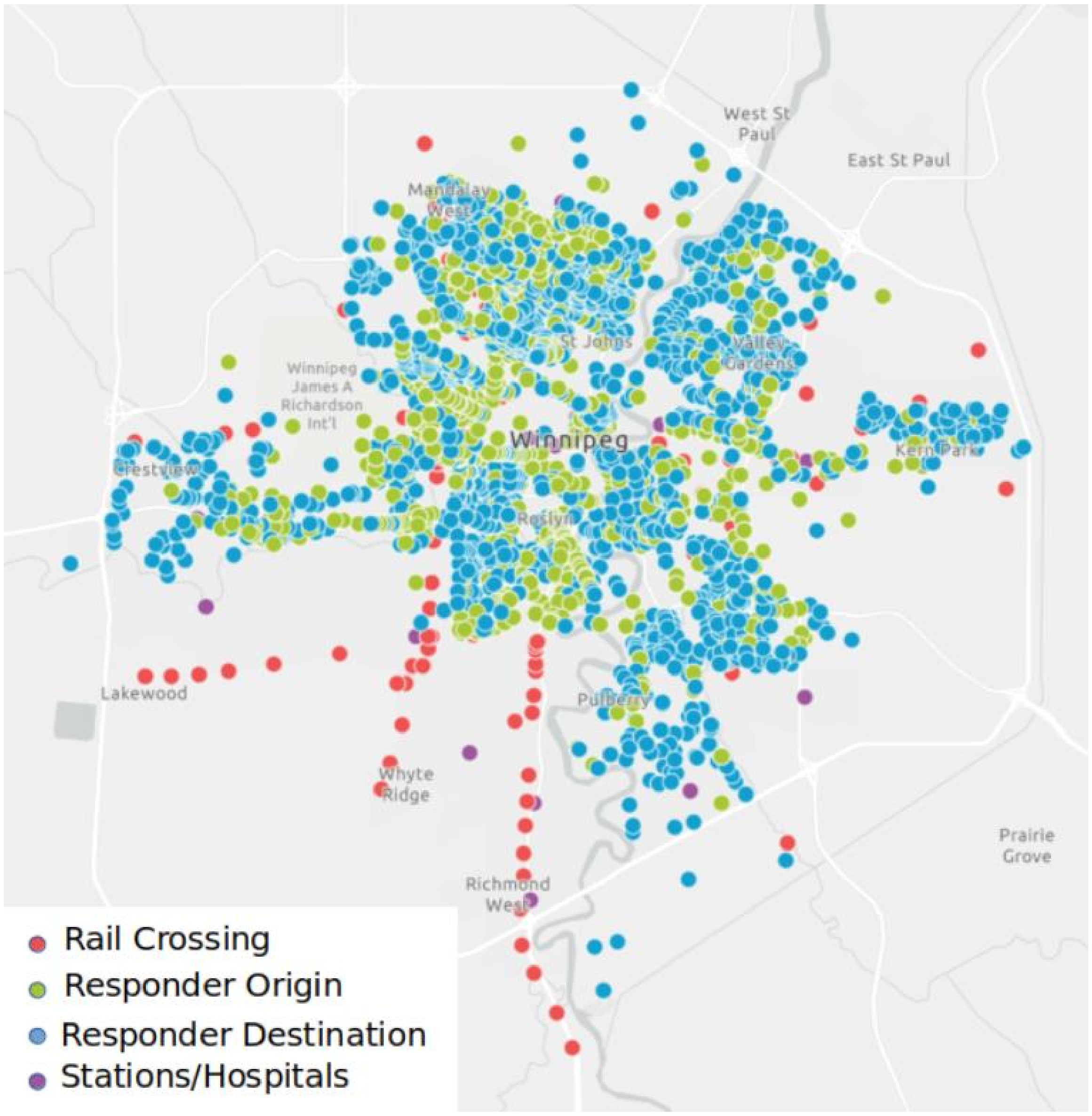


Figure 5.1: WFPS call origins and destinations

2. a train was blocking the crossing during the unit's travel time, and

3. the unit's actual response time was longer than the expected response time.

We estimated unit travel times based on the time of day, day of the week, speed limit, the number of lanes, and traffic volume characteristics. We called this estimate the expected travel times. The model compared the expected travel time to the actual travel time for units meeting the three conditions and calculated the amount of delay caused by the train. For delayed units, the model simulated travel times along alternative routes. Alternative routing was to determine if delayed trips could have saved time by re-routing and estimated how much time the unit could have saved.

The EMS Risk Model also produced two metrics: the Crossing Risk Score and the Area Risk Score. Crossing risk scores show the frequency of first response delays caused by blocked rail crossings. We stored this value as an attribute of the crossing node in the graph. The Area Risk Score reveals which areas of Winnipeg had the highest risk of experiencing first responder delays because of trains (measured as units delayed per month).

We should mention that TRAINFO decided on how to conduct the analysis. Thus, we will limit the scope to how we used the hybrid database to perform the analysis.

5.2 WFPS Case Study in Canada

Winnipeg is the capital city of the Province of Manitoba in Canada. It has a growing population of 800,000 and a land area of 180 square miles. There are two Class 1 railroads with mainline tracks running through the city, resulting in over 100 rail crossings and upwards of 50 trains moving through the city daily. Winnipeg Fire Paramedic Service (WFPS) provides fire and EMS services to the city, from 27 fire stations, three stand-alone ambulance stations, and six hospitals. WFPS responders complete around 3,000 trips per week, with approximately 15% of trips crossing railway tracks.

WFPS dispatchers and call-takers do not know if or when a rail crossing will be blocked when they select units and routes to respond to an emergency. When first responders encounter a blocked rail crossing, they radio dispatch and await instructions. Dispatchers do not know when the crossing will be clear and cannot judge if it is faster for the unit to re-route, wait for the train to clear, or dispatch a second unit. Depending on the situation, they use one or more of these options. Dispatchers automatically send multiple units for specific call locations just in case a train blocks the primary route. Dispatchers can contact the railroad and request they cut the train to allow first responders to go through. However, they rarely use this process because it takes longer than the other options.

WFPS explored various approaches to address first responder delays at rail crossings. One approach was grade separation (building over-or underpasses at rail crossings); however, this was cost-prohibitive and infeasible in dense parts of the city. A second approach involved WFPS requesting live train location data directly from the railroads. However, the railroads were unable to share this information. A third approach was to pull a signal from the flashing lights and bells at the crossing when they were activated. This approach, known as an interconnected crossing, is occasionally used to adjust traffic signals, so vehicles do not get stuck on the tracks when trains are approaching. Interconnection requires a physically-wired connection from the railroad's cabinet at the crossing to a city-owned cabinet on public right-of-way. WFPS concluded that this approach was infeasible since only a few crossings in Winnipeg were interconnected, and the interconnection data did not provide enough prediction to avoid blockages. They did not have the internal resources to manage this system.

Finally, WFPS turned to TRAINFO to help solve this problem. WFPS provided TRAINFO with a one-year history of EMS and Fire trips completed in 2018 as a basis to study and come up with a solution.

5.2.1 Additional Data Attributes for This Study

The data collection process involved:

1. EMS and Fire trip logs for 2018 (168,000 rows of data).
2. Collecting postal code geographical data to aggregate findings by area.
3. Collecting crossing blockage data from train detection sensors at various crossings in the city.
4. Generating and storing blockage impact data for blockage events in 2018.
5. Collecting occupancy distribution data for crossings in 2018.

We discuss their attributes as follows: The files that hold additional information that we will need to perform the analysis for this study include:

1. *EMS trip logs*, which contain the emergency call log data of the WFPS for 2018. It contains vehicle origin, destination coordinates, and time of departure and arrival of individual trips. WFPS removed any personally identifiable information prior to sharing this data set. We need the coordinates for the analysis phase to determine the route taken by the EMS. We then use the route to determine if they could have avoided delays (if any). Thus, having the following attributes:

- incident number: The number assigned to the incident, and is only present for calls that have turned into incidents. This value is an integer.
- unit number: Expressed as a character string.
- unit type: Whether the unit is a fire truck or EMS, expressed as a character string.
- unit origin: Location of the unit at the time it begins to respond to an incident, expressed in GPS coordinates.
- call location: Location of the incident the unit is dispatched to, expressed in GPS coordinates.
- dispatch time: The time they dispatched the unit. All the time values are timestamps.
- responding time: The time the unit begins to respond to an incident.
- at scene time: The time the vehicle arrives at the call location.
- depart scene time: The time the vehicle leaves the call location (EMS) or is cleared from the call (FIRE).
- at hospital time: The time the vehicle arrives at the hospital.
- depart hospital time: The time the vehicle leaves the hospital or is released from an incident.
- destination address: The address of the hospital if required, or the address of the EMS station if it is a return trip to the station. The value is a character string.

2. *EMS stations*, which usually assign emergency responders to stations. Stations are where the responders stage until dispatched to a call location. If we need to make summations at the station level, we would also need to collect location records of the EMS stations across the study region. In our study, the data contains the following attributes:
   - name: The name of the station, expressed as a character string.
   - address: The address of the station, expressed as a character string.
   - geometry: The geometry of the location, expressed as a point.
   - boundary: The geometry of the area around the geometry, expressed as a polygon. The polygon represents the compound (a circle with a radius of about 100 metres)
3. *Postal code data*, which summarize data by postal code, we also needed to ingest the Postal Code record for the study region. We generated the Postal Code records by querying the Google Maps Geocode API. We generated 17,371 postal code records with the following attributes:
   - name: The name of the postal code, expressed as a character string.
   - geometry: The geometric record of the area the postal code covers, expressed as a polygon.
4. *Crossing blockage data*, which were retrieved from TRAINFO's database. It comprised the start and end times of individual blockage events at crossings in 2018. Here, each entry will have the following attributes:

- crossing identifier: a unique id used to identify a particular grade crossing, expressed as a character string
- blockage start time: a timestamp of the time train blocks a grade crossing expressed as a timestamp.
- blockage end time: a timestamp of the time a blockage clears a grade crossing, expressed as a timestamp.
- train recording: ten-second audio of the train captured by the sensor, expressed as a waveform audio file format (WAV) audio blob file.

5. *Blockage impact*, which is a temporal component of blockage events. When a railway crossing is active, vehicles begin to queue up on both sides, anticipating the crossing to clear. Eventually, the barrier preventing vehicle movement lifts when the blockage clears. However, it will take some time for the queued-up vehicles to go over the crossing and for the traffic velocity to resume its usual rate before the initial blockage. When long blockages occur, the queue formed can impact surrounding streets. As a result, for each blockage event, a dynamic geofence models the queue that increases in size as the blockage duration increases. The size increases up to a point if the roadway reaches capacity and then shrinks as the theoretical vehicles move after the blockage. As a result, the blockage impact will have the following attributes:

   - blockage event identifier: a unique id used to identify a particular blockage event, expressed as an integer.
   - start time: a timestamp of when this impact started, expressed as a times-

tamp.

- end time: a timestamp of when this impact ended, expressed as a timestamp.
- geofence: a geo-spatial polygon covering the area of this impact, expressed as a polygon.

Note that an active crossing can impact a trip even though it does not interact with a crossing. We explained this by the queue formed when the crossing is active. We consider a trip impacted if the travel path is within the temporal geofence. So, we need also to determine if a blockage activity impacted trips.

6. *Occupancy distribution*, which was generated on a per-crossing basis using historical blockage records for the crossings monitored in 2018. The data attributes are summarized as follows:

   - crossing identifier: a code used to identify a crossing uniquely, expressed as a character string.
   - hour of day probabilities: a list of 24 probability values indicating the likelihood of the crossing being occupied each hour of the day, for each day of the week.

In summary, Table 5.1 outlines these additional data requirements and their structure.

Table 5.1: WFPS study: additional data sources, their formats, and structure

Data	**Owner**	**Structure**
Trip logs of Emergency Response Units for 2018	WFPS	structured
Occupancy distribution at crossings	TRAINFO	structured
Historical crossing blockages	TRAINFO	unstructured
Spatial blockage impacts at crossings	TRAINFO	semi-structured
Postal code geo-spatial records	TRAINFO	semi-structured
EMS station location information	WFPS & TRAINFO	semi-structured

5.2.2 Hybrid Model for WFPS Study

Firstly, we stored these raw data in Table 5.1 in the data lake. For this study, we used MongoDB as the document-oriented database, MySQL for the relational database and Neptune for the graph database. For each data set, we either extracted them into the graph or loaded them into the relational database. We should mention that we used the approach in Chapter 4 to build the basics of the graph. Transport Canada provided the grade crossing inventory mentioned in Table 4.1. Each crossing in that data set has a unique Transport Canada ID number. When we discussed the additional attributes above, we referred to this ID as the crossing identifier.

We use Table 5.2 to describe how we inserted the additional data sets into the database.

Table 5.2: How additional data fit in the hybrid model

Data	**MySQL**	**Neptune Graph**	**Comments**
Crossing blockages	✓	✓	Blockage start and end dates, and crossing identifier were stored in MySQL. The train recordings were stored in a collection in MongoDB with a reference to the location stored as a node attribute on the graph.
Trip logs	✓		Transactional in nature
Spatial blockages at crossings	✓		Loaded into MySQL, referenced crossing identifier.
Occupancy distribution at crossings	✓		same as above
Origin-destination records	✓		This is an additional table for caching graph traversals from one node to another. It represents a trip between a unique combination of origin-destination postal codes.
Postal code geo-spatial records		✓	Each node has a single value postal code string attribute based on the coordinates. Edges also has a postal code attribute with a list of postal code strings for lines that appear in multiple postal codes.
EMS station location information		✓	This was a node attribute. It was assigned to the closest intersection node to the EMS station

In addition to this model's primary analysis, we organized the entire trip log using a star schema model as fact and dimension tables. We used this to show the city a classification of their trips throughout the year by different dimensions. Table 5.3 shows these dimensions, yielding the resulting schema in Figure 5.2.

Table 5.3: Aggregation of WFPS trips

Name	**Fact/Dimension Table**	**Description**
Trips	Fact	Holds aggregated numeric facts on travel time and distance based on the dimension the data is filtered by
Station	Dimension	For aggregating data based on EMS stations
TimeOfDay	Dimension	Time of day when a trip took place. There will be 24 of this for each hour of the day
Vehicle Class	Dimension	Classification of Emergency Vehicle Response. Fire or Paramedic Vehicle
Origin-destination traversal	Dimension	Holds traversal records from Point A to Point B where A and B are two Postal Codes
StationPostal-CodeTraversal	Dimension	Holds traversal records from one Station to a Postal Code

5.3 The Municipalities of Charleston Case study in the USA

As mentioned earlier, this study is in the beginning stages. However, there are preliminary results using the hybrid database architecture that we would like to highlight. In this case study, we did not have as many data inputs as the WFPS study. The data collected two **CSV** files containing:

1. EMS and Fire trip logs for 2021 (80,000 rows of data), and
2. AVL Pings of Emergency Response Units as they respond to events (233,000 rows of data).

The EMS log had similar attributes as the WFPS data. AVL ping data contain individual trip identifiers, a timestamp, and the latitude and longitude coordinates

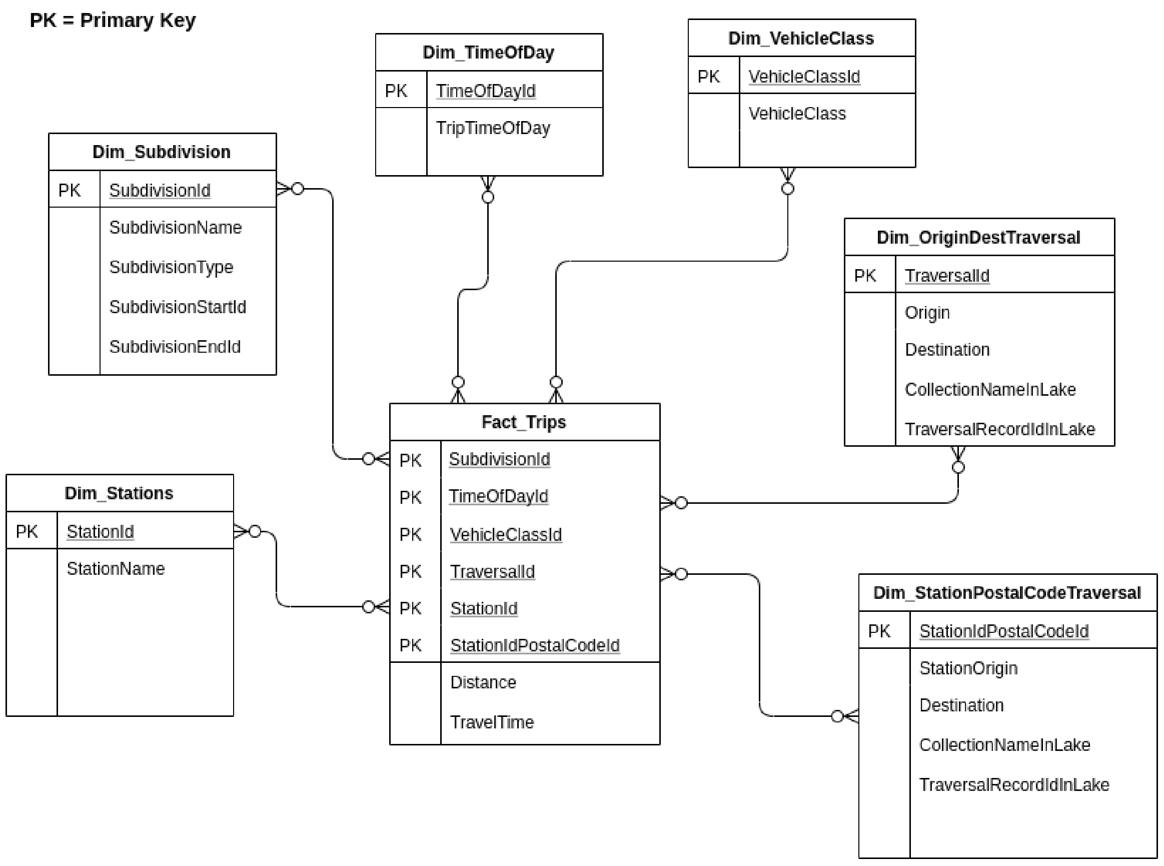


Figure 5.2: ER-Diagram of WFPS trips using a star schema

of the vehicle at that time. The AVL explicitly reveals the route taken by the EMS. Thus, having the following attributes:

- incident number: The number assigned to the incident, expressed as an integer.
- unit number: Unit number, expressed as a character string.
- unit location: Location of the unit at the time of the ping, expressed as a string.
- unit coordinates: Location of the unit at the ping time, expressed in GPS coordinates.

- timestamp: The time of the ping, expressed as a timestamp.

Consequently, Table 5.4 outlines these additional data requirements and their structure.

Table 5.4: Charleston Municipalities Study: Additional data sources, their formats, and structure

Data	**Owner**	**Structure**
Trip logs of Emergency Response Units for 2021	Charleston	structured
AVL Pings of Emergency Response Units for 2021	Charleston	structured

5.3.1 Hybrid Model for Charleston Study

Both data were stored in tables as they are transactional. We should mention that, unlike the WFPS study, TRAINFO did not have any sensors deployed in Charleston in 2021. The lack of sensors made it impossible to confirm potentially impacted trips as impacted. Nevertheless, through the use of the graph database, we were able to make educated inferences on potentially impacted trips. We flagged trips where the actual travel time was at least 1.5 times greater than the expected travel time as potentially impacted. The process involved using the graph to compute the expected time of travel given the AVL ping data.

We reconstructed the path travelled by each unit by using its periodic ping data. With the reconstructed path, we queried the same edges on the graph. We computed the total expected travel time using the posted speed limit and length of the road segments. Those values are edge properties retrieved from the shapefiles. We then compared the expected travel times to the actual trip times. Trips were then flagged as potentially impacted if they met two criteria:

1. the travel path went over a crossing node, and
2. the actual travel time was at least 1.5 times greater than the expected travel time

5.4 Discussion

We have demonstrated that this database architecture can find more interesting information from a large data set. The insights from the data can help make crucial decisions. As seen from the case studies, identifying and predicting these blockages and regions of impact is a massive advantage in the effort to save lives.

The insights also help inform fleet management planning and assessing cities' Emergency Response systems. Given the results from such studies, cities can make more informed decisions to relocate responder stations to areas where units will encounter fewer active crossings. They could also reallocate units to different staging locations to mitigate the risk of them getting blocked. Decisions like these would be better aided with this information now at their disposal, heavily aided by our hybrid database architecture. We have shown that the database can be extensible for analysis in different cities. We have analyzed data from Winnipeg and Charleston using the hybrid database model, strengthening the importance of using a hybrid database for modern transportation analysis.

Our solution is not just a database. We can use it to perform analyses that can save lives. Our hybrid design supporting this type of study highlights its importance in the real world. As a footnote, the results from both case studies encouraged the FRA to expand this research effort to a few other cities, including Houston, Texas; Chattanooga, Tennessee and Worcester, Massachusetts.

5.5 Summary

This chapter outlined two real-life data sets used in our EMS transportation analysis. We showed that our approach could be used in EMS transportation analytics for different cities and across countries. In particular, we used our hybrid database architecture to support studies conducted in both the city of Winnipeg, Canada, and Charleston, USA. One study contained an additional six data sets, the other an additional two. We show that ultimately, it does not matter the city where we conduct the study; the process generally remains the same.

With the help of the graph and the relational database, we made several observations about first responder trips. Such observations could have been near impossible without our hybrid architecture.

Finally, we showed that the observations from the case studies helped provide two cities with decision-support making tools to manage and improve their emergency response process.

Chapter 6

Evaluation

In the previous chapters, we described how the relational database was inefficient when handling well-connected data like those in shapefiles. We suggested that having the data in a graph was more efficient. We proposed that our hybrid architecture is a more effective solution overall for supporting transportation analysis in the modern-day. This chapter determines the effectiveness of our claim. We describe the testing experiments and results, which will help to conclude whether this proposed architecture improves our ability to support the modern-day data requirements or limits it.

We conducted our evaluation in three phases. The first phase is on the features. The second phase of our assessment measured performance in terms of run times of each database approach. The final stage of our evaluation measured the usefulness of the hybrid database in supporting studies to generate metrics to help EMS administrators minimize their response times.

6.1 Feature-wise Evaluation

This evaluation phase considers a list of desirable features for a database that would support this kind of study. We attempt to show how the single-style database approaches fared compared to the hybrid database comprised of both databases. Effectively, we compared the features of individual components of the hybrid model alone to our hybrid model, which consists of both.

As Table 6.1 shows, the hybrid approach meets all the the outlined criteria compared to each database approach on their own.

Table 6.1: Feature-wise comparison of the three database approaches

Criteria	**Relational database**	**Graph database**	**Hybrid database**
Easily support visualization [VMZ+10; Bik18]	Yes	No	Yes
Store and manage structured data efficiently [VMZ+10; Bje18; ZBP+18]	Yes	No	Yes
Store and manage semi-structured data efficiently [Bje18; VML20]	No	Yes	Yes
Store and manage unstructured data efficiently [VMZ+10; Bje18; VAOA22]	No	Yes	Yes
Easily modify schema [VMZ+10; SMA+18]	No	Yes	Yes
Easily model geo-spatial data [WE18; VAOA22]	No	Yes	Yes

6.2 Performance on Real-World Data

The next phase of our evaluation was runtime performance using real-world data. We had shown in Chapter 5 how the hybrid model was used to support two studies in Winnipeg and Charleston. However, since we performed extensive studies in

Winnipeg, we decided to use that study to evaluate the performance of our hybrid approach compared to the single-style database approaches. Hence, we calculated four metrics of interest using all three systems and recorded the time it took to generate our results.

6.2.1 Experiment Setup and Dataset Details

We ran all the tests using a computer running Ubuntu 20.04 LTS as the primary operating system. The CPU was an Intel Core i7 with eight cores clocked at 2.8 GHz. The machine also had 32 GB of RAM and a Solid State Drive.

The total data obtained was 168,419 WFPS trip records from the Winnipeg Fire and Paramedics in 2018. Hence, the data comprises trips with different emergencies and road conditions from all four seasons. We performed some quality control as follows:

- We removed trips where the units did not have an assigned station.
- We removed trips whose origins and destination were at the same location.
- We removed trips without unique responder identification or unit numbers.
- We removed trips missing any of the following values: origin, destination or waypoint location, unit number or type, incident number, time of call

After the data pruning, we broke down the remaining trips into legs, resulting in 196,265 leg records. In order to fully perform the analysis conducted in Chapter 5, we also obtained the following data sets:

- 11,630 time-based blockage impact records from TRAINFO.
- 13,104 occupancy distribution records from TRAINFO.
- 17,371 postal code records from TRAINFO.
- 4,446 rail crossing records from National Railway Network of Canada.
- 2,262 rail crossing records from Transport Canada.
- 5,774 rail track records from National Railway Network of Canada.
- 47 EMS station records, which includes hospitals and staging areas from WFPS.

6.2.2 Test Overview

We mentioned in Chapter 4 that we could store all of the data in a relational database or graph, but both strategies would be inefficient. We evaluate this claim by computing the following metrics:

1. Crossing Risk Scores of top three crossings: This calculation involves evaluating every trip undergone by first responders and counting the frequency of the crossings that interacted with the trips. We picked three crossings with the highest frequency.
2. Area Risk Scores of top three city areas: This calculation involves the product of two distributions. The first distribution is the crossing occupancy distribution used for the crossing risk. The second distribution is the responder-crossing interaction distribution. The responder-crossing interaction distribution is a

subset of the responder destinations. We picked the highest three areas of the city.

3. Number of responders delayed by a train per week: Aggregating trips confirmed as delayed per week.

4. Average delay per trip.

We intend to compare the possibilities of calculating these four metrics with the support of:

1. The relational database approach only

2. The graph database approach only

3. The hybrid model

Concerning our data architecture shown in Figure 4.2, when we performed our analysis with individual database components of our hybrid model, we omitted one of the two databases in the data warehouse. Specifically, with the relational model, we used the Google Maps application programming interface (API) to retrieve the fastest route between the origins and destinations of the WFPS legs. It was impractical to find the path solely using the relational model. In comparison, in the other two database approaches containing the graph, we used the graph to find the fastest route by querying the edges between origins and destinations. Since the speed limits are characteristics of the edge, we could still find the fastest route.

We ran the analysis in Chapter 5 two additional times. One with the support of only a relational database and another with just a graph database. We recorded the run time to complete getting each result. In cases where errors occurred during communication timeout, we used retry functions to repeat the queries, adding to the run times.

6.2.3 Recorded Run Times for Each Database Approach

There are some sequential preliminary steps in order to be able to compute the desired four metrics. They include:

1. Getting fastest path, best travel time for each leg.
2. Examine paths and flag legs that go over (a) crossing(s) as interacted with crossing.
3. Cross-reference flagged legs with historical blockage event records at crossings to see if there were trains at the time of travel. Flag as potentially impacted.
4. If the actual travel time of the leg was a threshold higher than the expected travel time, we flag the trip as confirmed impact by a train.

Table 6.2 shows how long it takes to run each model step for the study. We automated this process using Python scripts and recorded the time it took the programs to run to completion. We see that the relational database has the worst performance in Step 1.

In Step 2 of Table 6.2, we flagged legs that interacted with crossings. We first converted the path from Google's API into points in the relational database approach. We then iteratively checked each point to see if it touched a crossing in the crossing table. We see that the database approaches containing graphs have no values because

Table 6.2: Run times for preliminary steps when computing metrics for study

	Run times per database style		
Step / DB style	**Relational only**	**Graph only**	**Hybrid**
1. Fastest path, travel times	738 minutes	100 minutes	74 minutes
2. Flag interacted legs	459 minutes	-	–
3. Flag potentially impacted legs	414 minutes	750 minutes	415 minutes
4. Flag confirmed impacted legs	54 minutes	139 minutes	54 minutes
TOTAL	**1,665 minutes**	**989 minutes**	**543 minutes**

this step is included in Step 1 while finding the travel path between the origin and destination of WFPS legs.

In Step 3, we see that the relational database approach is the fastest. It was the fastest because we are cross-referencing tabular data to find a connection at this stage. As relational databases excel in this situation due to indexes, we expected this to be the fastest approach. In the graph-only approach, we had to search the blockage data for that crossing on the node itself. Due to the load of data on the node and the amount of blockage information stored on the node, we see that we have the most run-time when using this approach. Compared to the hybrid approach, it was the second-fastest. We attribute the longer run time to latency and communication between databases.

In Step 4, we also see that the relational-database-only approach performed the best due to the transactional nature of the task. The same goes for the hybrid approach. The graph-intensive approach was significantly worse because we had to run through all the legs on the graph again. We did this to confirm expected travel times with actual trip travel times. In a tabular structure, this is a much simpler operation.

From Table 6.3, we can see that the relational-database-only approach excelled when computing aggregation metrics like the number of responders delayed and the average delay. As we expected, the hybrid approach was significantly faster overall when calculating the area and crossing risk scores because these are primarily geographic-driven computations. The graph-only method slightly underperforms the hybrid due to the load on the graph. However, both graph approaches underperform when calculating the other two metrics due to the complexity of storing and retrieving and cross-referencing the tabular records when calculating those metrics.

Table 6.3: Run times to generate results using three different database approaches

	Run times per database style		
Operation / DB style	**Relational only**	**Graph only**	**Hybrid**
Crossing risk score	117 minutes	51 minutes	36 minutes
Area risk score	157 minutes	49 minutes	40 minutes
#responders delayed per wk	60 minutes	492 minutes	62 minutes
Average delay per trip	45 minutes	284 minutes	44 minutes
TOTAL	**379 minutes**	**876 minutes**	**182 minutes**

Additionally, we explored the differences between the graph structure in a graph-only approach and a hybrid approach. Table 6.4 shows the average query response times when we used the graph-database-only approach compared to the hybrid ap-

Table 6.4: Query response times and size of graph with both approaches

Metric	**Graph-intensive**	**Hybrid approach**
Size of entire graph	69.4 MB	18.3 MB
Average query response time	7.0046 seconds	4.3852 seconds

proach. As expected, we see that we are spending more time getting data back during graph retrievals when we store lots of information on the nodes and edges of the graph. We also see in Table 6.4 that the graph size is almost four times bigger in the graph-only approach than in the hybrid approach, which supports our claim that the graph load would affect latency. It is important to note that this gain in performance would be more significant with a much bigger graph that represents a larger area, for example, the Greater Vancouver area.

Our resulting graph was the same for both the graph-database-only and hybrid approaches. Table 6.5 shows the graph's composition of nodes and edges.

Table 6.5: Graph database composition counts

Component	**Count**	**Total**
Rail nodes	8,826	
Road nodes	48,330	**57,156 nodes**
Rail edges	5,774	
Road edges	35,326	**41,100 edges**

6.3 Usefulness of the Hybrid Model

The last phase of our evaluation was its usefulness. In this section, we show how the hybrid model supports the EMS study to minimize response times and how the results can be helpful to EMS administrators. We discuss the results generated in both studies conducted in Winnipeg and Charleston.

6.3.1 Winnipeg, MB, Canada Results

Let us analyze the following types of trips:

1. Identifying trips that went over crossing(s): We have a start and end position for each trip. We then use the graph to find the **fastest path** between the start and end position nodes on the graph. The path is a collection of nodes and edges on the graph. Suppose any of the nodes is a grade crossing. In that case, we flag the trip row on the relational database table as "possibly impacted". In turn, we will increase the count attribute for the number of trips the crossing node encountered.

2. Identifying delayed trips: The expected travel time is 240 seconds, as the standard requires for any EMS trip. Identifying delayed trips is a multi-stage process:

   (a) Retrieve travel time information of "possibly impacted" trips from the trips table.

   (b) We then pick those trips with travel time above their expected travel time. Next, we retrieve their route from the origin-destination table. We correlate the route to the spatial blockage impacts table to verify if the route was within any geofence within an active crossing blockage time window. If the route is within the geofence, we flag that trip in the trips table as "**confirmed impacted**".

3. Rerouting trips: The last phase of the analysis is rerouting trips confirmed as impacted by an active crossing, then calculating the travel time difference. We selected an active crossing node to avoid, then ran the traversal on the graph. We then calculated the expected travel time for this new route. Figure 6.1 illustrates this process at the Shaftesbury crossing. A senior complex near the rail crossing is a regular first responder destination with 11 calls per week. First responders that respond to emergencies at this complex and other locations north of the tracks typically travel north on Kenaston Blvd. They turn west on Wilkes Ave and then north on Shaftesbury over the tracks. When the crossing is clear, this is usually the fastest route. However, when the crossing is blocked, an alternate route would be continuing north on Kenaston, turning west on Grant Ave, and then turning onto Shaftesbury. This alternate route is nearly 2 miles further and takes about four minutes longer to reach the senior complex. However, if there were a 5-minute train at the Shaftesbury crossing, it would be at least one minute faster.

The model produced the Crossing Risk Scores and Area Risk Scores. The Crossing Risk Scores, shown in Figure 6.2 reveal which crossings cause the most risk for first responders. This value is in units delayed per month. The Area Risk Scores, shown in Figure 6.3 reveal which areas of the city had the highest risk of experiencing first

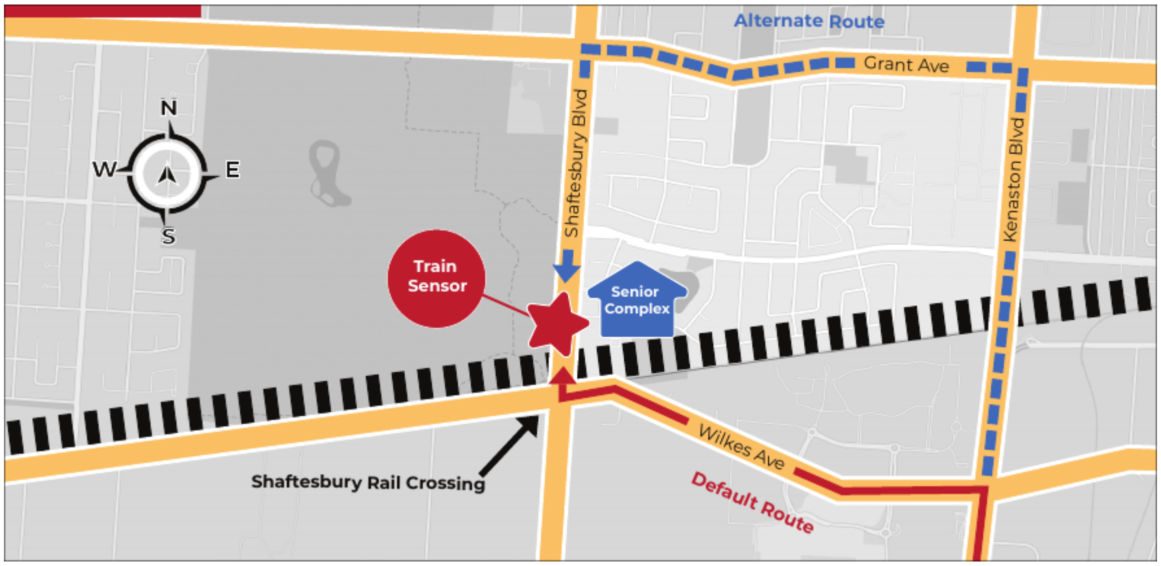


Figure 6.1: Rerouting simulation to calculate time savings

responder delays due to trains. This value is in units delayed per month. Crossing Risk Scores larger than 1.5 units delayed per month and Area Risk Scores larger than 0.5 units delayed per month indicate a high risk.

The EMS Risk Model produced detailed rail crossing blockage statistics and estimated the potential benefits of using predictive train information to re-route around blocked crossings. The Shaftesbury Blvd rail crossing provides a useful example of this feature. Figures 6.4, 6.5, and 6.6 illustrate train information statistics for the Shaftesbury rail crossing. The model revealed several other important findings, including:

1. 3 crossings in Shaftesbury Blvd, Panet Road, and Marion St have a high Crossing Risk Score (exceeding 1.5 units delayed per month) and contributed to more than 30% of the delayed trips.

2. 3 areas of the city in R3P2T4, R2J4L4, and R3R3X2 have a high Area Risk

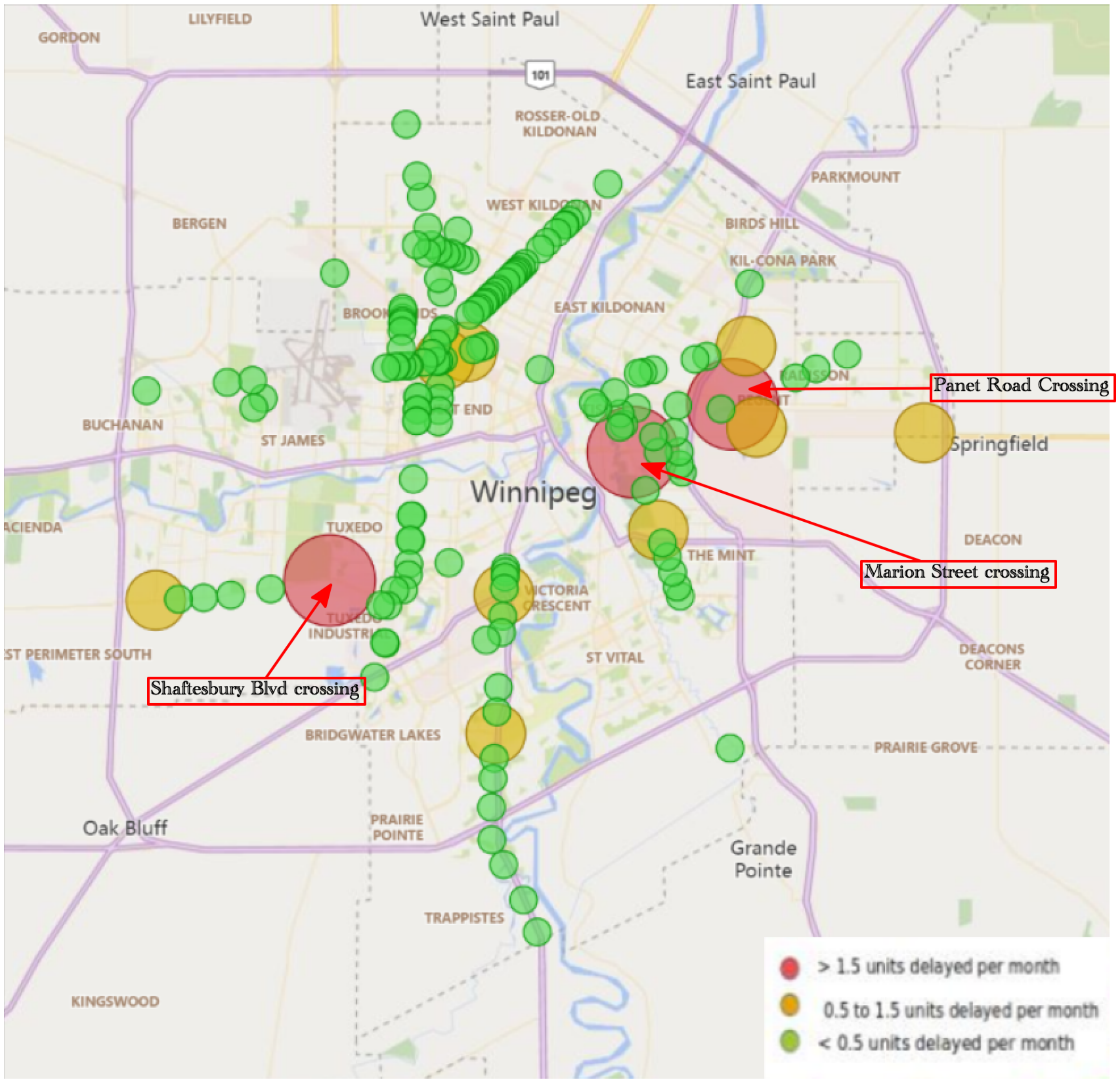


Figure 6.2: Crossing risk scores—crossings that impact first responders most frequently

Score (exceeding 0.5 units delayed per month), and

(a) 26% of the delayed trips were servicing 8 out of 2,688 areas of the city

3. On average, nearly five first responder trips are delayed by a train each week. Their average delay is 3 minutes per event

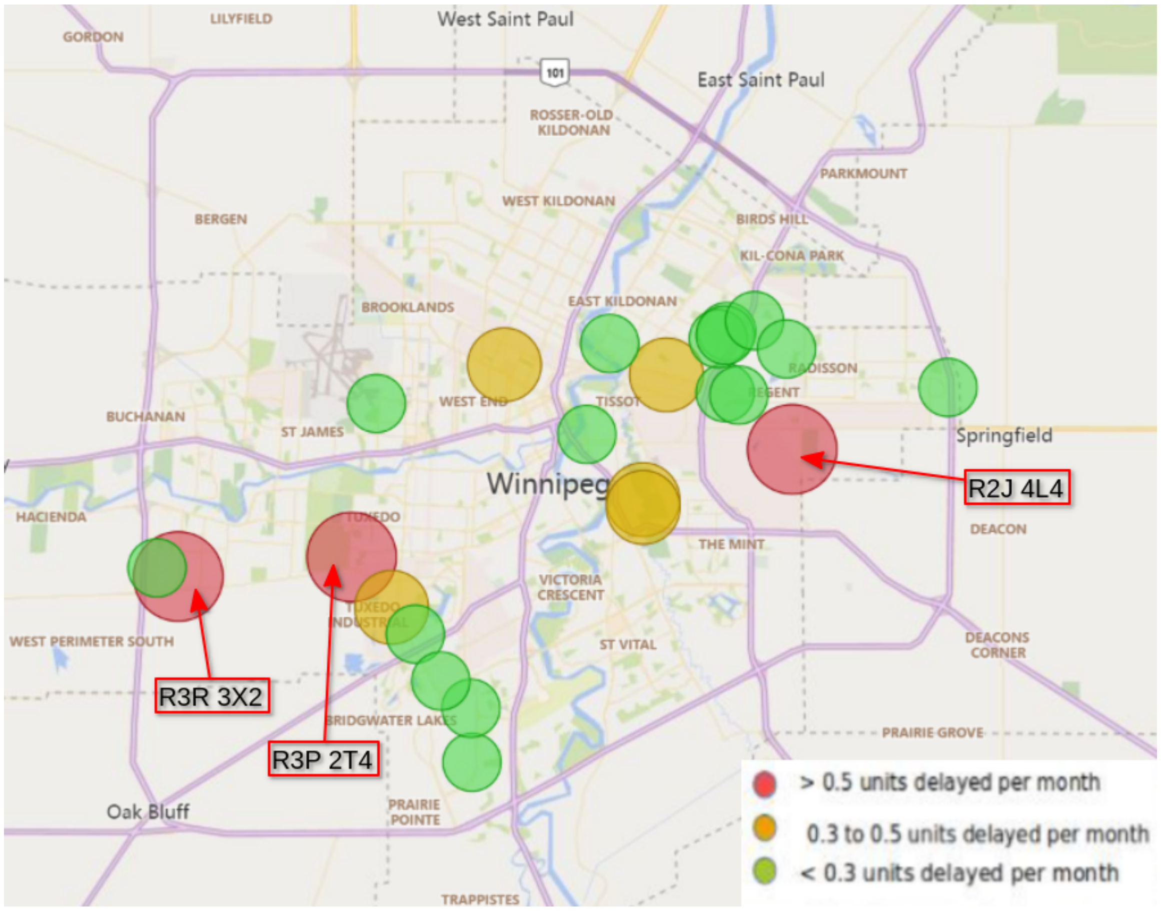

R2J 4L4
R3R 3X2
R3P 2T4
● > 0.5 units delayed per month
● 0.3 to 0.5 units delayed per month
● < 0.3 units delayed per month
Winnipeg
West Saint Paul
East Saint Paul
Springfield
Oak Bluff

Figure 6.3: Area risk scores—areas most impacted by first responders delayed by trains

4. 10 of the 165 crossings in Winnipeg contribute to 60% of the delayed responders
5. 10% of every EMS interaction with Shaftesbury and Marion crossings resulted in a delayed responder
6. In particular to the Shaftesbury crossing:
   (a) 3 first responder trips per week travel over the tracks
   (b) Trains delay 1-in-5 trips

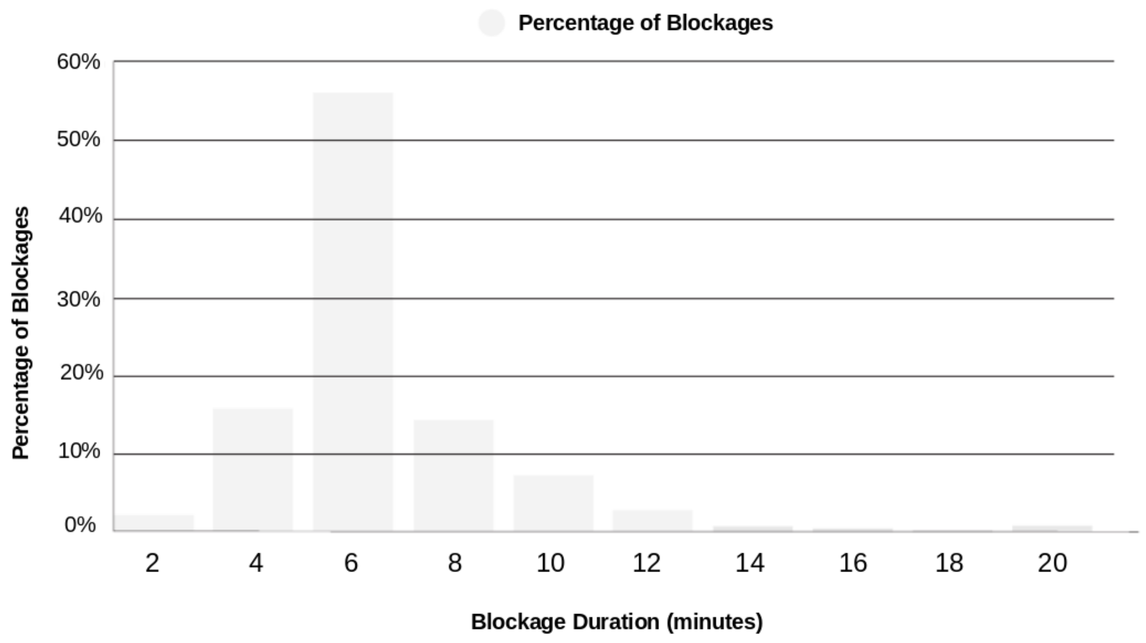


Figure 6.4: Shaftesbury Blvd crossing: percent of blockages by blockage duration

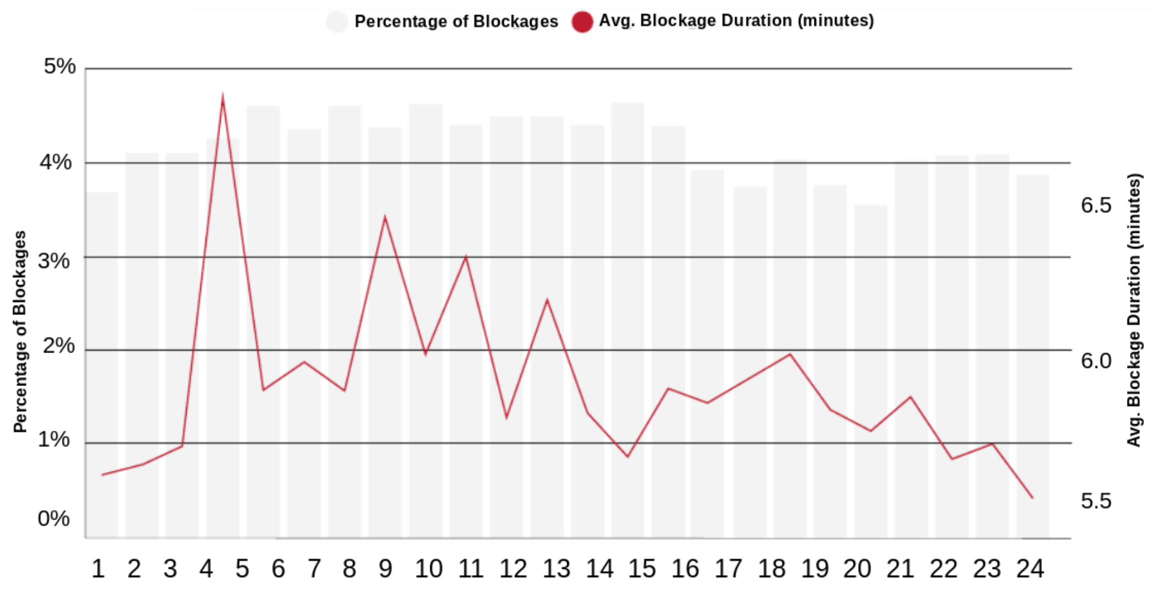


Figure 6.5: Shaftesbury Blvd crossing: Number of blockages and average duration by hour of day

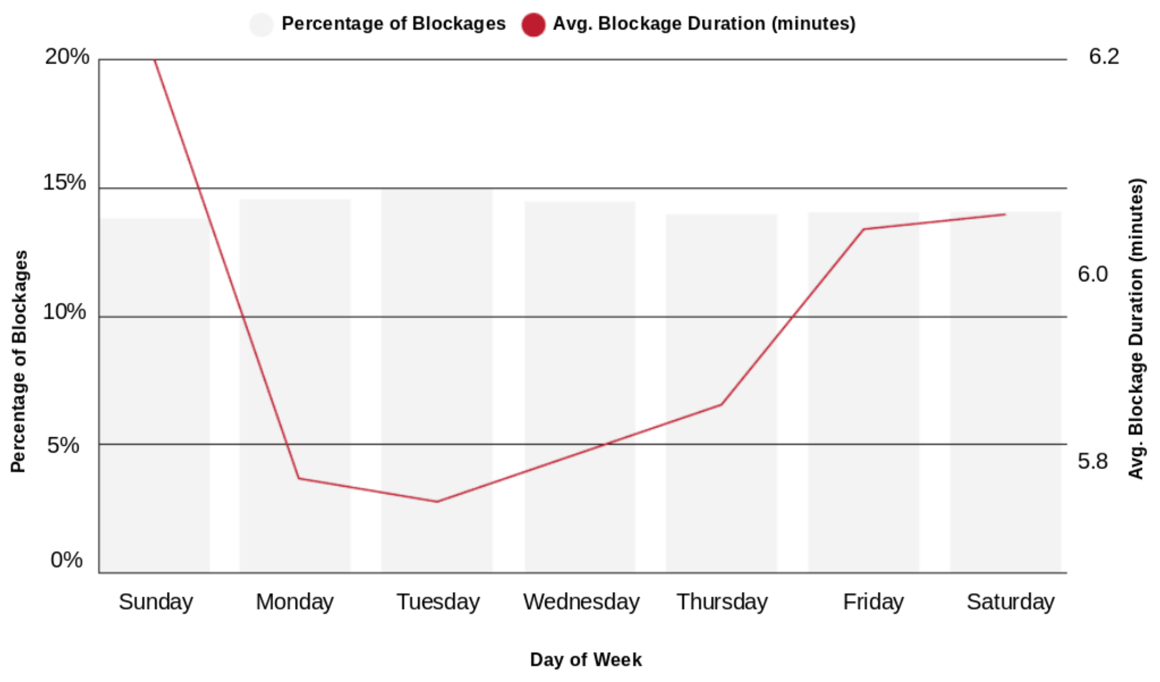


Figure 6.6: Shaftesbury Blvd crossing: number of blockages and average duration by day of week

(c) Average delay per trip of 184 seconds

(d) 81% of delayed trips would have saved time by re-routing

(e) Average time savings for re-routed trips would have been 182 seconds

6.3.2 Charleston, SC, USA Results

Since we did not have actual crossing blockage data to cross-reference, we could only go as far as producing railway crossing interaction statistics, given the data available. Table 6.6 contains a summarized set of results produced by the model based on first responder trips in 2021.

Table 6.6: Crossings with the most interactions with EMS trips in 2021

Crossing	**Interactions per week**
Ashely Phosphate Road	18.18
Hillcrest Drive	9.97
Meeting Street Road	7.31
Ladson Road	3.54
Stewart Avenue	3.23
All Crossings (Total)	**58.23**

6.4 Discussion

The hybrid approach shows an overall significant performance gain when calculating the four desired metrics. We used the relational database to retrieve and calculate the transactional data. Then, we used the graph to identify the related data quickly. We are therefore getting the best of both worlds. Besides the performance gain in using the graph, we should recall that we used the Google Maps API to get routing information using the relational-database-only approach. This process costs many dollars. Not to mention, Google has a strict policy on data scraping. As a result, when using their API, we had to introduce sleep intervals during these queries to avoid getting our account flagged for violating Google's usage policy.

The graph-only approach is also not desirable as it does some things very poorly. In our case, performing tasks as simple as finding the average delay per trip and the number of responders delayed weekly was incredibly cumbersome. Our program had to query the graph numerous times to get this data. In contrast, this was a relatively more straightforward procedure using the relational database. Table 6.4 also shows that the speed of querying the graph improves with a lighter load on the nodes and edges. This result supports our claim that one database style is not good enough for the demands of modern-day applications.

6.4.1 Possible Issues with the Hybrid Approach

There were two concerns we had when we proposed this hybrid approach. They were communication issues between the database components and the consistency of data across the databases. There ultimately is a concern for communication failure when dealing with non-isolated software. This concern is not limited to the database world. Hence, as with good software design practices, we have to engineer our applications to catch those instances and either retry the communication or report it. In our evaluation, we wrote retry functions that perform the communication task up to three times for every failure. When all three attempts fail, we report the event in the logs. We saw that we were getting primarily MySQL-related errors due to too many parallel connections to the database simultaneously. We explored tuning the number of parallel processes and having each process manage a pool of threads for communication. We saw that in many cases, with an increased number of processes, we were getting many failed MySQL database connection errors. We settled on running a maximum of 25 processes in parallel. At 25 processes, we stopped getting these connection failure errors. The cost of doing this was that we had a much slower process, as we can see from the high run-time numbers. Since we used the results from the study offline to support decisions after the fact, we felt this was an acceptable trade-off.

The other concern was data consistency among databases. In our hybrid database design, we are not replicating data, as the data we store across database components is mainly disjoint. The databases serve as a historical archive, where we keep adding data and are not deleting data.

6.5 Summary

In this chapter, we discussed the steps it takes to compute the metrics of interest we saw in Chapter 5. We outlined the four steps of the EMS Risk model. We then attempted these four steps using our hybrid approach, the relational-database-only approach and the graph-database-only approach. We measured and compared the run times of all three approaches. The numbers in Tables 6.2 and 6.3 show that overall, the hybrid approach performs better than the other two for this study.

We also show in Table 6.4 that although the graph is fast at working with connected data, the load on the graph is also crucial for performance. As such, one should decide which data attributes make more sense to store on the graph and which ones should not.

Finally, we discussed challenges with MySQL when working with all three database approaches. Mainly how tuning the number of parallel processes querying the database impacted our program run times. We found that keeping parallel processes to a maximum of 25 yielded a situation where we had no more loss of connections to the MySQL database. The trade-off for this was longer run times, which we deemed acceptable for our use case.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This thesis aimed to demonstrate that a single-style database architecture was no longer an acceptable solution for modern-day transportation analytics. Our position was that relational databases alone could not handle unstructured data; non-relational databases alone would underperform when handling structured data. To the best of our knowledge, current work in this area of research focused heavily on demonstrating the power of NoSQL database solutions. Specifically, they focused on NoSQL's handling of unstructured data while ignoring the obvious benefits of relational databases. Some bodies of work identify the need to use both styles of databases for handling both data structures. However, they show little or no real-world application on a large enough data set. Our model addresses both concerns by leveraging the strengths of relational and non-relational databases in our hybrid architecture. We also went further to show its applicability in the real world.

Additionally, we showed that the approach is not geographically limited as studies were on data from two countries. Even further, we can state that the generic database architecture proposed in Chapter 4 is a sound solution for any transportation analytics. We can make this claim because we showed that it ultimately does not matter if the studies are on first responder travel activity or trucking transportation. This hybrid architecture can still support it better than a single-style database.

We believe that our work has great value for current application development and future research. We used this hybrid database design to study the risk of delay to emergency responders and evaluated alternative routing solutions. The results will go a long way to helping cities understand their risks of delay. It will positively transform their emergency response process and save lives. From a business perspective, we uncovered several essential factors about notorious crossings and areas in Winnipeg in the WFPS Case study. Such revelations can help resource planners focus resources on areas that need more immediate attention. We can take this design strategy for analysis and apply it to different emergency response processes in North America. We can also apply it to a different field of the transportation industry altogether.

In Section 1.1, we stated the goal of our thesis with three questions. We answered these research questions in Chapters 4 and 5:

Q1: How well can the hybrid DBMS handle traditional structured data?

A1: The hybrid data architecture comprises a relational database (Section 4.6.1), and we loaded traditional structured data into the relational database component (in Chapter 5).

Q2: How well can the hybrid DBMS handle semi-structured and unstructured data?

A2: We observed in Section 4.6.1 that we used the Data Lake to store meta-graph sources and the semi-structured shapefiles files from OSM. We then extracted their attributes and loaded them onto the graph. In Table 5.2, we mentioned that we stored the audio recordings of trains at crossings in the data lake. Moreover, the graph data generated is unstructured and stored in the graph database component.

Q3: How well can the hybrid DBMS handle some real-life applications with structured, semi-structured and unstructured data?

A3: Yes, we can. In Chapter 5, we showed that we could use the DBMS to support two studies. In both case studies we had all three kinds of data as input. We used the hybrid DBMS to support evaluating the emergency response process in the city of Winnipeg, Manitoba and the municipalities of Charleston, South Carolina.

Both databases used in this approach pooled their strengths to provide an ultimately better solution than one or the other. We saw the results from the experiments conducted in Chapter 6 to support this.

7.2 Future Work

While we believe we have satisfactorily answered the questions we set to address in this research, there are still possibilities for future work. In Section 4.5, we mentioned that a limitation of using the graph-only architecture is that there are currently little to no visualization support tools for graph databases. A visualization tool that seamlessly supports the graph's data structure could remove the need for storing summary tables on the relational database.

Further probing on the WFPS study revealed that the EMS data for 2018 informed us of some social demographics of the population. We noticed that the top five postal codes had adult living facilities within the boundary of all the first responders' destination locations. One of those five postal codes was also reported to have a high crime rate by the Winnipeg Police Department that year. From a data management perspective, the question arises as to how, if possible, our hybrid architecture can ingest social demographic data and make sense of it? Could the data be stored as graph attributes or in a tabular structure?

Chapter 6 also discussed the tradeoff between error rates and run times. With more research into the problem of lost MySQL connections, we can look to improve our run times by increasing the number of processes working in parallel without having an increased error rate.

In our conclusion, we mentioned we could use the architecture proposed in Chapter 4 to perform a non-first-responder-specific analysis. Exploring how this design can support other transportation analyses would be valuable to this research. For example, it is common knowledge that trains transport a significant portion of North America's goods. It would be interesting to analyze how the movement of crop yields from the farms, for example, affects demand and supply.

Lastly, while we believe that having a document database benefits our hybrid model. A scenario where we eliminate the need for a document database serving as a data lake could also be worth exploring.

Bibliography

[AC18] Jacky Akoka and Isabelle Comyn-Wattiau. Roundtrip engineering of NoSQL databases. *Enterprise Modelling and Information Systems Architectures (EMISAJ)*, 13:281–292, 2018.

[ACP17] Jacky Akoka, Isabelle Comyn-Wattiau, and Nicolas Prat. A four V's design approach of NoSQL graph databases. In Sergio de Cesare and Ulrich Frank, editors, *Advances in Conceptual Modeling*, pages 58–68, Cham, 2017. Springer International Publishing.

[ADF$^+$14] Marcelo Arenas, Gonzalo I Diaz, Achille Fokoue, Anastasios Kementsietsidis, and Kavitha Srinivas. A principled approach to bridging the gap between graph data and their schemas. *Proc. VLDB Endow.*, 7(8):601–612, April 2014.

[Ama21] Amazon Web Services. Amazon dynamodb database service, 2021.

[Ang18] Renzo Angles. The property graph database model. In *AMW*, 2018.

[APR$^+$16] V. Ashok, T. Priyadarshini, N. Raghavi, B. Rajashree, and S. Sanjana. A secure freight tracking system in rails using gps technology. In *2016*

Second International Conference on Science Technology Engineering and Management (ICONSTEM), pages 47–50, March 2016.

[Bes22] Pedro Francisco Mendes Bessa. Transportation management in an era of big data: from data to knowledge. Master's thesis, Faculdade de Engenharia da Universidade do Porto (FEUP), Portugal, July 2022.

[Bik18] Nikos Bikakis. Big data visualization tools. *arXiv preprint arXiv:1801.08336*, 2018.

[Bje18] Srdja Bjeladinovic. A fresh approach for hybrid SQL/NoSQL database design based on data structuredness. *Enterprise Information Systems*, 12(8-9):1202–1220, 2018.

[BK02] Thomas H Blackwell and Jay S Kaufman. Response time effectiveness: comparison of response time and survival in an urban emergency medical services system. *Academic Emergency Medicine*, 9(4):288–295, 2002.

[CHB$^+$20] Zhiyong Cui, Kristian Henrickson, Salvatore Antonio Biancardo, Ziyuan Pu, and Yinhai Wang. Establishing multisource data-integration framework for transportation data analytics. *Journal of Transportation Engineering, Part A: Systems*, 146(5):04020024, 2020.

[CR17] Yifeng Chen and Laurence R. Rilett. Train data collection and arrival time prediction system for highway–rail grade crossings. *Transportation Research Record*, 2608(1):36–45, 2017.

[CS17] Carlos Costa and Maribel Yasmina Santos. Big data: State-of-the-art concepts, techniques, technologies, modeling approaches and research challenges. *IJCS*, 4, 2017.

[Dar12] Hugh Darwen. The relational model: Beginning of an era. *IEEE Annals of the History of Computing*, 34(4):30–37, 2012.

[DCL18] Ali Davoudian, Liu Chen, and Mengchi Liu. A survey on NoSQL stores. *ACM Computing Surveys (CSUR)*, 51(2):1–43, 2018.

[DE20] DB-Engines. DB-engines ranking, 2020.

[Del20] José Delgado. Interoperability effect in big data. *Handbook of Smart Cities*, pages 1–28, 2020.

[Dul15] Jenna Dulewich. Winnipeg's railway crossings, 2015. Winnipeg Free Press.

[EHG+21] Ahmed Eldawy, Vagelis Hristidis, Saheli Ghosh, Majid Saeedan, Akil Sevim, AB Siddique, Samriddhi Singla, Ganesh Sivaram, Tin Vu, and Yaming Zhang. Beast: Scalable exploratory analytics on spatio-temporal data. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 3796–3807, 2021.

[Env98] Environmental Systems Research Institute (ESRI). Shapefile technical description, 1998.

[FPS96] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37–37, 1996.

[Goo21] Google Cloud. Cloud bigtable, 2021.

[GOR$^+$15] Abraham Gomez, Rafik Ouanouki, Anderson Ravanello, Alain April, and Alain Abran. Experimental validation as support in the migration from SQL databases to NoSQL databases. *Cloud Computing*, 162, 2015.

[Gov18] Government of Canada. Grade crossings inventory, 2018.

[GTH$^+$17] Douglas Gettman, Alan Toppen, Kelsey Hales, Alison Voss, Shane Engel, Dayana El Azhari, Cambridge Systematics, et al. Integrating emerging data sources into operational practice: Opportunities for integration of emerging data for traffic management and TMCs. Technical report, US Department of Transportation, 2017. Report FHWA-JPO-18-625.

[hea12] Public access to automated external defibrillators (AEDs). `https://www.heartandstroke.ca/-/media/pdf-files/canada/other/pad-eng-final.ashx`, 2012.

[HELD11] Jing Han, Haihong E, Guan Le, and Jian Du. Survey on NoSQL database. In *2011 6th international conference on pervasive computing and applications*, pages 363–366. IEEE, 2011.

[JMM$^+$18] Sladana Janković, Snezana Mladenović, Dusan Mladenović, Slavko Vesković, and Drazenko Glavić. Schema on read modeling approach as a basis of big data analytics integration in eis. *Enterprise Information Systems*, 12(8-9):1180–1201, 2018.

[Kan19] Saleh Kanani. A method to evaluate database management systems for big data: focus on spatial data, 2019.

[KL18] Henning Köhler and Sebastian Link. SQL schema design: foundations, normal forms, and normalization. *Information Systems*, 76:88–113, 2018.

[KLAA20] Samiya Khan, Xiufeng Liu, Syed Arshad Ali, and Mansaf Alam. Storage solutions for big data systems: A qualitative study and comparison. *ArXiv*, 2020. https://arxiv.org/abs/1904.11498.

[KP17] Douglas Kunda and Hazael Phiri. A comparative study of nosql and relational database. *Zambia ICT Journal*, 1(1):1–4, 2017.

[KSA16] Samiya Khan, Kashish A Shakil, and Mansaf Alam. Educational intelligence: Applying cloud-based big data analytics to the indian education sector. In *2016 2nd international conference on contemporary computing and informatics (IC3I)*, pages 29–34. IEEE, 2016.

[Lak18] Dany Laksono. Testing spatial data deliverance in SQL and NoSQL database using NodeJS fullstack web app. In *2018 4th International Conference on Science and Technology (ICST)*, pages 1–5. IEEE, 2018.

[Leu22] Carson Leung. An introduction to databases. lecture notes, 2022. University of Manitoba.

[MMR18] Suzanne McCarthy, Andrew McCarren, and Mark Roantree. An architecture and services for constructing data marts from online data sources.

Technical report, Dublin City University, 2018. Insight Technical Report Ref. 2018-1.

[MNL20] Diana Martinez-Mosquera, Rosa Navarrete, and Sergio Lujan-Mora. Modeling and management big data in databases—a systematic literature review. *Sustainability*, 12(2):634, 2020.

[Mon20] MongoDB. MongoDB atlas database as a service, 2020.

[MPS+19] Ayan Mukhopadhyay, Geoffrey Pettet, Chinmaya Samal, Abhishek Dubey, and Yevgeniy Vorobeychik. An online decision-theoretic pipeline for responder dispatch. In *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, pages 185–196, 2019.

[Nat21] National Fire Protection Association (NFPA). Standard for Fire Safety and Emergency Symbols, 2021.

[Neo20] Neo4j. Neo4j – the leader in graph technology. `https://neo4j.com/`, May 2020.

[PA20] Gabriela Picado-Aguilar and Jonathan Aguero-Valverde. Emergency response times and crash risk: an analysis framework for costa rica. *Journal of Transport & Health*, 16:100818, 2020.

[PJY+16] Peter Y. Park, Wook Rak Jung, Godfred Yeboah, Garreth Rempel, Dan Paulsen, and Dave Rumpel. First responders' response area and response time analysis with/without grade crossing monitoring system. *Fire Safety Journal*, 79:100–110, 2016.

[PPK$^+$19] Andrii Prokhorchenko, Larysa Parkhomenko, Andrii Kyman, Viacheslav Matsiuk, and Jelena Stepanova. Improvement of the technology of accelerated passage of low-capacity car traffic on the basis of scheduling of grouped trains of operational purpose. *Procedia Computer Science*, 149:86–94, 2019. ICTE in Transportation and Logistics 2018 (ICTE 2018).

[PPP$^+$19] Andrii Prokhorchenko, Artem Panchenko, Larysa Parkhomenko, G Nesterenko, Mykhailo Muzykin, G Prokhorchenko, and Alina Kolisnyk. Forecasting the estimated time of arrival for a cargo dispatch delivered by a freight train along a railway section. *Eastern-European Journal of Enterprise Technologies*, 3:30–38, 2019.

[PVSZ19] Romain Pinquié, Philippe Véron, Frédéric Segonds, and Thomas Zynda. A property graph data model for a context-aware design assistant. In Clement Fortin, Louis Rivest, Alain Bernard, and Abdelaziz Bouras, editors, *Product Lifecycle Management in the Digital Twin Era*, pages 181–190, Cham, 2019. Springer International Publishing.

[RCPR02] Ralph Renger, Adriana Cimetta, Sydney Pettygrove, and Seumas Rogan. Geographic information systems (GIS) as an evaluation tool. *American Journal of Evaluation*, 23(4):469–479, 2002.

[RS20] Noa Roy-Hubara and Arnon Sturm. Design methods for the new database era: a systematic literature review. *Software and Systems Modeling*, 19(2):297–312, Mar 2020.

[SÇ18] Michael Stonebraker and Uĝur Çetintemel. ”one size fits all” an idea whose time has come and gone. In *Making Databases Work: the Pragmatic Wisdom of Michael Stonebraker*, pages 441–462. ACM, 2018.

[SGB20] Marina V. Sokolova, Francisco J. Gómez, and Larisa N. Borisoglebskaya. Migration from an SQL to a hybrid SQL/NoSQL data model. *Journal of Management Analytics*, 7(1):1–11, 2020.

[SMA$^+$18] Michael Stonebraker, Samuel Madden, Daniel J. Abadi, Stavros Harizopoulos, Nabil Hachem, and Pat Helland. *The End of an Architectural Era: It's Time for a Complete Rewrite*, page 463–489. Association for Computing Machinery and Morgan & Claypool, 2018.

[SSPM16] Sugam Sharma, Ritu Shandilya, Srikanta Patnaik, and Ashok Mahapatra. Leading NoSQL models for handling big data: a brief review. *International Journal of Business Information Systems*, 22(1):1–25, 2016.

[TRBB15] Ciprian-Octavian Truica, Florin Radulescu, Alexandru Boicea, and Ion Bucur. Performance evaluation for crud operations in asynchronously replicated document oriented database. In *2015 20th International Conference on Control Systems and Computer Science*, pages 191–196, 2015.

[US 19] US Department of Transportation Federal Highway Administration (FHWA). Railroad-highway grade crossing handbook, 2019.

[VAOA22] Marlen Treviño Villalobos, Leonardo Víquez Acuña, Rocío Quirós Oviedo, and Oscar Víquez Acuña. Evaluation of the response time of a geoservice using a hybrid and distributed database. *Revista Colombiana de Computación*, 23(1):34–43, 2022.

[VCGF16] M. Villari, A. Celesti, M. Giacobbe, and M. Fazio. Enriched E-R model to design hybrid database for big data solutions. In *2016 IEEE Symposium on Computers and Communication (ISCC)*, pages 163–166, 2016.

[VKT18] Harsha Vyawahare, P.P. Karde, and V. M. Thakare. A hybrid database approach using graph and relational database. In *2018 International Conference on Research in Intelligent and Computing in Engineering (RICE)*, pages 1–4, 08 2018.

[vKvB17] Martin van Buuren, Geert Jan Kommer, Rob van der Mei, and Sandjai Bhulai. Ems call center models with and without function differentiation: A comparison. *Operations Research for Health Care*, 12:16–28, 2017.

[VML20] Diego Valdeolmillos, Yeray Mezquita, and Alberto R. Ludeiro. Sensing as a service: An architecture proposal for big data environments in smart cities. In Paulo Novais, Jaime Lloret, Pablo Chamoso, Davide Carneiro, Elena Navarro, and Sigeru Omatu, editors, *Ambient Intelligence – Software and Applications –,10th International Symposium on Ambient Intelligence*, pages 97–104, Cham, 2020. Springer International Publishing.

[VMZ$^+$10] Chad Vicknair, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen, and Dawn Wilkins. A comparison of a graph database and a relational database: A data provenance perspective. In *Proceedings of the*

48th Annual Southeast Regional Conference, ACM SE '10, New York, NY, USA, 2010. Association for Computing Machinery.

[WE18] Sujing Wang and Christoph F Eick. A data mining framework for environmental and geo-spatial data analysis. *International Journal of Data Science and Analytics*, 5(2):83–98, 2018.

[Zam20] Jo Zambito. Personal conversation, September 2020.

[ZBP+18] Igor Zečević, Petar Bjeljac, Branko Perišić, Stevan Stankovski, Danijel Venus, and Gordana Ostojić. Model driven development of hybrid databases using lightweight metamodel extensions. *Enterprise Information Systems*, 12(8-9):1221–1238, 2018.

[ZCW+16] X. Zheng, W. Chen, P. Wang, D. Shen, S. Chen, X. Wang, Q. Zhang, and L. Yang. Big data for social transportation. *IEEE Transactions on Intelligent Transportation Systems*, 17(3):620–630, 2016.

[ZJH+21] Liming Zhang, Shuo Jiang, Ke Huang, Yao Xiao, Linlin You, and Ming Cai. Knowledge graph-based network analysis on the elements of autonomous transportation system. In *2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 536–542. IEEE, 2021.