Leveraging Graph Databases for Multimodal Travel Data Integration and Analysis

Isaac Ramsaw Otchere

A Thesis

in

The Department

of

Gina Cody School of Engineering and Computer Science

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Quality Systems Engineering) at

Concordia University

Montréal, Québec, Canada

March 2026

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Isaac Ramsaw Otchere**

Entitled: **Leveraging Graph Databases for Multimodal Travel Data Integration and Analysis**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Quality Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Nizar Bouguila

_____ Examiner
Dr. Bilal Farooq

_____ Examiner
Dr. Manar Amayri

_____ Supervisor
Dr. Zachary Patterson

Approved by _____
Dr. Chun Wang, Chair
Department of Gina Cody School of Engineering and Computer Science

_____ 2026

_____
Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Leveraging Graph Databases for Multimodal Travel Data Integration and Analysis

Isaac Ramsaw Otchere

Recent advances in communication, mobility, and sensing technologies have enabled the collection of detailed travel data through smartphones. These advances offer the potential to capture not only traditional trip characteristics such as mode, purpose, and timing, but also real-time contextual feedback from travellers. However, most existing systems focus on post-trip, text-only feedback, overlooking live events that can significantly influence travel time, distance, and user experience. This research develops and applies a real-time, multimodal feedback framework integrated into the BDMobility app, a bespoke smartphone application designed to link trip records with video, audio, images, text, and documents. The framework records detailed contextual metadata covering who, when, how, what, and where feedback originates and associates it directly with ongoing trips. The framework's benchmark assessment demonstrated the significant computational advantage of the graph database (Neo4j) compared to the relational database (PostgreSQL and SQL) with increasing data sizes. Neo4j achieved up to a 90–95% reduction in query latency, maintaining response times below 150 ms at 50,000 records, whereas SQL exceeded 18,000 ms and PostgreSQL averaged around 9,500 ms under equivalent loads. In terms of indexing and scalability, Neo4j processed operations up to 85% faster, with latency stabilizing at under 10 ms, compared to PostgreSQL (60–120 ms) and SQL (80–140 ms). CPU utilization further reinforced this trend. PostgreSQL performed adequately during insertions and retrievals, however, its efficiency decreased significantly at peak utilization with CPU usage exceeding 300% with increasing record sizes. Neo4j consistently used 60–70% less CPU and 40% less memory than SQL during complex graph traversals. Neo4j's storage footprint decreased by 80% and demonstrated horizontal scalability, validating its suitability for real-time, multimodal, and feedback-enhanced urban transportation analytics. These findings

reiterate that the graph-based system offers enhanced computational performance and establishes a structural and energy-efficient basis for future human-centered, data-intensive transportation system capable of integrating structured and unstructured data at scale through capturing comprehensive travel experiences in large-scale, real-world mobility studies.

Acknowledgments

First and foremost, I would like to express my heartfelt gratitude to my supervisors for their guidance, support, and encouragement throughout my studies. Their insights and constructive feedback have been very resourceful and have provided me with great motivation to shape this work. I am also truly thankful to the Bridging Divides Program for providing me with the opportunity to collaborate with diverse academic and productivity-focused groups. The ISCTSC reviewers equally provided valuable insights that allowed me to apply my research in a real-world setting. I would especially like to thank all my colleagues for their unflinching support, constant reassurance, and the many insights they shared.

I learned so much through the collaboration among teams and professors, which really challenged me to be my best. To my friends and family, thank you for always being around. Your encouragement has always kept me going, helping me to overcome each challenge I have faced. Your support has been the foundation of everything I have achieved. Finally, I want to thank myself for not giving up, showing up every day, and believing that I could make it through. This journey has taught me more than I could have imagined, and I am grateful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Modern transportation systems generate data that spans two fundamentally different forms. On one side is *structured data*: timestamped GPS traces, mode classifications, origin-destination pairs, and schedule adherence metrics, all of which fit naturally into the rows and columns of a conventional database table. On the other side is *unstructured data*: photographs of overcrowded platforms, audio recordings of accessibility complaints, video clips of route diversions, and free-text narratives describing the lived experience of a journey. Individually, each form provides a partial picture of urban mobility; together, they offer the contextual richness that transportation planners need to understand not only *where* and *how* people travel, but *why* they make the choices they do and what they experience along the way. The central challenge addressed by this thesis is that the backend database architectures used in current transportation systems cannot efficiently store, integrate, and query these two forms of data together.

Historically, Intelligent Transportation Systems (ITS) and travel survey platforms have relied on Relational Database Management Systems (RDBMS) such as PostgreSQL, MySQL, and SQL Server to store trip records. RDBMS organise data into tables with fixed schemas, where each table contains rows (records) and columns (attributes). Relationships between entities, for example between a user and their trips, or between a trip and its feedback, are represented *implicitly* through foreign keys: column values in one table that reference the primary key of another. When a query requires data from multiple tables, the database engine must perform *join operations* that scan, match, and combine rows at execution time.

This tabular architecture works well for structured, predictable data. However, it encounters a fundamental limitation when the data becomes heterogeneous and densely interconnected. Consider a multimodal travel record that must link a commuter to a trip, the trip to its GPS geometry, the trip to one or more feedback entries, each feedback entry to its associated multimedia files (images, audio, video, documents), and each file to its metadata (timestamps, geotags, file type, size). In a relational schema, reconstructing this chain of associations requires four or more consecutive join operations. The computational cost of each join grows with the number of records in the participating tables, and when multiple joins are chained, these costs compound. As data volume increases from hundreds to hundreds of thousands of records, query latency in relational systems can degrade by orders of magnitude.

Furthermore, RDBMS offer limited native support for storing multimedia content alongside structured records. The two conventional approaches, converting files to Binary Large Objects (BLOBs) stored in table columns, or encoding media as base-64 strings, both introduce significant drawbacks. BLOB storage inflates the database, hinders backup and restoration operations, and demands costly server resources that a file system handles more efficiently. String encoding degrades data quality and is impractical for large files such as videos. These limitations mean that existing transport systems either exclude multimedia data entirely or manage it through ad-hoc external file systems that are disconnected from the trip records they describe, severing the semantic link between a commuter's experience and the structured data that contextualises it.

Graph databases offer a fundamentally different approach to data storage that directly addresses these limitations. A native graph database such as Neo4j represents data using *nodes* (entities), *relationships* (directed, typed connections between entities), and *properties* (key–value attributes on both nodes and relationships). Crucially, relationships are stored as first-class objects in the database: each node maintains direct physical pointers to every node it is connected to, a property known as *index-free adjacency*. This means that traversing from a user to their trips, from a trip to its feedback, and from feedback to its attached multimedia files follows a chain of direct pointers rather than executing join operations against indexed tables. The cost of such a traversal is proportional to the number of connections followed, not to the total size of the database.

For multimodal transportation data, this architecture offers three concrete advantages. First, the

heterogeneous chain of user, trip, geometry, feedback, and file entities is represented as a natural graph structure where each relationship carries its own type and properties, preserving the semantic context that relational foreign keys discard. Second, the graph query language Cypher expresses these traversal patterns using an intuitive visual syntax that directly mirrors the data structure, reducing query complexity compared with multi-table SQL joins. Third, because traversal cost does not depend on total dataset size, graph databases maintain low query latency even as the system scales to hundreds of thousands of records, a critical requirement for real-time feedback systems.

Previous studies that incorporated graph databases in transportation focused primarily on representing transport *networks* as graph structures, using structured data alone. (Czerepicki, 2016) demonstrated that Neo4j could perform traversal and connectivity queries more quickly than relational methods using tabular General Transit Feed Specification (GTFS) data. (Fortin, Trépanier, & Morency, 2016) embedded GTFS datasets into a graph-oriented framework to enhance transfer efficiency and road network accessibility. (I. Maduako, Cavalheri, & Wachowicz, 2018; I. D. Maduako, Wachowicz, & Hanson, 2019) introduced time-varying graph models to depict changing transit states over time, yet relied on structured sequential datasets. (H. Huang, Bucher, Kissling, Weibel, & Raubal, 2019; Wirawan, Riyanto, Nugraheni, & Yasmin, 2019) expanded graph structures to include multimodal routing and schema design, integrating public transit, carpooling, and walking into cohesive graph architectures, although they did not incorporate real-time user-generated feedback. (Shibanova, Stroganov, & Rudakov, 2021) developed formalised graph schemas and semantic hierarchies for data interoperability and (Park & Cheng, 2023) created a real-time multimodal graph framework integrating multiple transport modes across London. None of these studies addressed the integration of unstructured multimedia feedback with structured trip data within a graph database backend, which is the gap that this study fills.

The architectural limitation described above is compounded by a parallel gap on the data collection side. Urban commuting is inherently unpredictable: delays, overcrowding, route diversions, accidents, and accessibility challenges each influence not only the immediate journey but also a commuter's perception of the entire transportation system (Y. Huang, Li, Han, Xu, & Medeossi, 2024; Joshi, Agarwal, Kumar, Dogra, & Nandan, 2024; Lemonde, Arsenio, & Henriques, 2021). Although transit agencies collect data on schedules, routes, and ridership, they rarely capture the

lived, in-the-moment experiences that shape daily travel decisions (Berggren, 2021). The absence of such context creates a gap between operational metrics and the realities faced by commuters (Clifton & Muhs, 2012).

Traditional travel surveys predominantly rely on post-trip, text-based recollections, which condense diverse experiences into simplified summaries and depend heavily on participants' memory, subjective interpretation, and willingness to report events retrospectively (C. Chen, Gong, Lawson, & Bialostozky, 2010; Dabiri & Heaslip, 2018). Although many travellers informally document their journeys using photos, videos, audio recordings, or notes, current transportation infrastructure cannot process or integrate this real-time multimodal feedback with trip records (Harding, 2019). Existing mobility applications focus on GPS logging (Stopher, Daigler, & Griffith, 2018), mode detection (Prelipcean, Gidófalvi, & Susilo, 2016), purpose detection, and passive sensing (Hosseini, 2025), which excel at quantifying movement but cannot incorporate the social, emotional, and accessibility-related dimensions of travel. Continuous GPS logging itself suffers from rapid battery depletion, reduced accuracy in urban canyons, data loss during signal interruptions, and increased privacy concerns (C. Chen et al., 2010; Dabiri & Heaslip, 2018), frequently resulting in incomplete or inconsistent trip records in densely populated metropolitan areas.

This methodological limitation prevents transport systems from fully understanding disruptions, accessibility barriers, and behavioural responses, especially for marginalised groups. Without integrated real-time multimodal feedback and a backend architecture capable of managing it, opportunities for responsive planning, targeted interventions, and inclusive policy development remain unrealised.

This thesis addresses both the backend architectural deficiency and the data collection gap through two integrated contributions. On the data collection side, the BDMobility travel survey application, developed through the Bridging Divides (BD) Program (Bridging Divides, 2025), was enhanced to combine structured trip data with real-time commuter observations in multiple multimedia formats: text, audio, video, images, and documents. The application was deployed and tested in Greater Montreal and Toronto, leveraging the diverse multicultural demographics and complex transit systems of these metropolitan areas. The system adjusts dynamically to users' travel patterns and socio-demographic characteristics, allowing for the collection of revealed preference data that

enhances stated preference modelling (Cabalquinto & Hutchins, 2020; Shaheen, 2016).

The application incorporates the MotionTag Software Development Kit (SDK), a mobility-sensing framework that optimises continuous trip tracking while minimising participant burden. MotionTag enhances data reliability through adaptive sampling, intelligently switching between GPS, Wi-Fi, cell-tower triangulation, and onboard sensors such as accelerometers and gyroscopes. This hybrid sensing approach maintains high spatial and temporal precision without excessive battery drain and mitigates recall and transcription biases by automating data capture in real time (MotionTag GmbH, 2025).

On the backend architecture side, the RDBMS (PostgreSQL) used in the previous version of the system was replaced with Neo4j, a native graph database, enabling the integration of structured trip data and unstructured multimedia feedback within a single, semantically rich data store. The graph query language Cypher replaces cumbersome SQL join statements with intuitive pattern-matching queries that naturally express the relationships among users, trips, feedback, and files. The purpose of this architectural transition is to advance transportation analytics beyond GPS traces, enabling deeper behavioural insights and supporting the development of more responsive, resilient, and inclusive transportation systems.

Guided by the identified research gaps, this study establishes clear objectives to deepen the understanding of commuter behaviour, optimise transport data architectures, and integrate unstructured multimodal feedback with structured trip data in the following terms:

(1) What structural and data requirements are necessary to maintain multimodal travel databases?

(2) How can commuter feedback be enriched using live multimodal inputs?

(3) In what contexts are commuters most likely to record their travel experiences, and which multimedia data are preferred and often reported in such contexts?

The aforementioned objectives, in turn, underpin the study's contributions toward more adaptive and human-centered mobility intelligence systems.

1.1 Contribution

This study makes two main contributions. First, this study, **optimized the BD Mobility Travel Survey Application to record real-time trip feedback** by offering a new dimension to transport data modelling to capture, store, and manipulate unstructured, multimodal commuter feedback alongside structured trip data on sequences of movement, locations, and halts across multiple travel modes through the BD Mobility app. The system extends conventional data collection by recording commuters' lived travel experiences in diverse multimedia formats such as video, audio, images, documents, and textual narratives. A novel enhancement within the application allows it to automatically detect unusual stops, defined as halts exceeding three minutes, and to trigger real-time feedback prompts irrespective of whether the user is travelling by public transport, private vehicle, or on foot. This contextual feedback mechanism captures the situational nature under thematic classification of each event, embedding precise timestamps, locations, and narrative annotations with files that enrich the interpretation of commuter experiences within the travel timeline.

More significantly, the study identified a critical gap in existing travel behaviour studies, indicating traditional surveys effectively capture the locations and modes of travel but fail to address the underlying reasons for travel behaviour. The findings demonstrated that commuters' mode choices are significantly influenced by their lived travel experiences, highlighting the necessity of documenting experiential and contextual data to better understand travel behaviour dynamics. The early development of this research was presented at the 13th International Steering Committee for Transport Survey Conference (ISCTSC 2025).

Second, to address the critical gap identified in the first study, the subsequent study focused on **Real-Time Multimodal Travel Feedback Processing with a Graph Database for the Travel Survey App.** This study provides a cohesive methodological and architectural framework for the storage, organization, and real-time manipulation of multimodal travel data, with particular emphasis on preserving semantic relationships among journeys, events, and user-generated context. The study conducts an experimental comparison of relational databases (PostgreSQL and MySQL) and a native graph database (Neo4j) to evaluate how different storage models support multimodal data manipulation workflows at scale, using performance metrics that capture ingestion efficiency,

retrieval latency, system resource utilization, and relational linkage across heterogeneous data types.

The contribution therefore lies in demonstrating how unstructured multimodal feedback can be systematically stored, manipulated, and queried alongside structured trip records, enabling richer contextual representations of mobility data. The novelty is not in behavioural inference, but in the design and evaluation of a backend system capable of efficiently managing and manipulating multimodal data at scale. Building on this architectural foundation, the research extends a travel survey application to support real-time multimodal feedback capture and illustrates how graph-based data structures more effectively represent the relational complexity inherent in human mobility. Collectively, the findings highlight the limitations of conventional transportation data systems and establish a scalable, feedback-enhanced framework that bridges structured mobility data with unstructured experiential input, thereby supporting more context-aware, responsive, and human-centred travel survey and transportation planning systems.

1.2 Thesis Organization

The remainder of the thesis is structured as follows: Chapter 2 offers a comprehensive overview of the existing literature on smartphone-enabled mobility collection, architectural and design challenges in transportation systems, and how the challenges are addressed by reinventing multimodal data management with graph databases to collect trip feedback. Chapter 3 introduces a comparative study of the architecture and design constraints in recording multimodal data as feedback on travels in our proposed transportation system. Chapter 4 presents analysis on insights inferred from user experiences based on observed trip characteristics. Chapter 5 concludes the thesis and summarizes our contributions.

Chapter 2

Literature Review

Travel data is crucial to understanding travel patterns, behaviours, and preferences, as it provides the empirical basis for understanding how people move and supports data-driven decisions to improve transportation systems and policy planning based on forecasted demands on travel models. Travel surveys are instrumental in collecting individual or household travel data to understand activity behaviours and measure daily travels. Typical travel surveys are specifically modelled to collect data on individual demographics, household and travel characteristics, for instance, travel modes, purpose, duration, distance, etc. In the past, travel surveys were solely known to depend on the participants' memory recall of completed trips, especially with state-of-practice methods (e.g., computer-assisted telephone interviews, mail, or web-based travel diaries). This approach has been known to suffer from inconsistent and inaccurate data (Servizi, Pereira, Anderson, & Nielsen, 2021). Modern technologies introduced GPS loggers and location-aware smartphones to automatically log locations and activities continuously, providing bases to validate recall-based travel diaries (Graells-Garrido, Opitz, Rowe, & Arriagada, 2023; Harding, Faghih Imani, Srikukenthiran, & Miller, 2021).

Reviewing existing studies in this domain, it is observed that the expressions of commuters during events, typically disruptions and others that deviate from the daily perceived travel events are not recorded. This remains so due to the architectural and design limitations inherent in existing transportation systems. In recent times travellers often record travel experiences in videos, audios, images, documents, and notes in digital travel journals. This data is ideal to enrich travel data by introducing more context and emotions into unplanned events that affect travels, including

prolonged waits, accidents, discomfort, and others (Singleton, 2020). This ensures transportation systems adapt with timely responses to real-time events (Harding, 2019; Servizi et al., 2021).

In this section, an overview of early studies into enhanced location-aware smartphone applications, their limitations, how the identified gap is addressed in our approach, and how the proposed system was transformed to record multimodal data along with trip data to overcome the technical and architectural deficiencies in existing transportation systems using a graph database.

2.1 Location-aware Transportation Applications

Location-aware mobile applications represent a notable result of research on smartphone-based mobility; nonetheless, their influence on travel behaviour and mobility systems is more complex than their technological advancement suggests. Applications such as Google Maps, Transit, and CityMapper integrate static transport schedules with live GPS data to facilitate dynamic routing, multimodal journey planning, and real-time updates. Prior research indicates that these factors can reduce uncertainty in daily commuting, improve user satisfaction, and promote shifts in transport modes towards more sustainable options (Kaasinen, 2003; Shaheen, 2016). These applications employ adaptive algorithms to customise travel recommendations, adjusting routes and modes of transport based on commuter preferences and historical data (Signer Del Cid, 2024). Research, such as that presented in (Kamargiannis et al., 2023) and (Patterson, Fitzsimmons, Jackson, & Mukai, 2019), advances the field by integrating passive GPS tracking with stated preference surveys, facilitating the correlation of objective mobility data with subjective behavioural insights in near real time.

Despite the recognition of these technical capabilities in the literature, the broader implications remain insufficiently explored and inconsistently addressed. A significant number of studies adopt a functionalist perspective, focusing on efficiency, route optimization, and system accuracy, while neglecting the behavioural, emotional, and social dimensions of mobility. The premise that increased information results in rational decision-making neglects the ways in which travellers interpret and emotionally respond to disruptions, crowding, or perceived risk factors that substantially influence

actual behaviour (Carrel, Halvorsen, & Walker, 2013; Ibrahim & Borhan, 2020). Moreover, ongoing GPS logging poses both technical and ethical challenges. High energy consumption and data gaps due to signal interference undermine data reliability, while persistent privacy concerns deter participation and bias samples towards users with higher technological confidence (Hosseini, 2025; Prelipcean et al., 2016).

A critical perspective suggests that these applications emphasize measurable metrics of mobility, including routes, distances, and times, while neglecting the qualitative dimensions of the travel experience. The gathered data predominantly emphasize movement, lacking sufficient consideration of the affective and contextual dimensions of urban travel, such as stress during delays, discomfort in overcrowded vehicles, and accessibility issues for vulnerable populations. This omission reinforces a narrow definition of "smart" mobility that equates optimization with enhancement, overlooking the essential social and emotional dimensions of the mobility experience. This gap is addressed by integrating multimodal data sources, including self-reported wellbeing indicators, environmental context, and situational feedback, into smartphone-based systems, which essentially improves travel feedback with contextual data. This synthesis would shift the focus of the field from technological enhancement to human-centered mobility analytics, aligning with recent calls in transport psychology and behavioural informatics to balance efficiency with equity and well-being (Fonseca et al., 2022; Yu, Cheung, Lau, & Woo, 2017; H. Zhang, Zhang, Liu, & Zhang, 2023).

2.2 Commuter Context and Experience Data

Researchers have increasingly recognized that commuting is not only a spatial and temporal phenomenon but also a deeply contextual and emotional one (Gatersleben & Uzzell, 2007). Delays, overcrowding, and accessibility challenges significantly influence commuter satisfaction and perceptions of system reliability (Lemonde et al., 2021).

Building on the foundational studies explored earlier, recent studies have progressively incorporated psychological and physiological viewpoints to more thoroughly understand the embodied and contextual aspects of commuting. (X. Zhang & Ma, 2024) established that prolonged commuting

times are associated with heightened stress and reduced mood, especially in congested settings, indicating that spatial and temporal pressures influence emotional well-being. (Jain & Handy, 2025) shown that commute well-being substantially affects total life satisfaction, with the mode of commuting (e.g., active versus motorized) modulating this association. Their findings suggest that commuting produces emotional spillover effects that transcend the travel itself. Recent advancements in sensor-based and experience-sampling research have enhanced the empirical comprehension of commuting as an emotional and context-dependent activity.

(Bosch, Luther, & Ihme, 2025) used physiological data, including electrocardiogram (ECG) scans, to delineate real-time stress and satisfaction levels in public transport riders. Their findings shown that emotional reactions vary dynamically over various route segments, affected by environmental stimuli (e.g., crowding and delays) and social interactions. In addition, (Li et al., 2025) examined commuting stress in healthcare professionals and found that emotional tiredness was significantly associated with the duration, unpredictability, and form of commuting, so identifying commuting as a psychosocial stressor with consequences for occupational health.

(Yang, He, He, & Peng, 2025) further discussed this viewpoint by creating a conceptual model that associates commuting with the depletion of physical and psychological energy, contending that urban factors such as density, noise, and congestion moderate the emotional effects of everyday travel. Similarly, research in transport geography, exemplified by (Sandberg, Hurmerinta, Helminen, Leino, & Vasankari, 2024), has examined how alterations in infrastructure, such as the implementation of new tramways, cultivate ”emotional commuting,” whereby commuters develop emotional bonds and ingrained habits associated with changing mobility contexts. The emotional aspect of commuting has been examined concerning mental health and social equality.

(Guan, Xue, Zhang, Chang, & Liu, 2025) indicated that extended commuting durations for rural students adversely impacted mental health outcomes, highlighting the disparate emotional strain of mobility among demographic groups. These findings correspond with extensive research trends highlighting affective mobility, wherein emotions, identity, and social context converge to influence daily travel experiences. These works collectively signify a paradigm change in transport research, transitioning from perceiving commuting as a mechanical process of movement across place and time to recognizing it as a contextually situated, emotionally charged human practice.

The emotional geographies of commuting are now pivotal in mobility research, underscoring the necessity for multimodal and multidimensional models that incorporate geographical, temporal, contextual, and affective data. These insights are crucial for building transportation systems that maximize efficiency and improve commuter well-being and quality of life.

Traditional travel surveys, however, are retrospective and text-based, often simplifying complex lived experiences into textual summaries. Such methods overlook real-time contextual details that are evident in situations such as discomfort, stress, or unsafe conditions (Harding, 2019). Prior studies have attempted to integrate smartphone surveys with experience sampling, asking commuters to report emotions during trips (Carrel, Sengupta, & Walker, 2017). Others explore social media and geographic information volunteered as proxies for real-time commuter feedback (Barns, 2020).

These studies collectively signify a paradigm change in transport research, transitioning from perceiving commuting as a mechanical process of movement across place and time to recognizing it as a contextually situated, emotionally charged human practice. The emotional geographies of commuting are now pivotal in mobility research, underscoring the necessity for multimodal and multidimensional models that incorporate geographical, temporal, contextual, and affective data. These insights are crucial for building transportation systems that maximize efficiency and improve commuter well-being and quality of life.

While these approaches enrich contextual data, they remain fragmented and difficult to integrate into structured transport databases. Hence, there is a need to understand the complexity and architectural and design restrictions inherent in transportation systems that inhibit the possibility to integrate real-time contextual trip feedback with recorded trip data.

2.3 Architecture of Existing Transportation Systems

The evolution of transportation networks demonstrates an ongoing interaction among technology advancement, urban expansion, and societal demands. Initial systems depended on human and animal labor, facilitated by primitive road networks, canals, and wagons. These infrastructures facilitated the spread of trade and settlement yet were hindered by geographical limitations and inefficiencies (Grubler, 1990). The industrial revolution signified the initial significant transition,

characterized by the advent of railroads and mechanized transport, which established standardized timetables and facilitated long-distance connectivity essential for industrial economies (Grubler & Nakicenovic, 1991). This was accompanied by urban tram systems, canals, and port developments, which collectively established the groundwork for mass transit. The 20th century witnessed the preeminence of automobility and aircraft, transforming mobility on both urban and global levels. The advent of highways and private vehicle ownership provided individual autonomy, however simultaneously resulted in congestion, pollution, and unequal accessibility (Dempsey, 2003). Consequently, rapid transit systems (subways and metros) have become essential for high-density cities and urban areas (Maslich & Voronova, 2024).

Modern transportation systems are increasingly conceptualized as systems-of-systems (SoS), integrating multiple independent subsystems such as road infrastructure, public transit networks, communication systems, and data management platforms. (DeLaurentis, 2005) argues that such architectures face inherent complexity because they must align competing objectives, such as efficiency, safety, sustainability, and accessibility, across heterogeneous components. This interconnectedness increases systemic fragility: failure in one component (e.g., communication delays) can cascade across the system. Intelligent Transportation Systems (ITS) represent the backbone of modern mobility solutions, integrating sensing, communication, and control technologies to optimize traffic flow, improve safety, and support sustainable urban development (Xu, Lin, & Yu, 2017). Typical ITS architectures consist of physical layers (sensors, vehicles, roadside units), communication layers (cellular, Wi-Fi), and data processing layers (centralized or cloud-based platforms). These multi-layered designs allow real-time traffic monitoring, automated incident detection, and multimodal travel information services (Elkosantini & Darmoul, 2013). Despite advancements, current ITS architectures face major challenges around fragmentation across agencies, which leads to incomplete integration of multimodal data (Macioszek, 2014) often described as data silos. Also, there exist scalability constraints in centralized cloud-based systems through struggles with latency and bandwidth demands in dense urban contexts (Nasim & Kassler, 2012). (Borzacchiello, Torrieri, & Nijkamp, 2009) also identified interoperability issues, as many systems lack compliance with standards, limiting cross-city or international integration. Public transit systems, on the other hand,

often rely on layered architectures comprising operational control centers, vehicle tracking modules, passenger information services, and fare collection systems. These architectures allow transit agencies to monitor fleets, estimate arrival times, and manage schedules (Sharma & Awasthi, 2022). Public transit systems often primarily optimize supply-side operations but underrepresent demand-side experiences, such as comfort, safety, and accessibility. Also, the architecture of public transit systems often excludes real-time commuter-generated data, limiting responsiveness to disruptions (Elkosantini & Darmoul, 2013).

Recent research explores distributed and hierarchical ITS architectures leveraging cloud computing, edge devices, and 5G-enabled software-defined networks (SDNs) to enhance flexibility and reduce latency (Din, Paul, & Rehman, 2019). Such architectures aim to address the scalability bottlenecks of centralized systems by distributing computation closer to users and vehicles. Similarly, communication architectures using Named Data Networking (NDN) improve efficiency in data dissemination for vehicle-to-infrastructure interactions (Sousa et al., 2017). While technically advanced, these systems still largely emphasize traffic efficiency and safety, with minimal integration of qualitative commuter experience data, and their implementation is costly and highly uneven across regions, creating disparities between "smart" and under-resourced cities (Picone, Busanelli, Amoretti, Zanichelli, & Ferrari, 2015).

The Intelligent Transportation Systems (ITS) have developed tremendously over the years to become an integrated system of systems with technological innovations, policy, and social changes focused on enhancing mobility efficiency, safety, and sustainability. The gradual development of ITS architectures and their corresponding innovations from mechanized systems to integrated information technology systems are summarized in figure 2.1. This figure illustrates the transformation of transportation networks through successive phases, ranging from initial infrastructure-based systems to modern data-driven and AI-enabled architectures. Each entry delineates the temporal framework, geographical context, technological or conceptual progress, resultant effects on transportation systems, and the principal researchers or studies that facilitated these advancements. The transitions in development is organized chronologically, illustrating how emerging technologies and infrastructures such as cloud computing, the Internet of Things (IoT), 5G, and artificial intelligence (AI) have collectively influenced the contemporary ITS landscape, addressing challenges related to

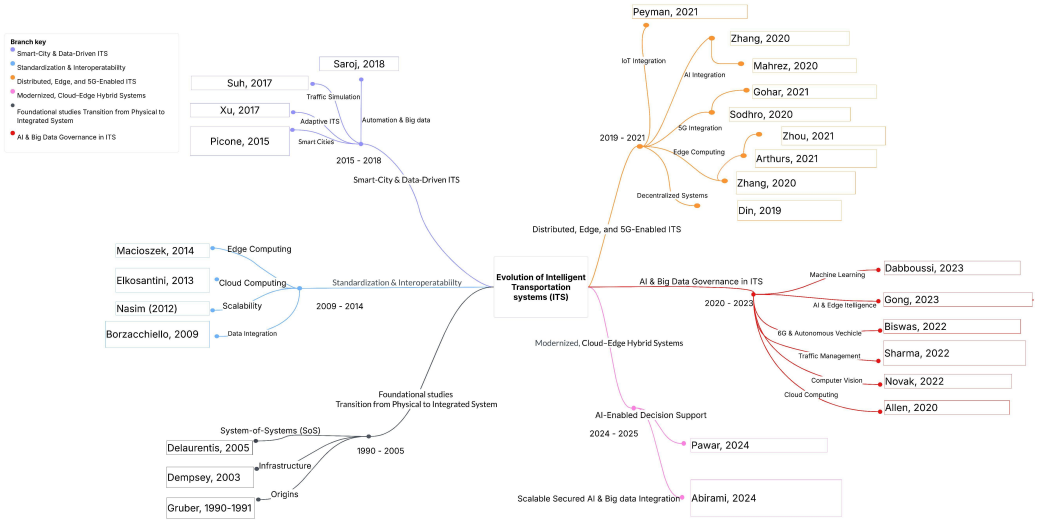

Figure 2.1: Merged Literature on Intelligent Transportation Systems (ITS) Architectures and Developments (1990–2025)

scalability, interoperability, and governance. This approach offers a thorough examination of the historical foundations and future directions of ITS research and implementation globally.

The figure 2.1 and table A.1 reveal notable patterns in the evolution of ITS reflecting the historical trajectories in early studies from the 1990s to 2000s which emphasized standardization, infrastructure development, and system interoperability, laying the groundwork for integrated transportation frameworks (Dempsey, 2003; Grubler & Nakicenovic, 1991). In the mid-2010s, emerging studies indicated a transition to data-centric and cloud-enabled Intelligent Transportation Systems (ITS) architectures (Elkosantini & Darmoul, 2013; Nasim & Kassler, 2012), signifying a shift from physical and mechanized systems to cyber-physical systems. The emergence of edge computing and 5G in the late 2010s and early 2020s (Din et al., 2019; J. Zhang & Letaief, 2020) marked a notable advancement, enabling real-time communication and decentralized control mechanisms that addressed latency and scalability issues inherent in earlier centralised models. Recent studies by (Abirami, Pethuraj, Uthayakumar, & Chitra, 2024; Gong, Zhu, Yu, & Tang, 2023; Pawar et al., 2024) highlighted a growing emphasis on AI-driven decision-making, blockchain governance, and sustainability-focused Intelligent Transportation Systems (ITS), demonstrating an interdisciplinary convergence of computer science, transportation engineering, and policy. The developments signify

a shift from infrastructure-driven to intelligence-driven transportation systems, marked by connectivity, automation, and ethical data governance that are influencing the future of global mobility.

2.4 Graph database usefulness in transportation systems

Graph database management systems (GDBMS) are a class of NoSQL databases that natively represent data as nodes (entities) and edges (relationships), making them well-suited for complex, interconnected systems such as transportation networks. Unlike relational databases, which rely on tabular schemas and joins, graph databases are optimized for relationship-centric queries such as pathfinding, transfer detection, and network traversal (J. Chen, Song, Zhao, & Li, 2020). Several graph database implementations have been applied in transport and smart city contexts. Neo4j, OrientDB, and TigerGraph are among the most widely studied. OrientDB has been noted for speed in certain multimedia sensor applications (Küçükkeçeci & Yazıcı, 2017), while TigerGraph is optimized for distributed, high-performance big data scenarios. However, Neo4j stands out for its maturity, ecosystem, and support for transportation-related graph queries (Santos López, 2015). Research comparing graph databases and relational databases shows that RDBMS architectures face scaling and flexibility issues when applied to transportation data. For instance, mapping public transportation networks onto RDBMS requires multiple joins across schedules, routes, and stop tables, which degrades performance in large, dynamic networks (Serin, Mete, Gül, & Celik, 2018). Graph databases avoid this overhead by natively encoding relationships, making path traversal queries (e.g., shortest route, alternative path during disruption) significantly faster (J. Chen et al., 2020). When processing multimedia data, relational databases are constrained by rigid schemas and poor support for semi-structured or unstructured data. In contrast, graph databases flexibly link unstructured commuter inputs (audio, video, images, documents, and text) to structured trip nodes, supporting hybrid analyses that combine movement data with experiential feedback (Küçükkeçeci & Yazıcı, 2018). Among available GDBMS, Neo4j has emerged as the de facto standard in both academic and industrial applications for transportation, smart cities, and multimedia systems, including social media platforms (Tjortjis, 2021). Neo4j is a superior choice based on the outstanding advantages it offers through:

(1) Neo4j is optimized for relationship-heavy queries with Cypher, a declarative query language designed for graph traversal, outperforming SQL joins for transportation routing and disruption analysis (Kousis, 2023).

(2) Neo4j allows multimedia data (e.g., commuter-uploaded images or videos) to be linked directly to nodes (e.g., trips, stops), enabling multimodal feedback systems that RDBMS cannot easily replicate (Mystakidis, 2023).

(3) Neo4j has been successfully deployed in smart city applications, including transportation, energy networks, and multimodal integration, demonstrating its scalability and adaptability (Serin et al., 2018).

(4) Neo4j's widespread adoption ensures robust documentation, tooling, and integration with machine learning pipelines, which further strengthens its role in mobility research (Santos López, 2015).

Data management technologies in transportation systems have been shaped by the swift advancement of computers, data modelling, and the emergence of new transportation networks. The transition from early relational database management systems (RDBMS), designed for structured, tabular data, to contemporary graph database management systems (GDBMS), suited for intricate, interconnected networks, exemplifies the growing demand for models that can depict dynamic, multimodal, and real-time transport environments. Table 2.1 delineates a comparison of essential technical and functional aspects of RDBMS and GDBMS within the framework of transport network modelling and their application in intelligent transportation systems (ITS). This comparison includes technological, methodological, and application-oriented viewpoints from recent studies, highlighting how developments in data architecture have revolutionized analytical capabilities, scalability, and decision support in mobility management.

Table 2.1: Comparison of Relational and Graph Databases in Transport Network Modeling and Management

Aspect	Relational Databases (RDBMS)	Graph Databases (GDBMS)	Representative Studies / Sources
Data Structure and Schema Flexibility	Structured in fixed tables (rows and columns) with rigid schema; suitable for transactional datasets.	Node–edge–property model offering flexible schema and dynamic relationship creation.	(Hrnčíř & Jakob, 2013; Smarzaro, Davis, & Quintanilha, 2021); (Neo4j, 2024).
Query and Traversal Efficiency	Performs well for aggregate or relational queries (e.g., SQL joins), but recursive pathfinding degrades performance.	Optimised for graph traversal and relationship queries using pathfinding algorithms (Dijkstra, A*).	(Microsoft Corporation, 2025); (Neo4j, 2024).
Temporal and Spatio-Temporal Data Support	Temporal modeling often relies on time-expanded tables or versioned schema extensions.	Dynamic or temporal graphs can model evolving edges and time-dependent relationships directly.	(Bellocchi & Gerolimi-nis, 2020; J. Chen et al., 2020; I. Maduako et al., 2018; Xue, Tan, Ma, Li, & Wang, 2025)

(continued on next page)

(continued from previous page)

Aspect	Relational Databases (RDBMS)	Graph Databases (GDBMS)	Representative Studies / Sources
Integration and Multi-Source Data Handling	Integration requires complex ETL and schema matching across relational sources.	Naturally accommodates heterogeneous data (CSV, JSON, sensor, and IoT feeds) using labeled property models.	(Smarzaro et al., 2021); (California Department of Transportation, 2024).
Use Cases in ITS / Transport	Efficient for managing static datasets such as schedules, timetables, and trip logs using PostGIS or PostgreSQL.	Ideal for multimodal network modeling, real-time routing, and connectivity analysis in smart cities.	(X. Chen & Tong, 2025; Czerepicki, 2016; H. Huang et al., 2019; I. D. Maduako et al., 2019).
Scalability and Performance	Scales vertically; schema changes require downtime and migrations.	Horizontally scalable; supports distributed graph processing and streaming integration.	(Fortin et al., 2016; I. D. Maduako et al., 2019; Neo4j, 2024).

(continued on next page)

(continued from previous page)

Aspect	Relational Databases (RDBMS)	Graph Databases (GDBMS)	Representative Studies / Sources
Advantages	Strong data consistency, mature SQL ecosystems, ACID-compliant transactions.	Superior performance in connected data, intuitive for network visualization and analytics.	(Hrnčíř & Jakob, 2013; Neo4j, 2024).
Limitations	Weak performance in recursive or path-based queries; inflexible for evolving schemas.	High memory footprint for large graphs; fewer mature query optimizers and analytic tools.	(Chondrogiannis, Kalogeraki, & Gunopulos, 2016; H. Huang et al., 2019; Neo4j, 2024; Schema App, 2023).
Emerging Trends (Post-2025)	Hybrid SQL + NoSQL architectures with embedded graph extensions (e.g., PGQ, Oracle Graph).	Graph neural networks (GNNs) and AI-driven dynamic graph updates for predictive mobility and routing.	(Xue et al., 2025; Q. Zhang, Ma, Zhang, & Jenelius, 2025).

This investigation revealed a clear shift from structured, schema-based relational models to adaptable, connection-oriented graph structures, which accurately depict the intricacies of modern transportation networks. Relational databases are crucial for the management of static and transactional data, such as schedules, fare records, and ridership logs. In contrast, graph databases excel at modelling dynamic, multimodal, and relational data, where connectedness and network traversal are essential. This shift corresponds with the comprehensive evolution of transportation systems, where real-time analytics, sensor integration, and multimodal routing increasingly rely on graph-based data structures. Recent research by (I. D. Maduako et al., 2019) and (Fortin et al., 2016) indicates that graph-based methodologies improve the efficacy of transit network analysis and multimodal trip planning. The integration of AI and graph neural networks marks a new phase in this evolution, providing predictive capabilities aligned with emerging trends in smart city mobility and autonomous transportation systems.

In this study the graph databases considered are compared against each other to reveal their strengths and weaknesses to signify the why Neo4j remains preferred over OrientDB in integrating structured and unstructed data. The comparison is highlighted in the table 2.2 below.

2.5 Summary and research gaps

In a nutshell, the review of existing transportation system architectures reveals a persistent tension between structured data efficiency and contextual depth. In the context of Intelligent Transportation Systems (ITS), the interconnected components of human mobility include travel patterns, traveller behaviour, and perceptions. Travel patterns denote the identifiable sequences of movement that Intelligent Transportation Systems (ITS) platforms monitor and analyze through location and sensor data. Traveller behaviour encompasses the decision-making processes that influence patterns, including the selection of routes or modes of transport, whereas perceptions relate to individuals' subjective assessments of their travel experiences. ITS frameworks integrate these dimensions through the use of sensor-based data to quantify patterns, behavioural analytics to infer decisions, and participatory or feedback mechanisms to capture perceptions. These components collectively

Table 2.2: Comparison of Neo4j and OrientDB for Multimodal Feedback System

Criteria	**Neo4j**	**OrientDB**
Data Model	Native property graph: highly optimized for relationships and traversal (J. Chen et al., 2020).	Multi-model (graph + document + key-value + object); versatile but less optimized for graph workloads (Fernandes & Bernardino, 2018).
Query Language	Cypher query language. Declarative and graph-optimized.	SQL-like query language with support for Gremlin and pattern matching.
Performance	Superior performance for relationship-intensive queries and transport networks (Virić & Pantelić, 2023).	Slower graph traversals compared to Neo4j; performance trade-offs due to the multi-model design.
Multimedia Handling	Stores metadata in graph links external multimedia files (audio, video, images, documents) efficiently (Vyawahare, 2024).	Can embed documents natively but lacks optimized multimedia-handling pipelines.
Ecosystem & Support	Largest community and rich ecosystem (Graph Data Science, visualization tools, connectors).	Smaller community, fewer advanced libraries, and limited integrations (Vereycken, 2016).
Scalability	Scales effectively for transport and feedback systems with millions of relationships.	Scales horizontally; however, the hybrid design complicates the optimization of graph-heavy queries.
Suitability for Research	Strong fit for analyzing when, why, and how feedback occurs across multimodal journeys.	Flexible for heterogeneous data but less efficient for narrative and relational analyses.

enable a thorough, human-centered approach to mobility intelligence, aligning operational performance with user experience.

Building on this, traditional ITS built largely on relational database infrastructures, have excelled in processing structured, operational data such as schedules, traffic flows, and GPS logs. However, these architectures exhibit fundamental constraints: (i) difficulty in scaling to model the complex, dynamic, and relational nature of urban transportation networks; (ii) reliance on siloed datasets that separate commuter experiences from system performance; and (iii) inability to accommodate multimodal, unstructured inputs such as audio, video, or images that reflect real commuter realities (J. Chen et al., 2020; Serin et al., 2018). Recent advances in graph databases offer a transformative response to these limitations. By representing transportation networks as nodes and edges, graph databases such as Neo4j overcome the join-heavy inefficiencies of RDBMS and support native pathfinding, disruption modelling, and multimodal integration (Kousis, 2023). Neo4j, in particular, has proven effective in smart city and transportation contexts, enabling the integration of heterogeneous data sources, from structured mobility logs to unstructured multimedia feedback, within the

same analytical framework (Mystakidis, 2023). Parallel research on smartphone-enabled mobility applications shows that location-aware technologies have revolutionized data collection by enabling continuous trajectory logging and passive sensing. Yet, these approaches remain technically and contextually constrained. Continuous GPS logging is limited by battery consumption, privacy risks, and signal inaccuracy in dense urban areas, leading to incomplete or biased datasets (Prelipcean et al., 2016). Moreover, smartphone-based systems focus heavily on movement metrics, such as travel time or mode detection, while overlooking the qualitative, emotional, and accessibility-related dimensions of travel experiences.

This thesis is positioned at the intersection of these two trajectories, integrating structured data on trips with multimodal unstructured feedback data within a graph database and an enhanced smartphone-enabled sensing application that records real-time trip feedback, by proposing the BDMobility application as a novel contribution. Developed under the Bridging Divides initiative, BDMobility integrates the MotionTag SDK for energy-efficient, multimodal trip and purpose detection while addressing privacy, connectivity, and accuracy concerns inherent in location-aware applications. Crucially, it extends beyond structured trip logs by enabling commuters to contribute real-time, multimodal feedback in audio recordings, video clips, images, text annotations, and documents, which are linked directly to their trip data.

Capitalizing on the limitations of both relational architectures and smartphone-only sensing, the proposed system advances transportation research in three distinct ways: 1. It creates a multimodal data-based travel database that unifies structured and unstructured commuter data. 2. It demonstrates the practical superiority of graph databases for modelling dynamic, user-centered transportation networks. 3. It embeds commuter experiences into the architecture of mobility systems, bridging the gap between operational performance metrics and lived realities.

In summary, the literature confirms the deficiencies in the architectural and methodological capacity of existing systems to represent the complexity of urban commuting in its full multimodal and experiential contexts. This study addresses that gap by introducing a graph-based, human-centered application and database architecture through the BDMobility application, thereby justifying the necessity and timeliness of this research.

Chapter 3

Methodology

The primary aim of this research is to evaluate different backend database architectures specifically PostgreSQL (relational), SQL Server (relational), and Neo4j (graph) for their capacity to support the enrichment of structured trip data, collected through smartphone GPS and sensory inputs, with unstructured multimodal commuter feedback encompassing voice, video, images, text, and documents. By benchmarking these three engines across insertion, retrieval, storage, and scalability metrics, the study identifies which architectural paradigm best accommodates the dual data dimensions of structured mobility records and unstructured experiential inputs, while also addressing practical constraints such as data privacy, GPS data accuracy, and smartphone battery depletion.

Existing ITS architecture and design emphasize efficiency, overlooking the actual circumstances of commuters and leading to shortcomings in terms of inclusivity, responsiveness, and user experience. This chapter delineates a method for executing and verifying the BDMobility application as a proof-of-concept platform that integrates these two data dimensions into a graph database framework. The comparative evaluation of relational and graph backends, combined with the merger of structured mobility data with multimodal commuter inputs, demonstrates the feasibility of creating human-centred transportation data frameworks that reflect real-time commuter experience.

This chapter is structured as follows: Section 3.1 outlines the research design that supports this study. Section 3.3 outlines the data collection framework, including the data sources, smartphone application, study context and ethical considerations. Section 3.4 outlines the system design, emphasizing the graph database schema and its advantages over the relational databases. Section 3.6

outlines the data processing and integration pipeline, whereas Section 3.3 describes the deployment and testing processes, including the pilot study design and evaluation criteria. Section 3.7 outlines the methodological limitations of this study. Section 3.8 concludes by linking the methodology to the research questions and literature evaluation.

3.1 Research Design

This study utilizes a proof-of-concept approach to design, implement, and evaluate the proposed system. This ensures that the research focuses primarily on developing and assessing a functional framework that consolidates structured and unstructured trip data into one database. It is worth mentioning that all the structured data were recorded through the participants' mobile device, however the unstructured data were rather generated from one principal test account using the tester's completed trip data. This formed the basis of the experiment to collect real-time multimodal travel data as the trip feedback as a case study to test the proposed backend system.

Furthermore, the central concept is a human-centred transportation data architecture that emphasizes the commuter experience rather than infrastructure efficiency. This corresponds with the acknowledged constraints in the architecture and design of current ITS and location-aware smartphone apps, which primarily limit travel data records to geospatial data while exhausting battery life through continuous location tracking, neglecting the emotional, social, safety, and accessibility aspects of travel.

This design choice prioritizes data acquisition, database architecture, and system integration and establishes a methodological basis for scaling multimodal data integration in future transportation systems and research through system design and pilot testing.

3.2 Use Case–Driven Data Integration

The approach used to integrate and manage the graph database considers the varying data formats and merges structured trip data with multimedia live feedback guided by a use-case-driven requirement-design process. This method guarantees that the system design aligns primarily with

the operational and analytical requirements of end-users, such as mobility planners, autonomous vehicle control systems, and public security analysts. In this addition, the requirement specifications act as the central basis to connect user needs to data modelling, integration architecture, and query semantics. It specifies the purpose the graph database must fulfill (functional requirements) and the performance criteria it must satisfy under practical limitations (nonfunctional requirements). The methodology is elaborated further through a systematic process, commencing with the discovery of fundamental use cases.

The use cases form the fundamental components of the requirement specifications described previously. Each use case specifies the interaction between an actor and the system, thereby guiding both schema design and data flow integration. The principal use cases identified for this system are as follows.

Autonomous vehicle navigation and decision support through querying multimodal trip histories and real-time feedback data to optimize navigation. The following data requirements make it possible to rely on the proposed system for this case study through the integration of spatiotemporal trip data (paths, stops, modes), access to contextual multimedia streams (e.g. videos, images), and low-latency query support for dynamic route adjustment using the best pathfinding feature from the graph database.

Security incident detection and analysis enable transportation authorities and security services to correlate real-time feedback data (audio, video, images, documents, and narrations) with trip sequences for post-event investigations. The following aspects of the backend system allow this case study to be explored through the semantic linkage between events, trips, and user-generated multimedia, and support event timeline reconstruction through graph traversal.

Passenger experience analysis can be performed using the merged data of trips and user feedback to provide contextual data into the lived experiences of commuters across different transportation modes to better understand shift factors in observed travel patterns and provide timely and tailored responses during disruptions. Inferences from the data streams could reveal situations such as overcrowded transits, accessibility challenges, and violence for timely safety considerations.

According to, (Kulak & Guiney, 2012) a use-case-driven approach can be used to elicit and categorize functional and non-functional system requirements. The functional requirements include:

(1) Support for multi-entity graph modelling: nodes (trips, vehicles, users, places) and edges (relationships, transitions, interactions).

(2) Real-time ingestion of multimedia feedback synchronized with trip timestamps.

(3) Query interfaces supporting pattern matching and temporal reasoning.

(4) APIs for integration with AI-driven modules (e.g., anomaly detection, path optimization).

While the non-functional requirements are

(1) Scalability to handle continuous multimedia streams.

(2) Fault tolerance in distributed data ingestion.

(3) Data privacy controls for user-contributed media.

(4) Graph query performance under sub-second latency for operational contexts.

When this system is implemented and adopted, the use case scenarios can be validated through simulations and live deployment trials to ensure the goal of this study in identifying the structural and data requirements necessary to maintain multimodal travel databases is met.

3.3 Implementation Framework

Figure 3.1 contrasts the architecture of existing transportation systems with the proposed BD Mobility platform, illustrating both the structural limitations that motivate this study and the architectural decisions that address them. The figure is organised into four layers, read from left to right: Existing Systems, Trip Segmentation, Backend Data Processing, and User Interaction & Experience.

The left panel of Figure 3.1 depicts existing systems and identifies two categories of limitation that the proposed system must overcome. The first is the fragmented architecture of current Intelligent Transportation Systems (ITS). Existing ITS operate as a collection of siloed subsystems: travel surveys and research, traffic and transit monitoring, ticketing and fare collection, security and incident management, and daily operations management each run concurrently as independent services

within separate organisational units (DeLaurentis, 2005; Elkosantini & Darmoul, 2013). Because these subsystems are not integrated into a unified data architecture, observations made in one subsystem cannot be seamlessly referenced by another, even within the same city or transit agency (Xu et al., 2017). This fragmentation prevents the collection and integration of real-time travel feedback using multimodal data (Lemonde et al., 2021) and limits the ability of agencies to provide timely responses to commuter feedback (Barns, 2020). The relational databases (RDBMS) that underpin each subsystem store data in isolated tables with rigid schemas, and relationships between entities across subsystems can only be reconstructed through costly join operations, as described in Section 3.4.3.

The second category of limitation shown in the existing systems panel concerns smartphone-based GPS data collection. As detailed in Section 2.1, continuous GPS logging suffers from privacy and GDPR compliance concerns, inconsistent network connectivity, GPS signal loss in urban canyons, high battery consumption, and dependence on device ownership and digital literacy (Hosseini, 2025; Prelipcean et al., 2016). These constraints result in incomplete or inconsistent trip records that compromise the representativeness of mobility datasets.

To address these limitations, the proposed system introduces two architectural changes. First, MotionTag is integrated to improve smartphone-based data collection by: (1) refining GPS data through hybrid sensing for higher accuracy, (2) ensuring data privacy through adaptive sampling, and (3) optimising battery usage through intelligent switching between GPS, Wi-Fi, and onboard sensors. Second, a Neo4j graph database replaces the RDBMS backend to handle multimodal data processing and storage within a single, semantically connected data store.

The trip segmentation module in Figure 3.1, supported by the MotionTag SDK, is central to the architecture. This component automatically identifies trips, categorizes them by purpose and mode, and optimizes the data. The system employs a hybrid sensing approach that integrates GPS, Wi-Fi, and onboard motion sensors to obtain precise location and time data while minimizing battery consumption and data usage. Wi-Fi-only transmission facilitates data synchronization, thereby enhancing privacy and conserving energy.

This layer integrates raw GPS recordings with structured trip data by converting continuous sensor information into significant travel events, such as trips, stops, and purposes. The optimized

data is transmitted to the backend via secure RESTful web services, ensuring consistency across mobile and web interfaces.

The backend data processing layer of Figure 3.1 comprises interconnected components designed for scalability, resilience, and the capacity to manage various data types. The system includes NPM and PM2 servers for application management and process monitoring, a REST API for data transfer, and a dedicated file server for multimedia storage.

The Neo4j graph database serves as the core component of the backend architecture. The model delineates the relationships among users, trips, feedback, files, and notifications. Neo4j distinguishes itself from conventional relational databases by allowing a more flexible data modelling approach that emphasizes relationships. This enables the observation of variations in journeys, locations, and commuter feedback over time. This structure facilitates the execution of complex graph queries, enabling the analysis and visualization of travel patterns and user experiences in real time.

The user interaction layer in Figure 3.1 constitutes the visible and operational component of the BD Mobility system for users. Access is available via cellphones, online portals, and wearable devices. Participants can document feedback in various formats, including text, photos, audio, video, and documents, which are directly associated with their verified visits. This user-centered design promotes diversity, reduces memory bias, and incorporates actual commuter experiences into standard travel survey data.

3.4 System Architecture

3.4.1 Overview of Architecture Design

Based on the proposed implementation framework illustrated above, the proposed system is designed as a layered pipeline that depicts the network data path and interactions among various system components within the proposed system. The adoption of a graph database in the proposed system optimizes the backend tasks to collect and process completed trips and multimodal data, along with stated-preference and revealed-preference survey data collected using the app. The initial version of the proposed system utilized an RDBMS, which suffered from architecture and design deficiencies similar to those of ITS. This limitation affects the potential to record and process

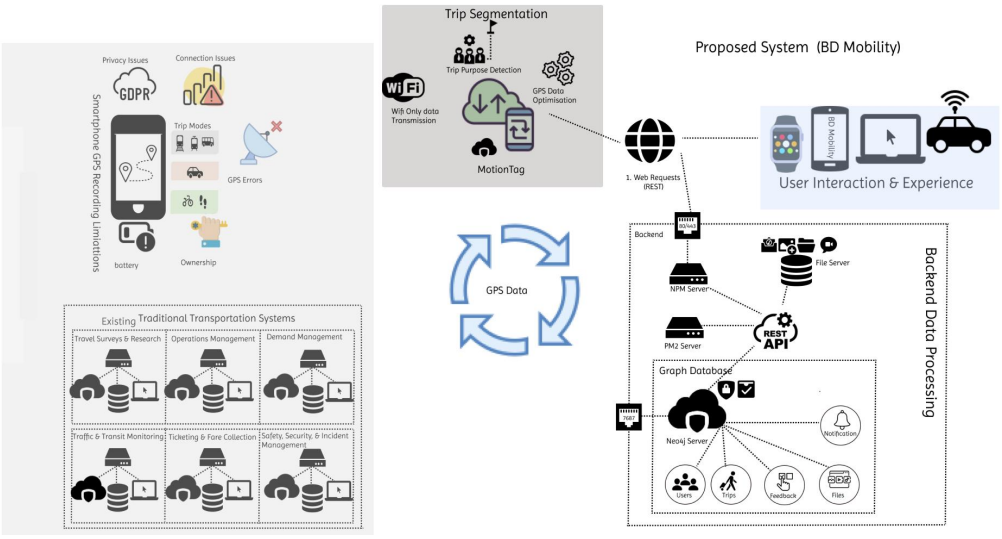

Figure 3.1: Implementation framework contrasting existing systems (left) with the proposed BD Mobility architecture (right). The existing systems panel identifies the siloed ITS subsystems and smartphone GPS logging limitations that the proposed graph-database-backed platform addresses.

multimodal data as trip feedback in real time.

Figure 3.2(a) and (b) present the architecture adopted to overcome the design bottlenecks in the RDBMS and the previous version of the proposed system, respectively. Figure 3.2(a) depicts the network architecture that is interfaced when users access the mobile app over public networks. After the user is authenticated, the MotionTag SDK logs user movement events, which are retrieved and displayed in the trip diary page when stored in the database in the backend system. From the backend system the various actors, i.e., researchers, developers, and system administrators, access the backend system to infer from the recorded data.

Figure 3.2(b) expands on the various components that exist between the frontend (mobile app) and the backend. The two main subcomponents are interlinked using APIs that verify client requests against the backend and respond with acknowledgment. The frontend embodies various subcomponents, including Motiontag movement detection and logging, in-app notifications, and google services for device authentication, maps, and location services. Other aspects that had been defined outside this study include the user surveys, which presents questionnaires on migration integration, travel preferences, and sociodemographics. As described in the implementation framework in

section 3.3 and figure 3.1 this backend system handles automations such as updating the trips and assigning feedback to trips via the Process Manager (PM2) server component included in the internal server in figure 3.2(b), the file server stored user feedback files, and the graph database component maintains the datastore by representing the recorded data as nodes and relationships in the backend system. Web APIs are provisioned to handle events such as user management and authentication to the backend system and to expose aspects of the backend for interconnection by other systems to enhance interoperability.

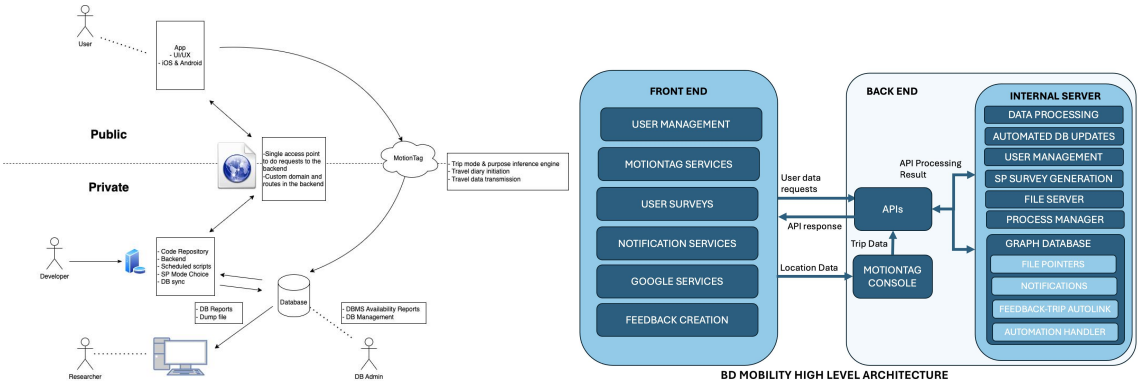

Figure 3.2: (a) Network Architecture (b) High-level Application Architecture

Figure 3.2(a) depicts the data transmission path from a commuter's mobile device to the internal servers, indicating the channels used to transmit and receive user data. In this figure, a complete trip with features such as trip time, mode, purpose, and location coordinates, along with multimodal feedback data is transmitted from the user's smartphone to the backend via wireless communication. The data received were further preprocessed in the backend to eliminate noise and stored. Figure 3.2(b), on the other hand, demonstrates the layers of the entire architecture with a high-level view of the components running on each subsystem, such as the frontend (smartphone application), backend, and graph database.

3.4.2 Graph Database Schema

Graph databases were chosen as the foundation of the multimodal trip feedback system because of their capacity to represent highly interconnected data structures and efficiently navigate interactions across diverse elements of the system. This subsection first describes the two graph database

platforms considered, Neo4j and OrientDB, explaining the technical concepts that distinguish them; it then presents the schema design adopted for this study and explains the transition from a relational entity-relationship model to a graph-based representation.

Neo4j: A Native Property Graph Database

Neo4j is a *native graph database*, meaning that its storage engine is purpose-built from the ground up to store and retrieve graph structures. Data in Neo4j is organised using the *property graph model*, a data representation in which information is stored as three types of elements: *nodes* (entities such as a user, a trip, or a feedback item), *relationships* (directed, typed connections between nodes, such as `TOOK_TRIP` linking a User node to a Trip node), and *properties* (key–value pairs attached to both nodes and relationships, such as `start_time = ''2025-10-15T08:30''`). The term *property graph workloads* refers to database operations that involve creating, reading, updating, and traversing these node-relationship-property structures, as opposed to operating on flat tables or document collections.

Neo4j's most consequential architectural feature is *index-free adjacency*: each node stores direct physical pointers to every node it is connected to. When a query needs to move from one entity to a related entity, the database follows this pointer directly, without consulting a separate index or scanning a table. This design means that the time to traverse a single relationship is constant, regardless of whether the database contains one thousand or one million nodes. Traversing a chain of k relationships costs $O(k)$, proportional only to the length of the chain, not to the total size of the dataset. This property is what makes Neo4j particularly efficient for queries that navigate chains of connections, such as moving from a user to their trips, then to the feedback on each trip, and further to the multimedia files attached to that feedback.

Neo4j queries are written in *Cypher*, a declarative query language designed specifically for graph pattern matching. Cypher uses an intuitive ASCII-art syntax that visually resembles the graph structure being queried: for example, `(u:User)-[:TOOK_TRIP]->(t:Trip)` reads naturally as "find a User node `u` connected by a `TOOK_TRIP` relationship to a Trip node `t`." This visual approach allows complex multi-hop traversals to be expressed concisely, without the explicit `JOIN` clauses required in SQL.

Neo4j's ecosystem is the most mature among graph databases. It provides a comprehensive set of built-in graph algorithms (shortest path, community detection, centrality measures), a visual browser for interactive graph exploration, official drivers for all major programming languages, and extensive integration with analytics and machine learning pipelines. The platform has been widely adopted in transportation, logistics, social network, and knowledge graph applications, providing a substantial body of reference implementations and community support (Neo4j, Inc., 2025; Webber, Robinson, & Eifrem, 2012).

OrientDB: A Multi-Model Database

OrientDB, developed by (OrientDB Community, 2025), takes a fundamentally different approach. It is a *multi-model database*, which means that it combines several different data storage paradigms within a single database engine. Specifically, OrientDB integrates four models: a graph model (nodes and edges), a *document model* (semi-structured records similar to JSON objects, where each record can have a different set of fields), a *key-value model* (simple lookup pairs where a unique key maps to a stored value), and an object-oriented model (data stored as programming-language objects with inheritance). The term *schemaless document management* refers to OrientDB's ability to store documents without requiring a predefined schema; each document can contain different fields and nested structures, which provides flexibility for applications where data formats vary or evolve over time.

However, this multi-model flexibility comes at a cost. Because OrientDB must support four different storage paradigms simultaneously, its internal architecture cannot be exclusively optimised for any single one. In particular, its graph traversal performance is slower than that of a native graph database like Neo4j, because OrientDB does not implement true index-free adjacency; instead, it resolves connections through internal document references that require additional lookup steps. Benchmark comparisons have consistently shown that Neo4j outperforms OrientDB on property graph workloads, particularly for deep traversals and pattern-matching queries (Fernandes & Bernardino, 2018; Virić & Pantelić, 2023). Furthermore, OrientDB's ecosystem and community support are substantially more limited than Neo4j's, constraining the availability of pre-built graph algorithms and their integration with analytics pipelines.

Table 2.2 presents a detailed comparison between Neo4j and OrientDB, highlighting why Neo4j was selected for this study.

Justification for Neo4j in the Multimodal Feedback System

The central requirement of this study's backend is to support what is referred to throughout this thesis as the *multimodal feedback process*: the workflow in which a commuter captures a piece of feedback during or after a trip using one or more multimedia formats (a photograph, an audio recording, a video clip, a text comment, or a document), and the system must store that feedback in a way that preserves its relationship to the specific trip, the trip's spatial geometry, the commuter's user profile, and the file metadata (timestamps, geotags, file type, and size). Each feedback submission therefore creates a chain of linked entities: a User who `TOOK_TRIP` to a Trip, which `HAS_GEOMETRY` referencing spatial coordinates, which `HAS_FEEDBACK` linking to a Feedback item, which in turn `HAS_FILE` connecting to one or more multimedia File nodes.

Neo4j's traversal efficiency makes it possible to query this entire chain in real time because each step in the chain follows a stored physical pointer rather than executing a table join. When a researcher queries "find all video feedback submitted by User X during trips that passed through Station Y," Neo4j begins at the User node, follows the `TOOK_TRIP` pointers to that user's Trip nodes, filters by geometry, follows the `HAS_FEEDBACK` pointers to Feedback nodes, and finally follows the `HAS_FILE` pointers to File nodes filtered by type. Each pointer traversal occurs in constant time regardless of how many total users, trips, or files exist in the database. In a relational system, the same query would require four consecutive `JOIN` operations across separate tables, with cumulative cost that grows with data volume. This architectural advantage is what enables Neo4j to support the real-time querying of relationships among trips, files, and feedback that the multimodal feedback process demands.

From Entity-Relationship Diagrams to Graph Schema

The data model for this study was designed by first constructing a traditional entity-relationship (ER) diagram and then translating it into a Neo4j graph schema. Understanding this transition clarifies how the graph representation preserves the same information as the relational model while

adding structural advantages that the relational model cannot provide.

An *entity-relationship (ER) diagram* is the standard design tool for relational databases. It represents data as a set of *entities* (rectangles), each with defined *attributes* (the columns that will become the table's fields), connected by *relationships* (lines between rectangles) that indicate how entities relate to one another. In an ER diagram, relationships are *implicit*: they are implemented through foreign keys, which are columns in one table whose values reference the primary key of another table. The ER diagram for this study (Figure 3.3) delineates four principal entities, User, Trip, Feedback, and File, with relationships indicated by foreign-key links: the Feedback table contains a `trip_id` column that references the Trip table, the File table contains a `feedback_id` column that references the Feedback table, and so on.

A *graph schema*, by contrast, represents the same entities as *nodes* and the same relationships as *explicit, typed edges* that are stored as first-class objects in the database. The Neo4j schema for this study (Figure 3.4) depicts the same four entities as node types (User, Trip, Feedback, File) interconnected through named relationship types (`TOOK_TRIP`, `HAS_FEEDBACK`, `HAS_FILE`, `IS_TRIP_ASSIGNED`, `HAS_GEOMETRY`). The critical difference is that in the graph schema, each relationship is a physical object stored alongside the nodes it connects, carrying its own properties (such as a timestamp or a role label). In the ER diagram, the same relationship exists only as a matching pair of values in two separate tables and must be reconstructed through a join operation every time it is queried.

This distinction has three practical consequences for the multimodal feedback system. First, *semantic preservation*: in the graph schema, a user's submitted video is not stored as an isolated row in a File table linked to a Feedback table through an integer foreign key; instead, it exists as a File node directly connected to a Feedback node through a `HAS_FILE` relationship, which in turn is connected to a Trip node and its Geometry node. The chain of relationships is stored explicitly, preserving the narrative context that gives each media object its meaning. Second, *query efficiency*: retrieving the full context of a feedback item (who submitted it, during which trip, at what location, with which attached files) requires following four stored pointers in Neo4j, versus executing four join operations in a relational system. Third, *schema flexibility*: adding a new entity type or relationship type to the graph schema (for example, adding a Notification node or a `SHARED_WITH`

relationship) requires no alteration to existing nodes or tables, whereas the equivalent change in a relational schema requires adding new tables, foreign-key columns, and potentially modifying existing queries.

The transition from relational to graph modelling therefore results in a framework that supports dynamic exploration of commuter experiences rather than static reporting. Relationships can be traversed and analysed through patterns, such as identifying the frequent co-occurrence of specific transport modes and feedback sentiments, or detecting temporal clusters of negative feedback at particular locations. These capabilities facilitate advanced applications including real-time trip monitoring, anomaly detection, and multimodal feedback analytics, none of which can be efficiently supported by the join-dependent architecture of a relational database at the data volumes encountered in this study.

In summary, Neo4j was selected over OrientDB because its native property graph architecture, index-free adjacency, mature ecosystem, and Cypher query language provide the combination of traversal efficiency, semantic expressiveness, and scalability that the multimodal feedback process requires. The graph schema directly corresponds to the project's objective of capturing not only data points but also the narratives and experiences that arise from the relationships among users, trips, and feedback.

3.4.3 Database Paradigms and Performance Implications

The choice of database paradigm fundamentally shapes how data is stored, queried, and scaled. In the context of this study, three database engines are evaluated: two relational database management systems (RDBMS), MySQL and PostgreSQL, and one graph database, Neo4j. Each paradigm embodies a distinct philosophy regarding the organisation and retrieval of data, and understanding these differences is essential for interpreting the performance, storage, and scalability results presented in Chapter 4. This subsection begins with the relational paradigm, examines its strengths and inherent limitations, then introduces the graph paradigm as an alternative that addresses several of these limitations. The theoretical contrasts are then summarised in Table 3.1, with each dimension explained in the discussion that follows.

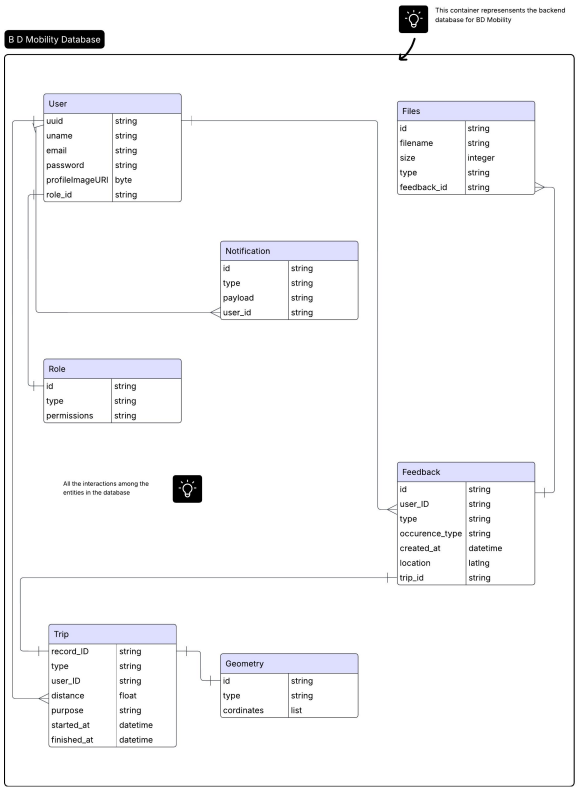

Figure 3.3: This is the Entity-Relationship diagram illustrating the Graph database in Figure 3.4

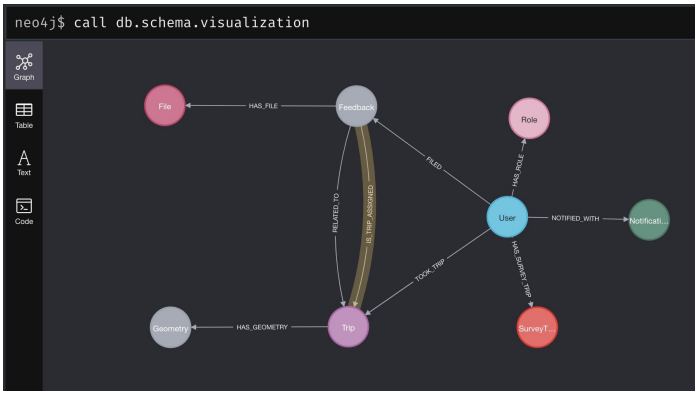

Figure 3.4: This is the Neo4j graph database schema supporting the backend processes and feedback creation.

The Relational Paradigm: MySQL and PostgreSQL

Relational database management systems store data in *tables*, where each table is composed of *rows* (individual records) and *columns* (attributes that describe those records). A table schema defines the columns, their data types, and any constraints in advance, meaning that the structure

of the data must be specified before data can be inserted. For example, a trip table might include columns for `trip_id`, `user_id`, `start_time`, and `trip_mode`, with each row representing a single trip record. This tabular organisation is conceptually analogous to a spreadsheet with rigid column headings.

Relationships between tables are established through *foreign keys*, which are columns in one table that reference the primary key of another table. For instance, a `feedback` table might include a `trip_id` column whose values must correspond to valid entries in the `trip` table. Importantly, these relationships are not stored as independent structures; they are *implicit*, existing only as matching values across columns. When a query requires data from more than one table, the database engine performs a *join operation*, which combines rows from different tables by matching their foreign-key values at the time the query is executed. A simple join, for example, combines each feedback row with its corresponding trip by scanning and matching on `trip_id`.

Both MySQL and PostgreSQL use *B-tree indexes* as their primary mechanism for accelerating data lookups. A B-tree (balanced tree) is a hierarchical data structure in which data entries are sorted and stored across multiple levels, much like an alphabetised phone directory where one can skip to the correct section, then the correct page, and finally the correct entry. This structure allows the database to locate a specific record in *logarithmic time*, expressed as $O(\log n)$, meaning that doubling the amount of data adds only one additional level to the search. For a table with one million rows, a B-tree lookup typically requires only about 20 comparisons rather than scanning all one million records. Both MySQL (through its InnoDB storage engine) and PostgreSQL rely on B-tree indexes for primary-key and most secondary lookups. PostgreSQL additionally supports several specialised index types: GIN (Generalised Inverted Index) indexes for full-text search and JSON queries, GiST (Generalised Search Tree) indexes for spatial and geometric data, and BRIN (Block Range Index) indexes for very large tables where data is physically sorted by a column such as a timestamp (The PostgreSQL Global Development Group, 2024).

While the relational paradigm excels at structured, tabular workloads, it encounters a fundamental limitation when data is highly interconnected. Consider a query that must traverse from a user to their trips, then to the feedback on each trip, and further to the files attached to that feedback. In a relational system, each step requires a separate join operation. Join cost grows with the number

of tables involved because the database must repeatedly scan, sort, or hash intermediate results. For two-table joins this overhead is manageable, but queries that involve three, four, or more consecutive joins can degrade rapidly. The time complexity of such nested joins is often characterised as $O(n \log n)$ per join step, where n is the number of rows in the participating tables. When multiple joins are chained, these costs compound, and the total query time can grow disproportionately with the depth of interconnection. This phenomenon, referred to as the *join penalty*, represents the central performance bottleneck of relational systems for relationship-dense data (Webber et al., 2012).

MySQL and PostgreSQL differ in how they address this limitation. MySQL, through its InnoDB engine, prioritises simplicity and raw throughput for read-heavy transactional workloads. Its optimiser is efficient for straightforward queries but offers limited capabilities for complex analytical operations. PostgreSQL, by contrast, provides a substantially more expressive relational framework. It supports advanced query constructs such as common table expressions, window functions, and recursive queries, as well as native handling of JSON documents, arrays, and user-defined data types. These features make PostgreSQL more adaptable for complex analytical workloads, though both systems ultimately share the same relational join-based architecture and therefore the same fundamental scalability constraint when queries involve deeply interconnected data.

The Graph Paradigm: Neo4j

Neo4j's architecture, query language, and ecosystem were described in detail in Section 3.4.2. For the purposes of the performance comparison that follows, the key architectural properties to recall are these. Neo4j stores data as nodes, relationships, and properties (the property graph model). Each node maintains direct physical pointers to its neighbours (*index-free adjacency*), so traversing a single relationship is a constant-time operation regardless of dataset size. Following a chain of k relationships therefore costs $O(k)$, proportional only to the chain length, not to the total number of records. Queries are written in *Cypher*, a pattern-matching language that expresses multi-hop traversals concisely without the explicit `JOIN` clauses required in SQL.

These properties produce a fundamentally different performance profile from the relational paradigm. Where MySQL and PostgreSQL must execute a separate join operation for each step in a relationship chain, with cumulative $O(n \log n)$ cost per join, Neo4j follows stored pointers at

each hop with constant cost per step. This contrast is most consequential for the multimodal feedback data model used in this study, where a single query may traverse from a User through their Trips, to Feedback entries, to attached multimedia Files, and to spatial Geometry nodes, spanning five entity types. In a relational system, this query requires four consecutive joins with compounding cost; in Neo4j, the same traversal follows four direct pointers (Neo4j, Inc., 2025; Webber et al., 2012).

Comparative Analysis

The architectural differences between the relational and graph paradigms produce measurable tradeoffs in performance, memory consumption, and suitability for different workload types. Table 3.1 summarises these contrasts across eleven dimensions, each of which is discussed below.

Table 3.1: Theoretical comparison of Neo4j, PostgreSQL, and MySQL

Dimension	**Neo4j**	**PostgreSQL**	**MySQL**
Database Model	Native graph database	Relational database	Relational database
Core Data Structure	Nodes, relationships, properties	Tables (rows and columns)	Tables (rows and columns)
Relationship Representation	Explicit (stored as edges)	Implicit (foreign keys and joins)	Implicit (foreign keys and joins)
Primary Query Language	Cypher	SQL	SQL
Indexing Mechanism	Native graph indexes	B-tree, GIN, GiST, BRIN	B-tree (InnoDB), limited hash
Lookup Complexity	$O(\log n)$ (indexed lookup)	$O(\log n)$ (B-tree)	$O(\log n)$ (B-tree)
Traversal / Join Complexity	$O(k)$ for k-hop traversal	Join-dependent, often $O(n \log n)$	Join-dependent, often $O(n \log n)$
Multimodal Data Support	Properties on nodes and edges	Native JSON, arrays, extensible types	Limited native JSON support
Memory Usage	High (relationship-heavy traversal)	Moderate and configurable	Lower, optimized for simplicity
Scalability Strength	Relationship-dense workloads	Complex analytical workloads	Read-heavy transactional workloads
Primary Limitation	Memory overhead at scale	Join cost at high connectivity	Limited extensibility for complex data

The first two rows of Table 3.1 distinguish the fundamental storage philosophy to explain the database model and core data structure. Neo4j is classified as a native graph database because its storage engine is purpose-built around nodes and relationships; internally, every entity and connection is stored as a distinct record with fixed-size pointers to adjacent elements. MySQL and PostgreSQL, as relational databases, store all data in tables composed of rows and columns. In the relational model, there is no native concept of a "connection" between records; all associations must be expressed through column values (foreign keys) and resolved at query time.

The relationship representation dimension captures the most consequential difference. In Neo4j, relationships are *explicit*: each edge is physically stored in the database alongside the two nodes it connects, carrying its own type label and properties. Moving from one node to a related node requires only following a stored pointer. In MySQL and PostgreSQL, relationships are *implicit*: they exist only as matching values across foreign-key columns in separate tables. The database discovers these connections only when a join operation is executed, requiring index lookups and row matching at query time.

The databases use different query languages. Neo4j uses Cypher, a pattern-matching language in which queries are expressed as visual graph patterns (e.g., node–edge–node chains). MySQL and PostgreSQL both use SQL, a set-based language that operates on tables through operations such as `SELECT`, `JOIN`, and `WHERE`. For queries involving chains of relationships, Cypher typically produces shorter and more readable statements, whereas SQL requires explicit join clauses for each table involved.

In terms of the indexing mechanism, all three databases support B-tree indexes for single-attribute lookups with $O(\log n)$ complexity. Neo4j additionally maintains native graph indexes that complement its pointer-based adjacency structure. PostgreSQL extends beyond the basic B-tree with GIN indexes (optimised for composite values such as JSON documents and arrays), GiST indexes (for spatial and range queries), and BRIN indexes (for very large, physically sorted tables). MySQL's InnoDB engine offers B-tree and limited hash indexing, providing fewer options for specialised workloads.

With regard to lookup and traversal complexity, when performing a single indexed lookup, all three systems achieve $O(\log n)$ performance. The critical divergence occurs during traversal.

In Neo4j, following k consecutive relationships costs $O(k)$ because each hop is a direct pointer dereference, independent of dataset size. In MySQL and PostgreSQL, each hop translates to a join operation, and the cumulative cost is often $O(n \log n)$ per join, where n represents the rows in the joined tables. As the number of hops increases, the compounding join costs in relational systems grow substantially, while Neo4j's traversal cost scales linearly with the number of hops alone.

Storing diverse data types is important for this study's multimedia feedback model. Neo4j accommodates heterogeneous data by attaching arbitrary properties (strings, numbers, arrays, spatial types) to both nodes and relationships, meaning that a feedback node and its associated file node can each carry different sets of attributes without schema modification. PostgreSQL provides strong support through its native JSON/JSONB columns, array types, and extensible type system, making it highly flexible for semi-structured data within a relational schema. MySQL offers more limited native JSON support and fewer extensible data types, which restricts its ability to handle complex, heterogeneous records without additional application-level processing.

Memory usage and scalability strength varies across the various databases. Neo4j's explicit storage of every relationship as a physical record results in higher baseline memory consumption compared with relational systems, where relationships exist only as column values. This overhead is the cost of maintaining the direct pointers that enable constant-time traversal. PostgreSQL occupies a middle ground with configurable memory allocation for caching, sorting, and query execution, making it suitable for complex analytical workloads that involve aggregations and subqueries. MySQL's simpler memory model is optimised for high-throughput, read-heavy transactional scenarios where query patterns are predictable and table structures are straightforward.

Each paradigm's strength is also the source of its primary constraint. Neo4j's memory overhead becomes significant as the total number of stored relationships grows into the hundreds of millions, since each relationship consumes physical storage regardless of whether it is actively queried. For PostgreSQL, the join penalty at high connectivity means that queries involving many table joins experience compounding latency as data volume increases. MySQL's limited extensibility, including fewer index types, less expressive query capabilities, and restricted support for complex data types, constrains its applicability for workloads that fall outside simple transactional patterns.

Understanding these theoretical distinctions is essential for interpreting the empirical results in

Chapter 4, where the three engines are benchmarked under identical data volumes ranging from 1,000 to 500,000 records. The tradeoffs summarised here, particularly the contrast between Neo4j's constant-time traversal and the relational join penalty, provide the conceptual framework for analysing the observed performance, storage, and scalability outcomes.

3.5 Data Processing and Integration

3.5.1 Data Preprocessing

Preprocessing was performed on both structured and unstructured data sources to ensure the analytical consistency of integration. Raw GPS logs from the BD Mobility application were partitioned into individual trips based on dwell time and distance thresholds, in accordance with accepted methodologies in transportation research (Patterson et al., 2019; Schuessler & Axhausen, 2009). Table 3.2 describes the trip attributes evaluated in the data preparation to remove erroneous or noisy data resulting from signal drift, multipath effects, or reception loss using spatial smoothing and temporal interpolation methods (Zheng, Li, Chen, Xie, & Ma, 2008). Through MotionTag integration, transfer sites among transport modes were determined by stop clustering and variations in speed profiles, facilitating the accurate delineation of multimodal routes (Gonzalez, Hidalgo, & Barabási, 2008; Y. Liu, Kang, Gao, Xiao, & Tian, 2012).

Figure 3.5 depicts the flow and interactions involving user-generated feedback in unstructured data, encompassing text, audio, images, documents, or video, augmented with metadata, including timestamps, geotags, and contextual indicators such as "before trip," "during trip," or "post-trip," adhering to best practices in multimodal data management for instance , data modelling, metadata integration, data governance and scalable data storage and retrievals. (Goodchild, 2007; Montoya-Torres, Moreno, Guerrero, & Mejía, 2021). In figure 3.5, the process of providing feedback requires more contextual data that enriches the feedback by providing multimodal data, which combines with metadata on files to ensure easy data retrieval and storage, as its possible to identify and secure multimedia data by generating authentication keys that render the data only accessible to the corresponding users.

Table 3.3 provides the feedback data structure and describes the attributes collected from feedback generated by commuters. Multimedia files were compressed into efficient forms to save storage requirements while preserving the analytical quality of the file metadata stored in the backend. The properties of the feedback stored are highlighted in Table 3.4. Feedback objects were indexed by time, position, and type to enhance retrieval efficiency in subsequent processes (Stonebraker & Çetintemel, 2005).

Table 3.2: Structured Trip Data Attributes

Attribute	**Description**
trip_id	Unique identifier for each trip segment.
user_id	Identifier linking the trip to a specific user.
trip_type	Classification of trip (e.g., work, leisure, shopping).
trip_mode	Primary mode of travel (e.g., bus, metro, walking, car).
trip_purpose	Declared or inferred purpose of the trip.
start_time	Timestamp marking the beginning of the trip.
end_time	Timestamp marking the end of the trip.
start_time_zone	Time zone corresponding to the trip start.
end_time_zone	Time zone corresponding to the trip end.
location_points	Sequence of GPS coordinates representing the trip path.
distance	Total distance travelled during the trip, calculated from GPS coordinates.

Table 3.3: Feedback Data Attributes

Attribute	**Description**
id	Unique identifier for each feedback record.
occurrence_type	Event triggering the feedback (e.g., sudden stop, delay, transfer).
feedback_type	Nature of the feedback (e.g., text, audio, image, rating).
comments	User-provided description or qualitative remarks.
location_points	Spatial reference (coordinates or stop ID) of the feedback.
user_id	Identifier linking feedback to a specific user.
trip_id	Identifier linking feedback to a specific trip (if matched).
mediaURLs	References to associated multimedia files (audio, video, images).
creation_time	Timestamp of feedback submission.
status	Processing state (e.g., linked, unassigned, pending review).

Integration of Structured and Unstructured Data

The synchronization of structured trips data and unstructured feedback data entails matching timestamps between recorded trips and feedback, this process filters both data streams by date and

Table 3.4: Files Data Attributes

Attribute	Description
size	File size (in bytes, MB, or GB depending on media).
type	File type or format (e.g., .mp3, .mp4, .jpg).
feedback_id	Identifier linking the file to its parent feedback record.
user_id	Identifier linking the file to the uploading user.
url	Storage location or retrieval link for the file.

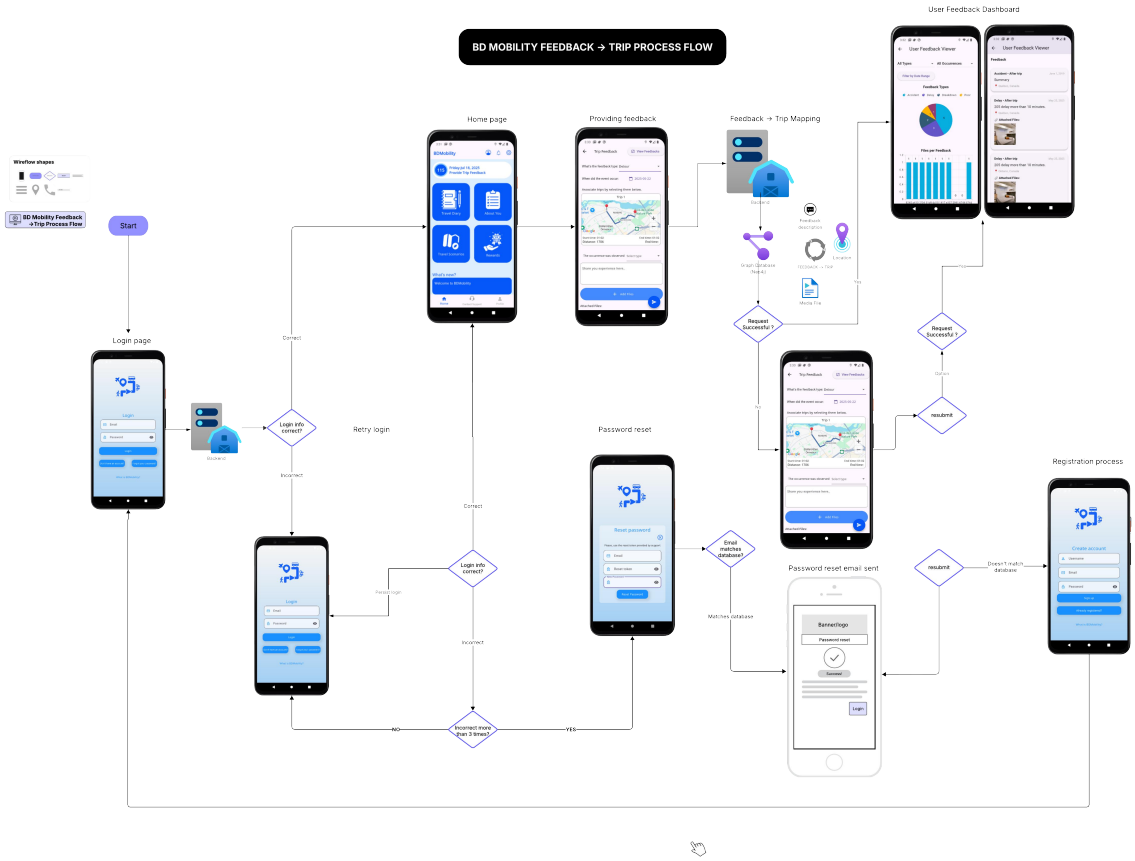

Figure 3.5: User interaction and Data Flow from mobile application to the backend

user data. In this process, trip feedback were associated with trip segments by spatio-temporal matching, wherein timestamps such as creation_time from table 3.3 and the midpoint of the start and end time in table 3.2 were cross-validated and the geotags were correlated with stop sites or trajectories (Shen & Stopher, 2014). Entries that failed to fulfill the matching thresholds were retained as unassigned to maintain data integrity and could be manually matched by the users and stored using the RELATED_TO relationship in figure 3.4. Feedback was directly integrated with the transport network within the Neo4j database by associating feedback nodes with trip nodes via the IS_TRIP_ASSIGNED relationship. This graph-based model facilitated the simultaneous study of mobility behaviour and user sentiment, enabling sophisticated query operations, including the extraction of negative feedback linked to certain routes or peak times (Angles & Gutierrez, 2008). The integration pipeline included a dynamic assignment logic, ensuring that the feedback was either aligned with the relevant trip segment automatically or marked for subsequent review to be manually assigned by the user, thereby balancing automation with data quality assurance, as the user can further review and verify the process by ensuring the feedback is matched to the right trip, as shown in Figure 3.6.

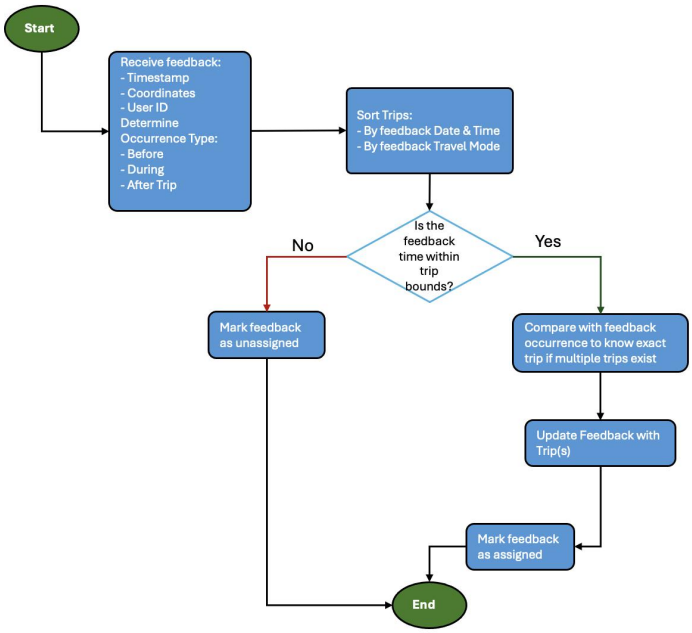

Figure 3.6: Feedback-to-Trip integration process flow

3.6 Deployment and Testing

To test the system, the main assumption made was that commuters often provide feedback on any transportation service, such as rides, flights and etc. based on a holistic review after the service is delivered. While it was believed that certain in-transit experiences, not limited to only safety and comfort but also the duration of the trip, could initiate real-time feedback, existing mobility and ride-hailing apps limit the feedback submission to text-based narrations and may require different service requests when certain complaints need to be made, which introduces a burden to the user during such situations. In light of this, our study considers the possibility of introducing a real-time feedback capture mechanism as a proof-of-concept that employs automations and multimedia user data as feedback to provide more context to the experiences of commuters while in transit, offering the possibility to extend post-trip reviews to as-and-when.

3.6.1 Proof-of-Concept Implementation

The proof-of-concept aimed to validate the viability of a graph-based methodology for simulating multimodal travel behaviour, which was enhanced by the user-generated input. The backend architecture was constructed using the Neo4j graph database, which was selected for its inherent support for relationship-oriented data and adaptable schema, which are crucial for recording temporally and geographically linked trip events, transportation modes, and user annotations (Webber et al., 2012). User trips were organized as nodes connected to metadata (timestamps, modes) and related feedback content, including photographs, audio notes and textual comments. This architecture facilitates in-depth querying of contextual mobility narratives and temporal insights beyond the functionalities of conventional relational databases. The BD Mobility application facilitated real-time sensing through the device's GPS and accelerometer, allowing passive trip detection. In addition, it offers an interface for commuters to submit feedback through multimodal inputs (e.g., audio narration, videos, images, and short comments), which are synchronized with the backend via secure RESTful Application Programming Interfaces (APIs). This architecture is illustrated in Figure 3.1, where the data ingestion layer, API gateway, graph database, and mobile interface collectively support the feedback-driven mobility system. The overall system design follows mobile

sensing paradigms successfully used in participatory transportation studies (Lane et al., 2010).

3.6.2 Pilot Study Design

A field deployment was conducted to assess the performance and user experience of the proposed prototypes. A few urban commuters were selected from Toronto and Montreal, representing a varied array of multimodal users, including those who used the subway, bus, bicycle, and walking. The participants used the initial version of BDMobility application for a sustained period of two weeks, which enabled the collection of organic data in typical commuting conditions. This pilot study aimed to test the enhanced BDMobility application and evaluate (i) the accuracy of automatic trip identification in relation to user-reported activity logs, (ii) the usability and effectiveness of the multimodal feedback interface, and (iii) the subjective user experience regarding the system's utility and burden. During the study, over 150 unique journeys were recorded, accompanied by several individual feedback responses from various media channels. This dataset enabled the technical validation and qualitative analysis of commuter tales, allowing for the thematic classification of user motivations and emotional reactions to disruptions or significant occurrences.

3.6.3 Evaluation Criteria

The evaluation of the proposed system addresses three dimensions: technical performance of the database backend, usability from the commuter's perspective, and research utility of the collected data. This subsection details the performance benchmark criteria, explains why each metric was selected, describes the rationale for testing under varying data scales, and summarises the usability and research-value assessments.

Performance Benchmark Framework

The central evaluation compares the Neo4j graph database against two relational systems, PostgreSQL and SQL Server, under identical workloads. Formally, let $D = \{d_1, d_2, \ldots, d_n\}$ represent the set of databases under evaluation and $M = \{m_1, m_2, \ldots, m_k\}$ the set of performance metrics. Each metric $m_k(d_i, s_j)$ is computed for database d_i at data scale s_j, producing a three-dimensional evaluation matrix of engine, metric, and scale. The ten metrics selected are organised into three

groups: *operation latency* (how quickly the database completes core tasks), *system resource consumption* (how much hardware capacity is required), and *scalability behaviour* (how performance changes as data volume grows). Each metric and its rationale are described below.

Operation Latency Metrics. **Insert Time** (T_{insert}, milliseconds), this metric measures the elapsed time to write a batch of trip records into the database. Insertion is the first operation in any data pipeline; if a database cannot ingest data quickly, downstream processes such as querying and analysis are delayed. Insert time is expected to differ across paradigms because relational systems must enforce schema constraints and foreign-key integrity checks during each write, whereas Neo4j creates nodes and relationships with direct pointer allocation. At small scales, relational engines may be faster because their single-table inserts avoid Neo4j's overhead of establishing relationship pointers; at large scales, the cumulative cost of constraint checking in relational systems is expected to grow, potentially shifting the advantage.

Fetch Time (T_{fetch}, milliseconds), records how long the database takes to retrieve a complete set of records. This metric is important because data retrieval is the most frequent operation in any analytical or reporting workflow. The key architectural difference is that Neo4j retrieves connected data by following stored pointers (index-free adjacency), while relational systems must execute join operations to reconstruct relationships between tables. As the number of records increases, relational join costs compound, whereas Neo4j's pointer traversal cost remains proportional to the number of connections followed. This metric therefore directly tests whether the theoretical traversal advantage of graph databases holds in practice.

Query Latency (L_q, milliseconds), measures the total round-trip time for a structured query that involves filtering, pattern matching, or multi-table access. While fetch time measures bulk retrieval, query latency captures the overhead of the database's query planner, optimizer, and execution engine. This distinction matters because relational databases employ cost-based optimizers that evaluate multiple execution plans before selecting one, adding planning overhead that may become significant for complex queries. Neo4j's Cypher planner, by contrast, is optimised for graph pattern matching and typically produces execution plans with fewer intermediate steps. Differences in query latency across engines reveal how each system's internal planning and execution architecture

responds to increasing data complexity.

Indexing Time (T_{index}, milliseconds), measures how long the database takes to build or update its index structures after data is inserted. Indexes are critical for query performance: without them, the database must scan every record to find matches. This metric is included because the three systems use fundamentally different indexing strategies. Neo4j builds native graph indexes that reference nodes and relationships through pointer structures, which requires traversing existing graph elements during index construction. PostgreSQL and SQL Server use B-tree indexes, which sort and balance data entries hierarchically. The initial overhead of Neo4j's graph-aware indexing is expected to be higher at small scales, but as node density increases, the marginal cost of adding new entries to an already-established graph index is expected to decrease, while B-tree rebalancing costs remain proportional to tree depth.

System Resource Metrics. **CPU User Time** (t_u, milliseconds) **and CPU System Time** (t_s, milliseconds), these two metrics decompose the total processor time consumed by the database process. CPU user time reflects computation performed by the database engine itself, such as query parsing, data transformation, and result assembly. CPU system time reflects operating-system-level work performed on behalf of the database, such as memory allocation, disk I/O scheduling, and network operations. Separating these components is important because it reveals *where* computational effort is spent. A database with high user time but low system time is spending most of its effort on data processing, whereas high system time may indicate excessive disk access or memory management overhead. Relational systems are expected to show higher user-time values at large scales due to the computational cost of evaluating join operations and maintaining transaction logs, while Neo4j's in-memory traversal model is expected to keep user-time growth modest.

CPU Utilisation (U_{cpu}, percentage). Overall CPU utilisation is computed as:

$$U_{\text{cpu}} = \frac{t_u + t_s}{t_p} \times 100\%$$

where t_p is the total wall-clock process time. This aggregate metric captures the proportion of

available processor capacity consumed by the database during the benchmark. Lower CPU utilisation at equivalent workloads indicates more efficient use of hardware resources. Graph databases are expected to maintain lower utilisation because pointer-based traversal avoids the repeated index lookups and sort-merge operations that consume CPU cycles in relational join processing.

RAM Usage (R_{rss}, megabytes). Resident set size (RSS) measures the amount of physical memory occupied by the database process at the end of each benchmark run. Memory consumption is a practical constraint in production deployments; a database that requires substantially more RAM limits the hardware on which it can be deployed. Neo4j's explicit storage of relationship pointers results in higher baseline memory usage, but this allocation is expected to remain relatively stable across scales because the pointer structure is pre-allocated. Relational systems, by contrast, dynamically allocate memory for join buffers, temporary result tables, and transaction logs, leading to memory growth that is more directly proportional to data volume.

Storage Consumption (S, bytes, normalised to $S_{\text{MB}} = \frac{S}{1024^2}$). Disk storage measures the total space occupied by the database files on disk. This metric is included because storage costs scale directly with data volume in production environments. The three systems differ in how they serialise data: Neo4j uses fixed-length records with compact adjacency pointers, PostgreSQL uses page-based storage with write-ahead logging, and SQL Server maintains index duplicates and transaction log files alongside the primary data. These architectural differences are expected to produce substantial divergence in storage footprint, particularly at large scales where index overhead and log files accumulate.

Scalability Metric. **Scalability Efficiency** (E_{scale}). This metric quantifies the rate at which performance degrades as data volume increases, expressed as:

$$E_{\text{scale}} = \frac{\Delta T}{\Delta N}$$

where ΔT is the change in average execution time and ΔN is the corresponding change in record count. A lower value of E_{scale} indicates that the database absorbs data growth with minimal performance loss; a higher value signals that execution time is growing faster than data volume, indicating

an architectural bottleneck. This metric is the most direct test of the theoretical complexity distinctions outlined in Section 3.4.3: Neo4j's $O(k)$ traversal cost predicts near-constant E_{scale}, while the relational $O(n \log n)$ join cost predicts increasing E_{scale} at larger scales.

Rationale for Varying Record Sizes

The benchmark evaluates all ten metrics across 27 data scales, ranging from 1,000 to 500,000 records. This range and granularity were chosen for three reasons.

First, the range spans from a size that is trivially small for any modern database (1,000 records) to a size that begins to stress single-node deployments (500,000 records with multiple relationships per record). This range is representative of the data volumes encountered in urban mobility applications, where a medium-sized pilot study may generate tens of thousands of trip records and a city-wide deployment may produce hundreds of thousands within months.

Second, performance differences between database paradigms are not uniform across scales. At small scales, relational systems benefit from mature B-tree indexing and simple query plans, often matching or exceeding graph database performance. As scale increases, the compounding cost of join operations in relational systems, and the corresponding stability of pointer-based traversal in graph systems, is expected to create a *crossover point* beyond which the graph database outperforms the relational alternatives. Testing at many intermediate scales (5,000; 10,000; 20,000; 30,000; and so on) allows this crossover to be identified precisely rather than inferred from only two or three data points.

Third, fine-grained scaling reveals whether performance degradation is *linear*, *sublinear*, or *superlinear* with respect to data growth. A database that degrades linearly can be provisioned with predictable resource requirements; one that degrades superlinearly may encounter capacity limits unexpectedly. The 27-point scale provides sufficient resolution to fit mathematical growth curves and compute scalability efficiency gradients with statistical confidence, as detailed in the extended analysis in Section 4.3.1.

This evaluation excludes multimedia data upload rates, as file transfer speed is solely dependent on the end-user's network connectivity and is not a property of the database engine. The benchmark metrics and scale range correspond to established practices in database performance evaluation for

graph analytics (Angles & Gutierrez, 2008).

Database Design Assumptions and Schema Considerations

The benchmark implementation required defining a concrete schema for each database engine under test. Because each paradigm imposes different constraints on data type declarations, attribute sizing, and storage semantics, the schema design itself introduces assumptions that influence the measured metrics. This subsection documents these design choices, explains how they differ across the three engines, and identifies the benchmarking implications that arise from those differences. The schema for each engine is summarised in table below.

Table 3.5: Benchmark schema comparison across PostgreSQL, SQL Server, and Neo4j

Attribute	PostgreSQL	SQL Server	Neo4j
record_id	VARCHAR(255)	NVARCHAR(255)	String (unbounded)
user_id	VARCHAR(255)	NVARCHAR(255)	String (unbounded)
type	VARCHAR(100)	NVARCHAR(100)	String (unbounded)
geometry	TEXT	NVARCHAR(MAX)	String (unbounded)
track_mode	VARCHAR(50)	NVARCHAR(50)	String (unbounded)
track_length	NUMERIC	FLOAT	Double
started_at	TIMESTAMP	DATETIME2	DateTime
finished_at	TIMESTAMP	DATETIME2	DateTime
timestamp	TIMESTAMP	DATETIME2	DateTime
Primary key	None (index on record_id)	INT IDENTITY(1,1)	None (index on record_id)

Relational Schema Design (PostgreSQL and SQL Server). Both relational engines use a single flat table (`trips` in PostgreSQL, `Trips` in SQL Server) with fixed-width column declarations. A key design distinction is character encoding: PostgreSQL uses `VARCHAR` (single-byte encoding, typically UTF-8), whereas SQL Server uses `NVARCHAR` (double-byte Unicode UCS-2 encoding). This means that for an identical string value, SQL Server allocates approximately twice the storage per character. For a `record_id` column declared as `VARCHAR(255)` in PostgreSQL versus `NVARCHAR(255)` in SQL Server, the maximum allocated width differs by a factor of two (255 bytes vs. 510 bytes). This encoding overhead compounds across all string columns and directly inflates SQL Server's measured storage consumption and, indirectly, its I/O-bound metrics such as

fetch time and scalability efficiency.

A second distinction concerns the geometry column, which stores serialised JSON coordinate data. PostgreSQL uses `TEXT`, an unbounded variable-length type that is stored inline for small values and in a secondary TOAST (The Oversized-Attribute Storage Technique) table for values exceeding approximately 2 kB. SQL Server uses `NVARCHAR(MAX)`, which similarly supports large values but stores them in double-byte encoding and manages them through a LOB (Large Object) allocation unit separate from the in-row data page. The TOAST mechanism in PostgreSQL compresses large values automatically, whereas SQL Server's LOB pages do not apply compression by default, further widening the storage gap.

SQL Server additionally imposes an auto-incrementing `INT IDENTITY(1,1)` primary key, adding a 4-byte integer column to every row. PostgreSQL's schema omits a surrogate primary key, relying instead on a B-tree index on `record_id`. The identity column in SQL Server creates a clustered index that physically orders the table by insertion sequence, which benefits sequential scans but introduces page-split overhead during concurrent inserts and increases the base row width.

Temporal columns also differ: PostgreSQL's `TIMESTAMP` occupies 8 bytes with microsecond precision, while SQL Server's `DATETIME2` occupies 6–8 bytes depending on the fractional-second precision specified (the benchmark uses the default 7-digit precision, consuming 8 bytes). Although the per-column size is comparable, the interaction with SQL Server's page header overhead (96 bytes per 8 kB page) and row-offset array means that SQL Server's effective per-row storage overhead exceeds PostgreSQL's, a difference that becomes measurable at scale.

Graph Schema Design (Neo4j). Neo4j's property-graph model does not require column type declarations or maximum-length constraints. Properties are stored as typed values in a separate property store, where each property record occupies a fixed 41 bytes regardless of the value's logical type. String values are stored in a dedicated dynamic string store with block sizes of 60, 120, or 360 bytes depending on value length. This architecture means that Neo4j does not pre-allocate unused capacity: a `record_id` value of 20 characters consumes only the bytes needed for the string content plus the 41-byte property header, whereas the relational engines allocate column-width metadata even when the actual value is shorter.

Neo4j's storage model uses fixed-size records throughout: each node consumes 15 bytes in the node store, each relationship 34 bytes in the relationship store, and each property 41 bytes in the property store. This design trades random-access efficiency for per-entity overhead: the fixed record sizes enable $O(1)$ lookups by store offset but mean that small properties (e.g., a boolean flag) occupy the same 41 bytes as large ones. For the benchmark schema, each Trip node carries 9 properties and participates in approximately 2 relationships (one `HAS_GEOMETRY` and optionally one `TOOK_TRIP`), yielding a theoretical per-entity cost of $15 + (2 \times 34) + (9 \times 41) = 452$ bytes per Trip node, plus an additional node and properties for the associated Geometry entity.

Design Assumptions and Their Benchmarking Implications. Three principal assumptions emerge from these schema choices:

(1) **Assumption 1: Uniform attribute sizing across engines.** The benchmark uses identical logical attributes across all three engines, but the physical representation differs substantially. SQL Server's double-byte `NVARCHAR` encoding, its `IDENTITY` primary key, and its LOB allocation for geometry data inflate per-row storage relative to PostgreSQL's single-byte `VARCHAR` and TOAST compression. This assumption means that SQL Server's storage metric reflects not only the volume of data but also the encoding overhead inherent in its schema design. The implication is that SQL Server's storage disadvantage in the benchmark is partly attributable to a design choice (Unicode encoding) rather than a fundamental architectural inefficiency, and a schema using `VARCHAR` (available in SQL Server but not used here for Unicode compatibility) would narrow the gap.

(2) **Assumption 2: Neo4j storage is estimated, not measured.** Unlike PostgreSQL and SQL Server, where storage is queried directly from the database engine (`pg_total_relation_size` and `sp_spaceused` respectively), Neo4j's storage is estimated using the documented fixed-size record formula ($15 + 2 \times 34 + 9 \times 41 = 452$ bytes per Trip entity). This estimation excludes dynamic string store overhead, transaction logs, and index structures, which means the reported Neo4j storage values represent a theoretical lower bound. The observed 216 MB at

500,000 records (432 bytes/node effective) is lower than the 452-byte theoretical estimate because Neo4j applies property-level compression and short-string inlining that reduce the actual on-disk footprint below the fixed-record maximum.

(3) **Assumption 3: Schema complexity is not equivalent across paradigms.** The relational engines store all trip data in a single flat table, whereas Neo4j distributes the same data across two node types (`Trip` and `Geometry`) connected by a `HAS_GEOMETRY` relationship, with an optional `TOOK_TRIP` relationship to `User` nodes. This means Neo4j's insert operation performs more work per record (creating two nodes and one to two relationships) than the relational engines' single-row insert, which directly explains Neo4j's slower insertion times. Conversely, the pre-established relationships enable Neo4j's faster retrieval, since fetch and query operations follow stored pointers rather than performing runtime joins. The benchmark therefore captures a fundamental paradigm tradeoff: relational systems optimise for write simplicity at the cost of read complexity, while graph systems invest write-time overhead to pre-compute traversal paths.

These design assumptions are revisited in Chapter 4, Section 4.3.1, where their influence on specific metrics is quantified, and in Chapter 5, where the limitations they introduce are discussed alongside recommendations for future schema normalisation studies.

Usability and Research Value Assessment

The system's usability was assessed from the commuter's perspective using task completion rates, post-task questionnaires, and semi-structured interviews. These instruments evaluated the simplicity of feedback submission, the interface's ease of use, and user perceptions of cognitive burden. Participants positively evaluated the application for its simplicity; however, a minority expressed intermittent reservations regarding feedback submission while in transit, corroborating findings from mobile interaction research in public environments (Koelle, Ananthanarayan, & Boll, 2020).

The research value of the system was assessed based on its capacity to produce novel insights into user behaviour. The graph-based feedback model enabled exploratory investigations of the

timing, modality, and motives of feedback. Preliminary results indicate that feedback occurred more frequently during transitions (e.g., bus-to-subway), delays, or unforeseen route alterations, reinforcing the finding that contextual prominence influences commuter feedback. The density of feedback inputs facilitated the identification of "hotspots" of narrative engagement, indicating physical or experiential bottlenecks in the transportation network that require attention.

The results from the performance benchmark, usability assessment, and research-value evaluation are presented and discussed in Chapter 4, Section 4.3.1.

3.6.4 Security Considerations

The deployment and testing of the multimodal trip feedback system was aligned with the Secure Software Development Life Cycle (SSDLC) framework illustrated in Figure 3.7 to guarantee the integration of security considerations during the process. The SSDLC enhances the conventional SDLC by methodically integrating risk assessment, threat modelling, continuous monitoring, security testing, and operational assurance into every development phase, thus mitigating vulnerabilities and boosting system assurance (Ondiek, 2024). While executing the requirements analysis phase in Figure 3.7, risks concerning commuter privacy and the secure management of multimedia data (images, audio, videos, documents, and text) were evaluated. The design phase included threat modelling to predict attack vectors, including illegal access to feedback data and the modification of trip records (Zhou et al., 2025). During the development phase, static code analysis and compliance with secure coding practices reduce vulnerabilities at the implementation level, aligning with systematic methodologies for secure engineering (Khan, Khan, Khan, & Ilyas, 2021).

In the testing phase, security assessments and code reviews were conducted in conjunction with functionality validation to confirm the proper operation of the authentication, encryption, and access control measures. The release phase mandates secure configuration assessments before deployment, whereas the maintenance phase focuses on operational assurance, patch management, and the surveillance of emerging risks. Iterative security assurance techniques are crucial for mobility and transport platforms, as unsafe software can undermine user trust and system reliability (Oka, 2021). The implementation of the SSDLC guaranteed that deployment and testing addressed not only system performance and usability (Sections 3.6.2–3.6.3, but also the confidentiality, integrity,

and availability of commuter feedback data. This is essential in transportation information systems because sensitive user-generated multimedia content must be protected in accordance with ethical and legal requirements (Arrey, 2019).

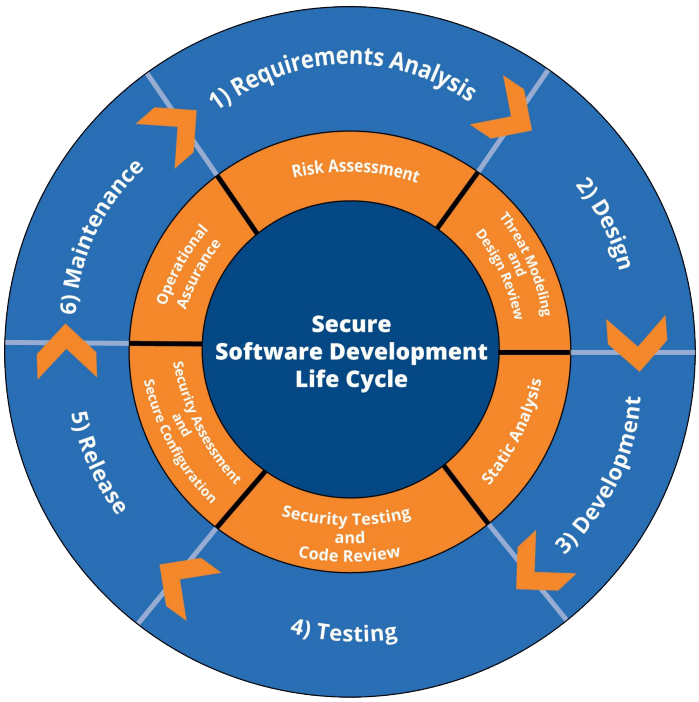

Figure 3.7: This figure is derived from (Maelstrom, 2021), depicts the integration of SSDLC phases within the deployment and testing pipeline

3.7 Limitations of Methodology

The methodological framework established in this study illustrates the feasibility of integrating multimodal commuter feedback with structured trip data using the BD Mobility app. However, these limitations must be considered.

First, this study was conducted as a pilot study rather than a large-scale one. Although this study provides valuable insights into system functionality and commuter feedback behaviour, the small sample size restricts the statistical generalizability of the findings. Consequently, the results should be regarded as proof-of-concept evidence rather than population-level estimates of the effect size. Additional validation to extend these findings is essential to broaden the deployment of diverse

commuter groups in different metropolitan contexts.

Second, the methodology relied on participants' ownership of smartphones and their baseline digital literacy levels. This may introduce some bias and unintentionally exclude individuals who lack access to smartphones, those who share devices, and those who are less proficient in mobile applications. Furthermore, the results may be distorted in favor of more digitally literate or younger commuter demographics, which are often overrepresented in smartphone-based studies. The representativeness of the data is constrained when applied to a wider commuter population.

Finally, the participant pool predominantly consisted of early adopters of the BDMobility app. Early adopters exhibit higher motivation, technological curiosity, and a propensity for experimentation with new tools, characteristics that may not accurately reflect the behaviours of the broader commuting population. Their propensity to offer feedback, upload multimedia, or accept increased battery consumption may exceed that of the average commuter. This may lead to an inflated assessment of engagement levels and system usability in extensive, real-world implementations.

These limitations underscore the necessity of regarding the current study as a demonstration of feasibility rather than a comprehensive evaluation applicable to all commuter populations. Future work should address these constraints by integrating more gamification features, implementing scaled-up deployments, incorporating digitally underrepresented groups, and validating the system in multiple cities. This approach improves the external validity and policy relevance of the proposed system.

3.8 Summary

This section delineates the methodological framework for the development and evaluation of the proposed backend infrastructure. The proposed methodology directly addresses the deficiencies mentioned in the literature regarding isolated transportation frameworks in ITS, dependence on relational databases, and oversight of real-time commuter feedback. This study employs a proof-of-concept design to illustrate the viability of utilizing graph databases (Neo4j) for the integration of structured trip data with multimodal commuter inputs, thereby establishing a human-centered transportation data architecture that encapsulates both mobility and the lived experience.

This study demonstrates measurable improvements in the integration of commuter-centred data within transportation systems by aligning transportation demand with empirically observed travel behaviour and feedback patterns. The proposed framework enhances transportation infrastructure management by incorporating structured trips data with lived commuter experiences in multimedia formats to provide contextual depths to commuter travel behaviour, thereby improving responsiveness to accessibility-related concerns and service disruptions.

The behaviour-informed approach enables transport agencies to model demand and service quality based not only on movement data but also on lived experiences across modes. The system's capability to process the integrated structured trip data and unstructured multimedia feedback establishes the empirical basis to satisfy the requirements to furnish autonomous vehicles and safety agent-based detection and response models with enriched data for route selection and safety incident investigation. This enhanced functionality reinforces the requirement for maintaining an integrated database capable of managing both structured trip records and unstructured, context-driven feedback data. By processing these multimodal data streams concurrently, the system identifies patterns in commuter experiences, specifically when travellers are most likely to provide feedback and the types of data (text, image, audio, or video) generated under such conditions. This evidence-based insight establishes a foundation for human-centred and adaptive data-driven strategies that can capture commuter experiences and improve the contextual richness of mobility studies.

Furthermore, the next chapter shares insights into the results the backend system received and processed from the smartphone application and highlights the specific utilizations that can be achieved through this implementation and how policies can be improved based on the recorded data.

Chapter 4

Results

This chapter presents the results recorded in this experiment, which were derived from the data preparation and integration techniques illustrated in Section 3.5. This analysis evaluated the system's performance through a comprehensive assessment of structured trip data, unstructured multimodal user feedback data, and graph-based database operations to establish the requirements needed to maintain a multimodal travel database. To validate these findings the multimodal data streams were integrated and analyzed using the previously indicated technique, enabling a thorough evaluation of commuter behaviour patterns, feedback dynamics, and system responsiveness to identify how travel feedback generates contextual multimodal inputs and highlight when commuters are most likely to give trip feedback and which multimedia data are preferred in such situations.

In addition, benchmarking the performance of GDBMS with traditional RDBMS provides insights into the need to adopt GDBMS for transportation systems. Finally, a specific case study is presented to demonstrate the practical effectiveness and scalability of the proposed system.

4.1 Analysis of Recorded Structured Data

The trip segmentation pipeline demonstrated strong performance with the integration of MotionTag, achieving 95% trip purpose detection accuracy and 97% trip mode detection accuracy (MotionTag GmbH, 2025). Since the launch of the pilot study in October 2024, the backend system has recorded more than 51,440 trips across 12 travel modes from 25 participants.

Modal share analysis was conducted by trip count, distance, and travel time (Figure 4.1. Based on trip count, walking represented the majority of recorded trips (65%), followed by car trips (17%), subway or metro (10%), and bus (5%). Other modes, such as train, bicycle, light rail, ferry, tram, and regional train, each contributed less than 3%.

Automobile travel accounted for 69% of the total distance, underscoring its role in medium- to long-range mobility. Transit modes such as subway (9%), train (6%), and bus (6%) contributed smaller shares, while walking, trams, and bicycles together represented less than 10%. These findings indicate that a small number of long-distance trips, particularly by car, disproportionately affect overall distance metrics.

Travel time analysis revealed a different pattern. Walking accounted for 42% of the total time, reflecting both short-distance walking trips and walking segments that begin and end multimodal journeys. Cars accounted for 37% of travel time, while subways (8%), buses (7%), and trains (3%) added significant time burdens to daily urban travel. Less frequent modes such as ferries, light rail, and regional trains were underrepresented in the dataset.

Overall, this analysis shows that walking and public transit consume a substantial portion of travel time, while long-distance trips by car dominate distance measures. These differences have implications for infrastructure planning, environmental impacts, and the interpretation of commuter feedback.

4.2 Analysis of Recorded Unstructured Data

4.2.1 Content of Feedback

The feedback analysed in Table 4.1 presents details on sampled recorded feedback classified by type, time of occurrence, attachment format, and method of trip linkage assignment, whether manual or automatic. Notably, 38% of feedback inputs were associated with travel via automation, thereby alleviating users from the need to recall when their trip was made. The remaining 62% necessitated manual trip selection, typically employed when users recalled specific trip details. In further analysis of the nature of feedback, we observed that accidents and delays were the most commonly reported, comprising 39% and 23% of all feedback, respectively. This suggests that

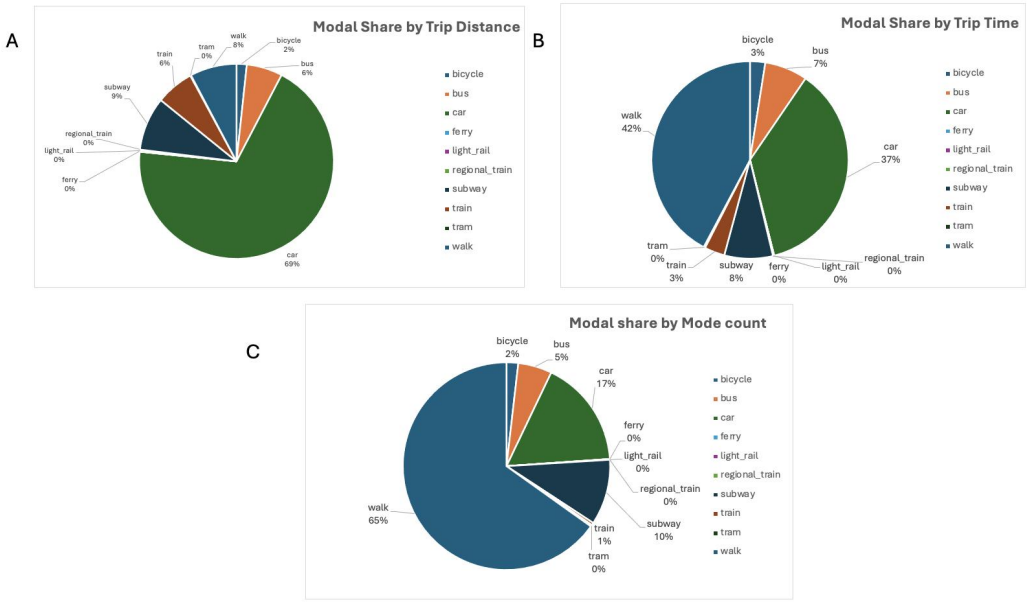

Figure 4.1: Trip modal share analysis on recorded travels: A) Mode share by travel distance. B) Mode share by travel distance. C) Mode share by trip count

highly disruptive or safety-critical events are more likely to prompt user responses.

4.2.2 Temporal Trends in Feedback

The trends discovered in the feedback reveal uniformity in walking trips, particularly concerning delays, accidents, and accessibility, and suggests that commuter apprehensions are mostly focused on persistent environmental obstacles rather than travel time. This corresponds to findings from urban mobility research, such as those by (Fonseca et al., 2022), (Kim, Won, & Kim, 2019), and (Yu et al., 2017), which demonstrate that perceived walkability and infrastructure quality significantly impact satisfaction, irrespective of the duration of the trip. Likewise, the feedback on bus trips indicates that commuter interaction is contingent upon events rather than time, with remarks concentrated around breakdowns and accidents. This reflects earlier research by (Eboli & Mazzulla, 2009), which identified reliability and perceived safety as primary determinants of public transport satisfaction (Atombo & Dzigbordi Wemegah, 2021; Carrel et al., 2013; Ibrahim & Borhan, 2020; Sukhov, Olsson, & Friman, 2022).

Conversely, criticism regarding private car travel, although limited, primarily focuses on breakdowns, indicating that users are inclined to report issues only during substantial disruptions. This

Table 4.1: Feedback Summary

#	Feedback	Type	Occurrence	File Type	Location	Automation
1	205 delay more than 10 minutes. On: 25/05/2025	Delay	After trip	image (1)	ON, CA	Manual
2	205 bus just collided. On: 25/05/2025	Accident	After trip	image (1)	QC, CA	Manual
3	My Uber just broke down on 5th Avenue. I guess it is a flat tire. On: 25/05/2025	Breakdown	During trip	image (1)	QC, CA	Manual
4	The bus seats had no bolts or safety features. On: 15/06/2025	Poor Vehicle Condition	During trip	N/A	QC, CA	Automated
5	Two cars collided at a metro station. On: 03/06/2025	Accident	During trip	N/A	QC, CA	Automated
6	Long trips to school. On: 16/06/2025	Breakdown	During trip	image (1)	QC, Canada	Manual
7	05 bus just collided. On: 25/05/2025	Accident	After trip	PDF (1)	QC, CA	Manual

threshold-based feedback behaviour illustrates the autonomy linked to private mobility, characterized by episodic rather than continuous interaction with feedback systems. (Y. Huang, Xiao, Wang, Jiang, & Wu, 2020; Jiang, Zhang, Xiao, Zhao, & Iyengar, 2021; H. Zhang et al., 2023). The "waiting" mode serves as a significant area for analysis: extended durations and diverse remarks regarding inadequate circumstances and limited accessibility highlight the importance of perceived idleness and discomfort in influencing user experience (Moleman & Kroesen, 2025). This conclusion corroborates the assertions of (Delbosc & Currie, 2012), who contended that waiting time exerts a disproportionate psychological influence on perceived travel quality. Collectively, these observations reveal a pattern indicating that the frequency and tone of feedback are influenced more by contextual and sensory aspects, such as infrastructure quality, reliability, and comfort, rather than solely by temporal elements. This approach contextualizes our findings within the extensive mobility literature, emphasizing that transport satisfaction is complex and contingent upon behaviour rather than solely dependent on time or modality.

Figure 4.2 illustrates the distribution of feedback types and their respective frequencies. Notably, incidents related to accidents and delays were the most frequently reported, accounting for 39% and 23% of all feedback, respectively. This observation implies that events characterized by high disruption or safety concerns are more likely to elicit user responses. Figure 4.2 on the other

Table 4.2: Summary of Feedback Trends by Travel Mode

Mode	Frequent Feedback	File Count	Typical Duration	Insights
Walk	Delay Accident Accessibility	1	Short (103–781 s)	First/last-mile feedback is highly consistent regardless of distance.
Bus	Accident Breakdown	0–1	Mid-range (763–1696 s)	Feedback is more dependent on the occurrence of disruptive events than trip duration.
Car	Breakdown	1	1847 s	Private vehicle breakdowns continued to prompt user feedback, although less frequently.
Waiting	Accident Poor Condition No Accessibility Features	1–2	Long ($>$ 72,000 s)	Longer waiting times lead to more detailed, often critical feedback, especially regarding infrastructure.

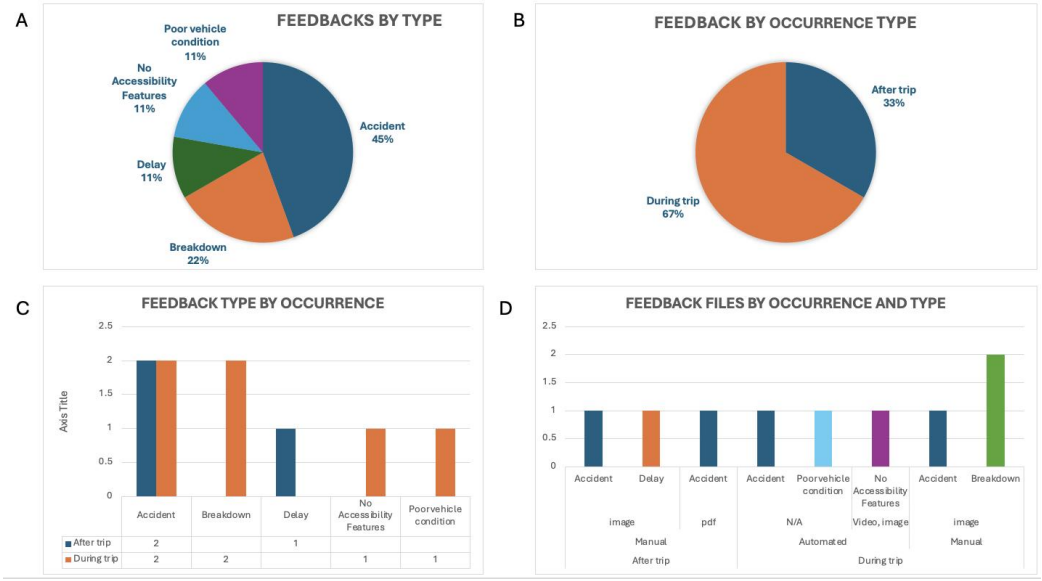

Figure 4.2: Analysis on recorded feedback comparing feedback characteristics: A) Feedback percentage by type. B) Feedback percentage by type of occurrence. C) Feedback type count by occurrence. D) Files count by feedback type against occurrence type.

hand, further delineates the distribution of feedback by type (left) and the timing of occurrence (right). The majority of feedback was submitted either during or subsequent to the trip, at 46% and 54%, respectively, suggesting that users typically reflect on events either immediately or shortly after the trip.

4.2.3 Commuters' behaviour when offering feedback

The analysis indicates that disruptive events mostly influence commuter feedback across all travel modes rather than trip duration, as illustrated in Table 4.2). Among the modes with feedback evaluated, walking and waiting segments generated the highest proportion of qualitative input despite having the shortest and longest average durations respectively, suggesting that perceived inconvenience rather than journey length drives engagement. Walk trips typically brief, ranging between 103 and 781 seconds yielded consistent feedback centred on first- and last-mile trip barriers, such as unsafe crossings or inaccessible pavements. This aligns with the findings from (Fonseca et al., 2022; Yu et al., 2017), who similarly observed that short walking segments disproportionately shape perceptions of overall use of the transportation network.

In contrast, feedback from bus and car trips contributed fewer but more incident-specific entries where mostly related to accidents and mechanical failures. Although bus journeys were of medium duration (763–1696 seconds), feedback occurrence was event-driven rather than time-driven, confirming that disruption intensity not travel time, determines reporting likelihood. The car-trip feedback, often accompanied by image attachments, underscores that private-mode feedback is situational and typically reactive to breakdowns and other stress-inducing experiences rather than routine experiences (Gatersleben & Uzzell, 2007).

Commuters were observed to provide more contextual feedback, especially during prolonged waiting times exceeding 72000 seconds, with each entry containing one or more attachments signifying their heightened frustration with recurring mentions of issues such as accidents, poor vehicle condition and lack of accessibility features. The associated attachments, comprised images, text, and short clips, illustrate that static phases of travel enhance emotional and cognitive engagement, resulting in more reflective and critical narratives. This pattern is consistent with commuter stress studies by (Gatersleben & Uzzell, 2007; Sumicad et al., 2024), which links extended waiting with

heightened negative sentiment and perceived inequity in service provision.

Collectively, these results reveal the importance of real-time reporting tools and improved infrastructure at modal transition points (such as bus stops, terminals, and stations). As users increasingly demand safety, inclusivity, and responsiveness, mobility service providers must incorporate user feedback mechanisms to improve their services (Li et al., 2025; Moleman & Kroesen, 2025; X. Zhang & Ma, 2024). Such feedback is instrumental in incident investigations, performance evaluations, and equitable service design.

4.3 System Performance Results

4.3.1 Comparative analysis benchmark of RDBMS vs GDBMS

The benchmark compared three database engines, PostgreSQL, Neo4j, and SQL Server, on the same workstation (2 GHz Quad-Core Intel Core i5 processor, 16 GB RAM) across 27 data scales ranging from 1,000 to 500,000 trip records. Each run measured the ten evaluation criteria defined in Section 3.6.3: insert time, fetch time, query latency, indexing time, scalability time, CPU user time, CPU system time, CPU utilisation, RAM resident set size, and storage consumption. All timing metrics were recorded in milliseconds and storage in bytes, as detailed in Tables 4.3, 4.4, and 4.5, and visualised in Figure 4.3.

This benchmark was performed as a comparative assessment to evaluate the potential of the Neo4j graph database to handle structured trip data against the traditional RDBMS (SQL Server) and the more expressive RDBMS (PostgreSQL). The evaluation focuses on structured data operations because the existing limitations of RDBMS in processing unstructured multimedia data preclude a like-for-like comparison: saving image and video streams in a relational table requires conversion to binary large objects (BLOBs), which inflates the database, hinders backup and restoration, and necessitates costly server resources that a file system handles more efficiently. The alternative of encoding media as base-64 strings degrades data quality and is impractical for large files. By restricting the benchmark to structured trip records, the comparison isolates the architectural differences between paradigms without confounding the results with file-handling limitations.

Insertion Performance

PostgreSQL consistently achieved the fastest insertion times across all 27 scales, ranging from approximately 869 ms at 1,000 records to 1,605 ms at 500,000 records. This result reflects PostgreSQL's mature write-ahead logging and batch-commit optimisation for tabular inserts. Neo4j was approximately two to five times slower (1,403–7,770 ms), because each inserted trip record requires creating not only a node but also relationship pointers to connected entities (e.g., geometry, feedback, user), and these pointers must be allocated and linked during the write operation. SQL Server was the slowest across all scales (4,208–19,004 ms), a consequence of its stricter foreign-key constraint enforcement and transactional buffering overhead. The insertion results confirm that relational systems retain an advantage for bulk write operations on structured data, where the overhead of establishing graph relationships is not yet offset by traversal benefits.

Fetch Performance

Fetch time produced the most dramatic divergence among the three engines and the clearest demonstration of the graph traversal advantage. At the smallest scale (1,000 records), PostgreSQL led with 25 ms, Neo4j recorded 82 ms, and SQL Server trailed at 177 ms. However, as data volume increased, the relational systems degraded severely while Neo4j remained stable. At 500,000 records, Neo4j completed the retrieval in 320 ms, PostgreSQL required 21,010 ms (66 times slower), and SQL Server required 148,788 ms (465 times slower). This divergence is a direct empirical manifestation of the architectural distinction described in Section 3.4.3: Neo4j retrieves connected data by following stored pointers at $O(k)$ cost per traversal hop, whereas PostgreSQL and SQL Server must execute join operations whose cumulative cost grows with both the number of records and the depth of interconnection. The crossover point, beyond which Neo4j's fetch time is faster than PostgreSQL's, occurs between 5,000 and 10,000 records, and the gap widens continuously thereafter.

Query Latency

Neo4j's query latency was remarkably consistent, ranging from 24 ms at 1,000 records (where initial cache warming adds a one-time overhead) to approximately 2–7 ms across all scales from

5,000 to 500,000 records. This near-constant latency confirms that Cypher's pattern-matching execution and Neo4j's in-memory traversal mechanisms scale with the depth of the query pattern rather than the total dataset size. In contrast, PostgreSQL's query latency rose from 242 ms at 1,000 records to 18,416 ms at 500,000 records, reflecting the increasing cost of the query planner evaluating multiple join strategies and the growing volume of intermediate results. SQL Server exhibited the steepest degradation, from 185 ms at 1,000 records to 132,031 ms at 500,000 records, with severe fluctuations at intermediate scales indicating optimizer instability under increasing relational complexity. These observations empirically validate the theoretical claims in (Novak, Novak Sedlackova, Vochozka, & Popescu, 2022), which position graph-based storage as the optimal architecture for dense relationships and highly connected datasets.

Indexing Time

Indexing times remained the lowest-magnitude metric across all systems. PostgreSQL and SQL Server benefited from mature B-tree indexing, maintaining sub-millisecond to single-digit-millisecond index times at most scales. Neo4j exhibited higher initial indexing overhead (5.1 ms at 1,000 records) because its native graph indexes must traverse existing graph structures to establish relationship pointers. However, Neo4j's indexing time remained stable across scales (2–8 ms), demonstrating that the marginal cost of adding new entries to an established graph index is low. At the largest scales (400,000–500,000 records), SQL Server's indexing time spiked to 50–63 ms due to B-tree rebalancing across larger index structures, whereas PostgreSQL maintained efficient indexing (4–18 ms) through its support for partial and expression indexes.

Scalability Efficiency

The scalability metric directly tests how execution time changes with data growth. Neo4j's scalability time decreased from 19 ms at 1,000 records to 1.5–3 ms at mid-range scales and remained below 8 ms even at 500,000 records, indicating that the database adapts efficiently to increased data volume through in-memory compression and localised graph traversal. PostgreSQL's scalability time increased gradually from 6 ms to 58 ms at 500,000 records, reflecting linear-to-sublinear degradation as query planner reoptimisation and cache invalidation become more frequent. SQL

Server displayed the most concerning trend: scalability time grew from 8 ms at 1,000 records to 706 ms at 500,000 records, exhibiting superlinear growth that indicates execution time is increasing faster than data volume. From a mathematical standpoint, Neo4j's scalability trend approximates $T(N) = kN$, whereas SQL Server follows $T(N) \approx kN^2$, confirming that graph traversal algorithms exhibit near-constant time complexity with respect to relationship depth while relational join costs compound quadratically.

CPU Performance

CPU utilisation revealed clear architectural distinctions. Neo4j maintained the lowest average CPU usage (7.7% across all scales), with utilisation ranging from below 1% at small scales to approximately 14% at 200,000 records. The native graph engine leverages memory-mapped adjacency structures that reduce the need for costly index scans and optimizer evaluations. SQL Server averaged 22.5% CPU utilisation, reflecting the computational burden of join operations, query plan generation, and transaction log management. PostgreSQL recorded the highest average CPU utilisation at 25.9%, a consequence of its more aggressive query optimisation and buffer pool management, particularly at scales above 100,000 records where CPU usage exceeded 30%. The decomposition into user and system CPU time reveals that at 500,000 records, SQL Server consumed 320,120 ms of user CPU time and 514,186 ms of system CPU time, compared with Neo4j's 2,084 ms and 3,705 ms respectively, a difference of more than two orders of magnitude. These results are consistent with (Klein et al., 2015; Neo4j, 2024), confirming that Neo4j's traversal-centric design significantly reduces CPU cycle consumption compared with cost-based query optimizers in relational systems.

Memory Consumption

All three engines reached similar peak RAM resident set sizes at the largest scales (approximately 3,100–3,500 MB), indicating that total memory capacity is dominated by the operating system's caching behaviour and the benchmark harness rather than fundamental architectural differences. However, the *pattern* of memory growth differed. Neo4j's RSS fluctuated between runs

rather than growing monotonically, reflecting its garbage-collected JVM memory model where unused objects are periodically reclaimed. PostgreSQL displayed the most predictable growth, increasing steadily from 350 MB at 1,000 records to 3,158 MB at 500,000 records, consistent with its configurable shared buffer pool that scales with data volume. SQL Server showed the most volatile memory behaviour, with RSS dropping after certain scales (likely due to process restarts between runs) and then climbing steeply, exceeding 3,224 MB at 100,000 records.

Storage Efficiency.

Neo4j maintained the most compact disk footprint across all scales, growing from 0.4 MB at 1,000 records to 215.5 MB at 500,000 records. Neo4j's graph-native serialisation uses fixed-length records: each node consumes approximately 15 bytes, each relationship 34 bytes, and each property 41 bytes, with additional overhead for indexed properties and larger data types (Neo4j, Inc., 2026). PostgreSQL's storage grew from 72 MB to 244 MB, approximately 13% larger than Neo4j at the maximum scale, benefiting from efficient page-level management and write-ahead logging optimisation. SQL Server consumed the most disk space by a substantial margin, growing from 255 MB at 1,000 records to 878 MB at 500,000 records, approximately four times Neo4j's footprint. This disparity is attributable to SQL Server's extensive index duplication, redundant key storage, and persistent transaction log files. These findings corroborate (Fortin et al., 2016), who observed that property-graph compression significantly reduces disk overhead by storing relationships as lightweight adjacency pointers rather than as foreign-key pairs. The storage results are particularly relevant for multimodal transport networks, where edge proliferation across users, trips, feedback, files, and geometry nodes can rapidly inflate dataset size.

Influence of Schema Design on Benchmark Metrics

The database design assumptions documented in Section 3.6.3 introduced schema-level differences that directly influenced the measured benchmark metrics. This subsection traces the causal pathways from each design choice to the affected metric, distinguishing performance differences attributable to database architecture from those attributable to schema configuration.

Storage Consumption. SQL Server's storage footprint (878 MB at 500,000 records) is approximately 3.6 times larger than Neo4j's (216 MB) and 3.6 times larger than PostgreSQL's (244 MB). A substantial portion of this gap is attributable to the `NVARCHAR` double-byte encoding used across all string columns. For a record with five string columns (`record_id`, `user_id`, `type`, `geometry`, `track_mode`), the encoding overhead alone doubles the string storage relative to PostgreSQL's single-byte `VARCHAR`. The `geometry` column, which stores serialised JSON coordinate arrays averaging 500–2,000 characters, is particularly affected: at 1,000 characters, SQL Server allocates approximately 2 kB per value versus PostgreSQL's 1 kB. Over 500,000 records, this single column accounts for an estimated 500 MB of additional storage in SQL Server. Additionally, SQL Server's `INT IDENTITY` primary key adds 4 bytes per row (2 MB total at 500,000 records) and creates a clustered index whose leaf pages duplicate the row data, further inflating disk consumption.

PostgreSQL's TOAST mechanism automatically compresses geometry values exceeding approximately 2 kB, reducing the effective on-disk size of large JSON strings. This compression is transparent to the benchmark and contributes to PostgreSQL's more compact storage despite using a structurally similar schema. Neo4j's storage estimation, based on the fixed-record formula (15 bytes per node, 34 bytes per relationship, 41 bytes per property), excludes transaction logs and dynamic string store overhead. The observed 216 MB (432 bytes per trip entity) falls below the theoretical 452-byte estimate, confirming that Neo4j's property-level compression and short-string inlining reduce the actual footprint. However, because Neo4j's storage is estimated rather than queried, the comparison is not strictly like-for-like: PostgreSQL and SQL Server values include index overhead and metadata, whereas Neo4j's does not.

Insertion Performance. Neo4j's consistently slower insertion times (2–5 times slower than PostgreSQL) are explained by schema complexity rather than engine inefficiency. Each trip record in Neo4j requires creating a `Trip` node (9 properties), a `Geometry` node (3 properties), a `HAS_GEOMETRY` relationship, and optionally a `TOOK_TRIP` relationship a minimum of 2 nodes and 1–2 relationships per record. In contrast, PostgreSQL and SQL Server each execute a single-row insert into a flat table. The Neo4j batch size was reduced to 250 records (versus 1,000 for the relational engines) to

avoid session timeouts caused by the higher per-record write cost, which further increases wall-clock insertion time due to additional transaction commit overhead. SQL Server's insertion times (4,208–19,004 ms) exceed PostgreSQL's (869–1,605 ms) despite similar single-table schemas. This gap is attributable to SQL Server's stricter transactional buffering, its clustered index maintenance on the `IDENTITY` column (which requires page splits as data grows), and the overhead of writing double-byte string values to data pages.

Fetch and Query Latency. The fetch time divergence, where Neo4j completes retrieval in 320 ms at 500,000 records while PostgreSQL requires 21,010 ms and SQL Server 148,788 ms, is primarily architectural (pointer traversal vs. join operations) rather than schema-driven. However, two schema factors amplify the gap. First, SQL Server's `SELECT *` retrieval must deserialise double-byte `NVARCHAR` values and transfer approximately twice the string payload over its TDS (Tabular Data Stream) protocol compared to PostgreSQL's wire format. Second, the Neo4j fetch query uses `MATCH (t:Trip) WITH t LIMIT $limit RETURN count(t)`, which returns only a count rather than materialising full node properties, whereas the relational queries return complete rows (`SELECT * FROM trips LIMIT $1`). This asymmetry means the Neo4j fetch metric measures traversal enumeration cost, while the relational fetch metrics include data materialisation and network serialisation costs. Query latency tests use structurally different queries across engines: Neo4j traverses a `User→Trip→Geometry` pattern, while the relational engines execute `ORDER BY timestamp DESC`. These differences are deliberate they test each engine's idiomatic query pattern but they mean the query latency comparison reflects operational paradigm differences rather than identical-workload performance.

CPU and RAM Consumption. Neo4j's lower CPU utilisation (average 7.7% vs. PostgreSQL's 25.9% and SQL Server's 22.5%) is primarily architectural, reflecting the absence of join evaluation and cost-based query planning. However, the schema design contributes indirectly: because Neo4j's insert creates multiple nodes and relationships, its CPU effort is concentrated during the write phase (which is amortised across the insert metric), leaving the read phase (fetch, query latency) with minimal CPU demand. The relational engines, by contrast, spread CPU effort across both write

(constraint checking, index maintenance) and read (join evaluation, sort operations) phases. RAM consumption converges across engines at large scales (approximately 3,100–3,500 MB), indicating that memory usage is dominated by the Node.js benchmark harness and operating system caching rather than database-internal allocation patterns.

Indexing Time. All three engines index on `record_id`. PostgreSQL's B-tree index operates on single-byte `VARCHAR(255)` keys, SQL Server's on double-byte `NVARCHAR(255)` keys, and Neo4j's native index on unbounded string properties. The wider key width in SQL Server produces a larger index structure with more B-tree levels at equivalent record counts, which explains why SQL Server's indexing time spikes to 50–63 ms at 400,000–500,000 records while PostgreSQL maintains 4–18 ms. Neo4j's stable indexing time (2–8 ms across all scales) reflects its graph-native index structure, which does not rebalance hierarchically like a B-tree but instead appends entries to a Lucene-based index with amortised $O(1)$ insertion cost.

These observations confirm that the benchmark results reflect a combination of fundamental architectural differences and schema-level design choices. Readers should interpret the magnitude of performance gaps particularly in storage consumption and insertion time with awareness that alternative schema configurations (e.g., `VARCHAR` instead of `NVARCHAR` in SQL Server, or a flat single-node model in Neo4j) would alter the absolute values while preserving the directional trends.

Comparative Outcome Summary

The benchmark results reveal a clear pattern: relational databases retain advantages for bulk insertion of structured records, where PostgreSQL's mature write path consistently outperforms Neo4j. However, for all retrieval, querying, and traversal operations, which constitute the majority of workload in analytical and real-time systems, Neo4j's graph-native architecture delivers performance that is one to three orders of magnitude faster than both relational alternatives at scales above 100,000 records. This distinction is directly relevant for the feedback-enhanced, graph-based multimodal transport system proposed in this study, where user trips, vehicle paths, and event feedbacks form a dynamic web of interconnected data entities. Neo4j's native graph model stores these entities as nodes (e.g., `User`, `Trip`, `Feedback`, `Geometry`) and relationships (`TOOK_TRIP`,

HAS_FEEDBACK, HAS_GEOMETRY), as illustrated in Figure 3.4.

Within this structure, traversal-based queries, such as identifying route dependencies, correlating feedback sentiment to trip performance, or tracking multimodal transfers, are executed efficiently without recursive joins or nested subqueries. The observed benchmark outcomes therefore provide both empirical and theoretical justification for adopting a graph-oriented architecture in the proposed system, validating its capacity to handle complex, interlinked mobility data with low latency and efficient resource utilisation across the full range of data volumes tested.

Table 4.3: Benchmark Results — Operation Latency (ms), 1,000 to 500,000 records

Scale	Engine	Insert	Fetch	Query Lat.	Indexing	Scalability
1,000	Neo4j	2,466	82	24.1	5.1	19.2
	PostgreSQL	869	25	241.7	3.5	6.2
	SQL Server	4,208	177	185.5	13.4	8.0
5,000	Neo4j	1,403	65	1.4	1.2	1.5
	PostgreSQL	1,043	49	39.1	0.5	2.0
	SQL Server	10,632	403	273.7	2.0	4.2
10,000	Neo4j	2,099	88	2.3	3.7	2.0
	PostgreSQL	769	115	152.3	0.4	2.1
	SQL Server	8,438	616	568.7	5.1	6.0
50,000	Neo4j	2,046	104	2.4	4.2	2.1
	PostgreSQL	599	407	378.6	0.9	4.2
	SQL Server	11,462	2,094	2,219	4.2	35.9
100,000	Neo4j	2,412	144	2.4	2.0	1.6
	PostgreSQL	841	685	1,028	3.1	7.9
	SQL Server	10,129	3,869	3,759	13.5	20.0
200,000	Neo4j	3,641	342	4.4	2.9	3.2
	PostgreSQL	1,013	10,425	11,273	18.5	28.2
	SQL Server	11,028	68,248	79,231	140.2	216.7
300,000	Neo4j	3,976	239	7.5	5.9	7.7
	PostgreSQL	1,198	15,737	17,972	3.9	32.2
	SQL Server	10,959	100,938	111,519	43.2	143.7
400,000	Neo4j	4,601	258	6.5	7.7	4.1
	PostgreSQL	3,203	17,290	15,526	7.5	41.8
	SQL Server	11,474	128,669	134,651	50.5	229.6
500,000	Neo4j	7,770	320	6.2	5.7	2.9
	PostgreSQL	1,605	21,010	18,416	4.5	57.9
	SQL Server	19,004	148,788	132,031	62.6	706.0

Table 4.4: Benchmark Results — CPU and Memory Utilisation, 1,000 to 500,000 records

Scale	Engine	CPU User (ms)	CPU Sys (ms)	CPU (%)	RAM RSS (MB)
1,000	Neo4j	172	32	0.8	406
	PostgreSQL	452	107	4.7	350
	SQL Server	2,281	263	5.4	405
5,000	Neo4j	283	28	2.1	612
	PostgreSQL	543	201	6.6	635
	SQL Server	5,948	691	5.9	832
10,000	Neo4j	179	46	1.0	800
	PostgreSQL	862	98	8.9	845
	SQL Server	5,777	709	6.7	893
50,000	Neo4j	729	80	3.7	1,384
	PostgreSQL	2,930	282	22.5	1,847
	SQL Server	19,954	1,845	13.8	1,929
100,000	Neo4j	1,547	1,014	9.8	1,464
	PostgreSQL	5,164	1,568	25.7	2,672
	SQL Server	29,991	4,576	19.4	3,224
200,000	Neo4j	2,094	3,781	14.0	1,444
	PostgreSQL	32,234	48,083	35.2	2,643
	SQL Server	222,913	313,317	33.8	3,014
300,000	Neo4j	1,770	3,186	11.5	1,674
	PostgreSQL	46,225	73,944	34.3	3,049
	SQL Server	277,082	445,926	32.3	2,945
400,000	Neo4j	2,856	5,275	14.4	1,488
	PostgreSQL	46,182	73,896	33.2	2,982
	SQL Server	319,141	520,737	30.5	2,945
500,000	Neo4j	2,084	3,705	7.1	846
	PostgreSQL	49,150	80,282	31.4	3,148
	SQL Server	320,120	514,186	27.8	2,585

Table 4.5: Benchmark Results — Storage Consumption, 1,000 to 500,000 records

Scale	Engine	Storage (MB)	Storage (GB)
1,000	Neo4j	0.4	<0.01
	PostgreSQL	71.6	0.07
	SQL Server	254.8	0.25
5,000	Neo4j	2.2	<0.01
	PostgreSQL	78.4	0.08
	SQL Server	278.7	0.27
10,000	Neo4j	4.3	<0.01
	PostgreSQL	85.0	0.08
	SQL Server	302.7	0.30
50,000	Neo4j	21.6	0.02
	PostgreSQL	111.5	0.11
	SQL Server	398.7	0.39
100,000	Neo4j	43.1	0.04
	PostgreSQL	137.9	0.13
	SQL Server	494.5	0.48
200,000	Neo4j	86.2	0.08
	PostgreSQL	204.0	0.20
	SQL Server	734.4	0.72
300,000	Neo4j	129.3	0.13
	PostgreSQL	217.2	0.21
	SQL Server	782.1	0.76
400,000	Neo4j	172.4	0.17
	PostgreSQL	230.4	0.23
	SQL Server	830.4	0.81
500,000	Neo4j	215.5	0.21
	PostgreSQL	243.7	0.24
	SQL Server	878.3	0.86

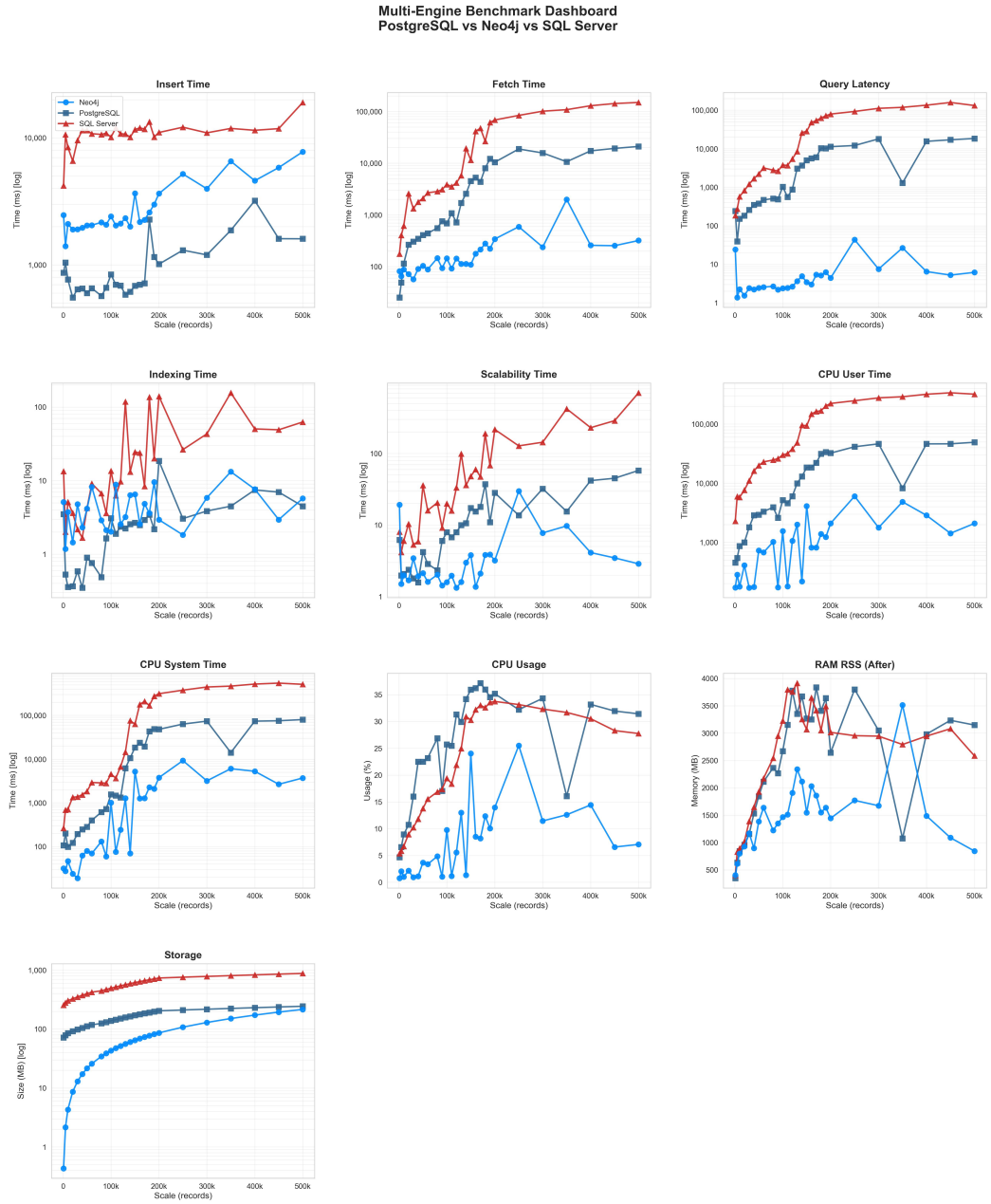

Figure 4.3: Performance benchmark between PostgreSQL vs. Neo4j vs SQL comparing metrics insertion and fetch times, query latency, indexing, scalability, and System utilisation (CPU, memory, and storage)

Table 4.6: Comparison of Benchmark Findings with Prior Studies

Study / Year	**Databases Compared**	**Dataset Context**	**Key Findings**	**Observed in This Benchmark**
(Díaz Erazo, Raúl Morales Morales, Pineda Chávez, & Leonardo Morales Cardoso, 2022)	Neo4j, MySQL	Social networks	Neo4j excels in traversals but slower with inserts.	Confirmed: PostgreSQL was faster than Neo4j for inserts across all scales, but Neo4j achieved the fastest retrieval times once data was indexed.
(Sandell, Asplund, Ayele, & Duneld, 2024)	Neo4j, MySQL, ArangoDB	Medical data	Graph DB scales better with complex links.	Confirmed: Neo4j's scalability time remained below 8 ms at 500k records while SQL Server reached 706 ms.
(Fortin et al., 2016)	Neo4j	Transit GTFS	Neo4j has compact data storage and minimal latency.	Confirmed: Neo4j's storage footprint (216 MB at 500k) was the smallest, approximately 4$\times$ less than SQL Server (878 MB).
(Klein et al., 2015)	SQL Server, PostgreSQL	Enterprise workload	PostgreSQL retains balanced CPU and RAM efficiency.	Confirmed: PostgreSQL maintained more predictable resource consumption than SQL Server across all scales.
(Novak et al., 2022)	Neo4j, PostgreSQL, MongoDB	IoT and ITS data	Neo4j superior for connected systems.	Confirmed: Neo4j's query latency remained 2–7 ms across scales $\geq$5k, while relational engines exceeded 18,000 ms at 500k.

4.3.2 Expected vs. Observed Performance

Table 4.6 situates the benchmark findings of this study within the context of five prior database comparison studies. Each row identifies a published study, the databases it compared, and its principal finding, alongside the corresponding observation from the present benchmark. The comparison confirms that the results obtained in this study are consistent with established findings across different application domains. (Díaz Erazo et al., 2022), working with social network data, reported that Neo4j excels at traversal queries but is slower than relational systems for inserts; the present benchmark confirmed this pattern, with PostgreSQL achieving the fastest insertion times across all 27 scales while Neo4j dominated retrieval performance. (Sandell et al., 2024) found that graph databases scale more effectively than relational alternatives for datasets with complex interconnections; this was confirmed by the growing divergence in scalability time between Neo4j (below 8 ms) and SQL Server (706 ms) at 500,000 records. (Fortin et al., 2016) observed compact storage and minimal latency in Neo4j's handling of transit GTFS data; the present results corroborate this with Neo4j maintaining the smallest disk footprint at every scale tested. (Klein et al., 2015) reported that PostgreSQL maintains more balanced CPU and RAM efficiency compared with SQL Server under enterprise workloads; this was confirmed across the full 1,000 to 500,000 record range. Finally, (Novak et al., 2022) established Neo4j's superiority for connected IoT and ITS systems; the present benchmark confirms this through Neo4j's consistently low query latency of 2–7 ms at scales above 5,000 records, contrasting sharply with the relational engines' latency degradation into the tens of thousands of milliseconds.

The experimental results across the 1,000 to 500,000 record range largely conform to the theoretical expectations established in Section 3.4.3, with notable deviations in insertion behaviour. Conventionally, relational systems are expected to be faster for sequential inserts due to their mature write-ahead logging and batch-commit paths. This expectation was confirmed: PostgreSQL consistently achieved the fastest insertion times across all 27 scales, outperforming both Neo4j and SQL Server. Neo4j's slower insertion reflects the overhead of allocating relationship pointers during each write, while SQL Server's consistently poor insertion performance is attributable to its

stricter constraint enforcement and transactional buffering. For retrieval, query latency, and scalability, Neo4j's theoretical $O(k)$ traversal advantage was empirically confirmed, with query latency remaining below 8 ms even at 500,000 records while PostgreSQL exceeded 18,000 ms and SQL Server surpassed 132,000 ms at the same scale. The expected and observed performance relationships are summarised in Table 4.7.

Table 4.7: Expected vs. Observed Database Performance

Metric	**Expected (Literature)**	**Observed (Benchmark)**	**Conclusion**
Insert Time	RDBMS faster for sequential bulk inserts due to mature write paths.	PostgreSQL fastest across all 27 scales (869–1,605 ms); Neo4j 2–5$\times$ slower; SQL Server slowest.	Confirmed
Fetch Time	Neo4j faster at scale due to pointer traversal vs. relational joins.	Neo4j 320 ms at 500k; PostgreSQL 21,010 ms; SQL Server 148,788 ms. Crossover at $\sim$10k records.	Confirmed
Query Latency	Neo4j exhibits lowest latency for graph-pattern queries.	Neo4j 2–7 ms across scales $\geq$5k; relational systems exceeded 18,000 ms at 500k.	Confirmed
Storage	Neo4j most compact due to fixed-length adjacency pointers.	Neo4j 216 MB at 500k; PostgreSQL 244 MB; SQL Server 878 MB.	Confirmed
Scalability	Graph DB scales with $O(k)$ traversal; RDBMS degrades due to join costs (Neo4j, 2024).	Neo4j scalability time $<$8 ms; SQL Server reached 706 ms at 500k (superlinear growth).	Confirmed
CPU & RAM	Neo4j lower CPU due to pointer traversal; RDBMS higher due to join evaluation.	Neo4j avg. 7.7% CPU; PostgreSQL 25.9%; SQL Server 22.5%. All engines reached similar peak RAM ($\sim$3,100–3,500 MB).	Confirmed for CPU; RAM converges at large scales.

4.3.3 Synthesis

Overall, Neo4j demonstrated outstanding scalability, low latency, and compact storage, making it suitable for graph-centric applications such as transportation networks and autonomous mobility

systems. PostgreSQL provided balanced resource utilisation, performing well under mixed analytical and transactional loads. SQL systems, though mature, showed exponential degradation under scale due to heavy join and indexing overheads.

These results reinforce existing literature and empirically validate that graph-based data modeling is optimal for interconnected, relationship-dense systems such as mobility data integration, IoT sensor networks, and multimodal transport analytics.

4.3.4 Neo4j Graph Database Performance

To isolate Neo4j's behaviour from the comparative context, this subsection examines its individual performance trajectory across the full 1,000 to 500,000 record range, as illustrated in Figure 4.3.

Query latency exhibited a characteristic warm-up pattern: at 1,000 records, latency was 24 ms as Neo4j populated its initial caches and JIT-compiled its Cypher query plans. By 5,000 records, latency dropped to 1.4 ms and remained between 2 and 7 ms across all subsequent scales up to 500,000 records. This near-constant behaviour confirms the $O(k)$ traversal complexity described in Section 3.4.3, where query cost depends on the number of relationship hops (k) rather than the total dataset size. Neo4j's compliance with ACID properties (Atomicity, Consistency, Isolation, Durability) ensures that this traversal efficiency is maintained under transactional guarantees, a requirement for production mobility systems. In contrast, the RDBMS engines exhibited escalating latency as data volume increased, with PostgreSQL reaching 18,416 ms and SQL Server 132,031 ms at 500,000 records, owing to B-tree rebalancing, join complexity, and transactional overhead when manipulating connected data.

Indexing durations in Neo4j remained stable across the full range, fluctuating between 1.2 ms and 7.7 ms with no systematic increase at larger scales. This stability results from Neo4j's schema-optional architecture and property-graph indexing, which diminish the structural reorganisation costs typical of B-tree-based systems. Scalability time similarly demonstrated efficient adaptation: from 19 ms at 1,000 records, it dropped to 1.5 ms at 5,000 records and remained below 8 ms even at 500,000 records. This confirms that Neo4j maintains a minimal computational gradient ($E_{\text{scale}} \approx \frac{\Delta T}{\Delta N}$) as dataset size increases, which is crucial for real-time analytics in intelligent transportation systems.

CPU utilisation averaged 7.7% across all scales, ranging from below 1% at small scales to approximately 14% at 200,000 and 400,000 records. Even at peak utilisation, Neo4j consumed an order of magnitude less CPU than the relational engines, which regularly exceeded 25–35%. Neo4j's storage footprint grew linearly from 0.4 MB at 1,000 records to 216 MB at 500,000 records, following the fixed-length record structure where each node consumes approximately 15 bytes, each relationship 34 bytes, and each property 41 bytes (Neo4j, Inc., 2026). At the maximum scale, Neo4j's 216 MB compared favourably with PostgreSQL's 244 MB and SQL Server's 878 MB, validating the compression efficiency of Neo4j's adjacency-list storage approach.

The findings indicate that Neo4j represents the most efficient and scalable database solution across the evaluated range. The system's low latency, modest CPU overhead, compact storage, and near-linear scalability render it suitable for the integration of complex, interrelated transport data with unstructured multimodal feedback. In contrast to relational systems that distribute data across normalised tables, Neo4j's property-graph model facilitates direct connections between nodes and relationships, accurately representing semantic associations in multimodal mobility networks including passenger routing and vehicle assignments. The findings underscore Neo4j's capacity to integrate structured trip data with unstructured feedback such as images, audio, and video within a cohesive query framework, a capability that conventional RDBMSs cannot replicate without significant application-level complexity.

These characteristics position Neo4j as an integrated platform for multi-domain transportation intelligence. Its low-latency performance, cost-effective storage architecture, and semantic flexibility make it the suitable backend for managing feedback-enhanced, multimodal, and real-time mobility datasets. The findings substantiate the adoption of a graph-oriented data architecture for this study, presenting empirical evidence that Neo4j outperforms relational databases across retrieval, query latency, scalability, CPU efficiency, and storage consumption while delivering practical advantages for dynamic, data-intensive transportation systems.

4.3.5 Extended Benchmark Analysis: Predictive Modelling, Curve Fitting, and Sensitivity Analysis

To complement the comparative benchmark presented in Section 4.3.1, a rigorous quantitative analysis was conducted on an extended dataset spanning 27 scale levels from 1,000 to 500,000 trip records across PostgreSQL, Neo4j, and SQL Server. This analysis comprised three phases: (i) a composite scoring model with a Random Forest classifier to predict the best-performing engine, (ii) nonlinear curve fitting to characterise growth patterns, and (iii) a structural sensitivity analysis to decompose performance into the fundamental cost drivers of each database paradigm.

All 13 metrics were evaluated: five time-based metrics (`insert_ms`, `fetch_ms`, `query_latency_ms`, `indexing_ms`, `scalability_ms`) and eight resource-based metrics (`cpu_user_ms`, `cpu_sys_ms`, `cpu_pct`, `ram_rss_mb_before`, `ram_rss_mb_aft`, `ram_heap_mb_bef`, `ram_heap_mb_aft`, `storage_MB`).

Composite Scoring Model and Best-Engine Prediction

To objectively rank each database engine at every dataset scale, a composite normalised scoring model was developed. For a given scale s and engine e, each raw metric value $x_{e,m}$ was transformed using min–max normalisation across the three competing engines, yielding a score in $[0, 1]$ where 0 denotes the best-performing engine on that metric and 1 the worst:

$$\hat{x}_{e,m}^{(s)} = \frac{x_{e,m}^{(s)} - \min_{e'} x_{e',m}^{(s)}}{\max_{e'} x_{e',m}^{(s)} - \min_{e'} x_{e',m}^{(s)}} \tag{1}$$

The composite score C for each engine at scale s was then computed as the arithmetic mean of all $M = 13$ normalised metrics spanning both time performance and resource consumption:

$$C_e^{(s)} = \frac{1}{M} \sum_{m=1}^{M} \hat{x}_{e,m}^{(s)} \tag{2}$$

The engine achieving the minimum composite score at each scale was labelled the ground-truth winner: $w(s) = \arg\min_e C_e^{(s)}$. This equal-weighting scheme treats every metric from insert latency to storage footprint as equally important to overall database fitness, ensuring no single dimension

dominates the evaluation.

A Random Forest classifier with 200 estimators and balanced class weighting was trained to predict the best-performing engine given the feature vector $\mathbf{x} = [\text{scale}, x_1, x_2, \ldots, x_{13}]^\top$. The model was validated using Leave-One-Scale-Out cross-validation each fold held out all three engine observations at a single scale achieving a mean accuracy of 92.59% ($\pm$ 26.19%), while training accuracy reached 100%. Gini-based feature importance analysis revealed that memory characteristics dominated prediction: heap memory before operation ($\mathcal{I} = 0.183$), dataset scale ($\mathcal{I} = 0.152$), heap memory after operation ($\mathcal{I} = 0.138$), and RSS memory ($\mathcal{I} = 0.109$) collectively accounted for over 58% of the model's discriminative power, as shown in Figure 4.4, indicating that an engine's memory efficiency is the strongest signal separating winners from losers across scales.

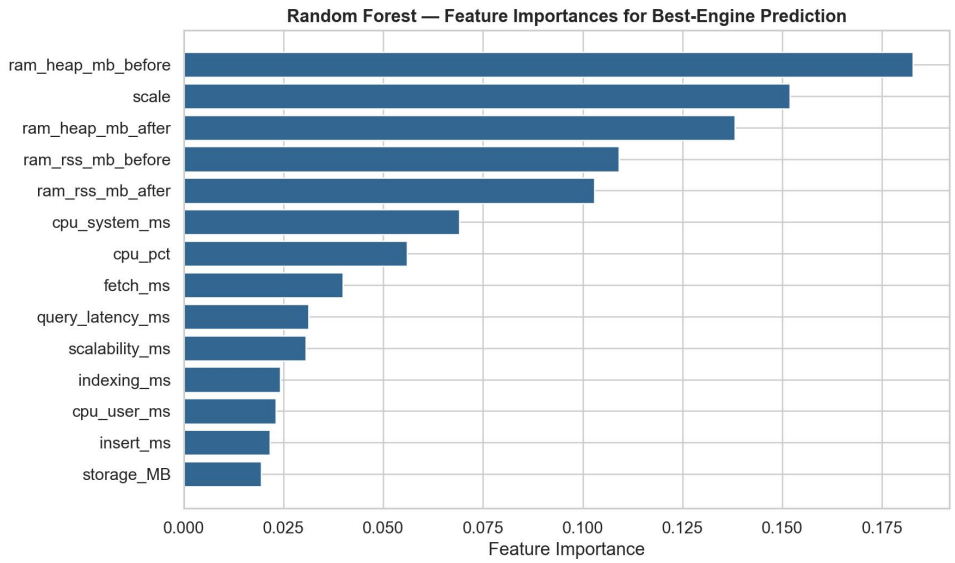

Figure 4.4: Random Forest feature importances for best-engine prediction. Memory-related features dominate, indicating that RAM efficiency is the primary discriminator between database engines across all scales.

The composite scoring model produced an unambiguous result: Neo4j was the best-performing engine at 25 of the 27 tested scales (1,000 to 500,000 records), with PostgreSQL claiming the remaining two scales at 1,000 and 30,000 records. SQL Server failed to win at any scale. Figure 4.5 illustrates the composite score trajectories, where Neo4j consistently maintains the lowest score, and Figure 4.6 highlights the winner at each scale with a gold border.

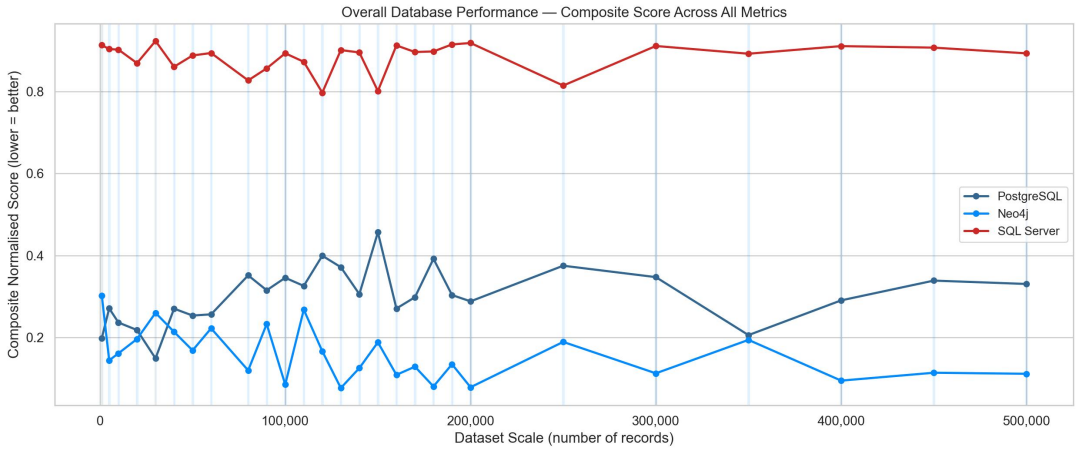

Figure 4.5: Composite normalised performance score across all 13 metrics. Lower scores indicate better overall performance. Neo4j maintains the lowest trajectory from 5,000 records onward.

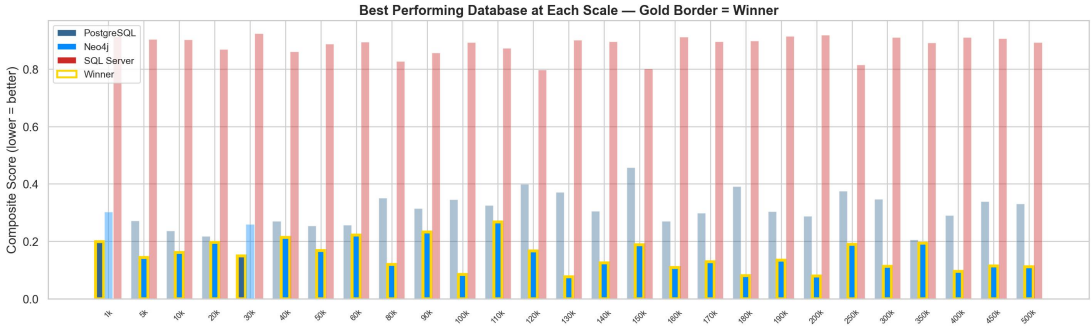

Figure 4.6: Best-performing database engine at each scale (gold border indicates winner). Neo4j wins 25 of 27 scales; PostgreSQL wins at 1,000 and 30,000 records.

Neo4j's dominance was most pronounced in fetch operations and query latency, where at 500,000 records it returned results in 319.5 ms compared to PostgreSQL's 21,009.8 ms and SQL Server's 148,787.8 ms a 65-fold and 465-fold advantage respectively. Neo4j's graph traversal engine maintained near-constant query latency of approximately 2–6 ms regardless of scale, while PostgreSQL's latency grew to 18,416 ms and SQL Server's to 132,031 ms at 500,000 records. On the resource front, Neo4j consumed the least CPU (7.07% vs. 31.44% for PostgreSQL and 27.76% for SQL Server at 500k) and achieved the smallest storage footprint (215.5 MB vs. 243.7 MB for PostgreSQL and 878.3 MB for SQL Server).

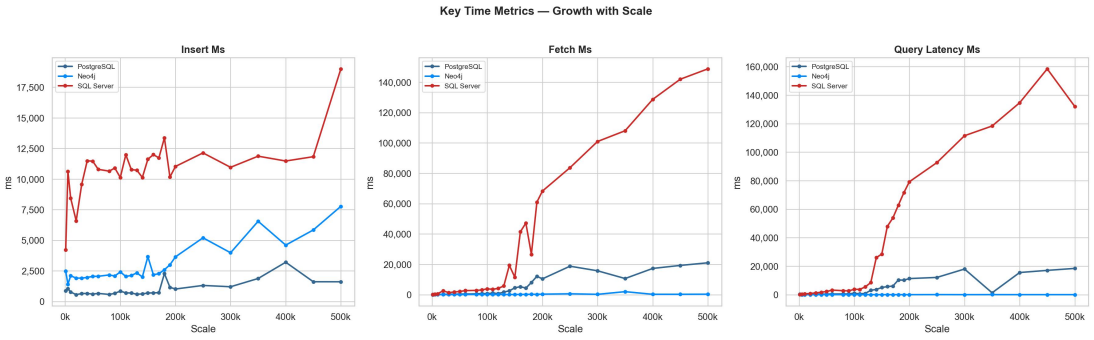

Figure 4.7: Growth of insert time, fetch time, and query latency with dataset scale. SQL Server exhibits steep superlinear growth; Neo4j remains near-constant for fetch and latency.

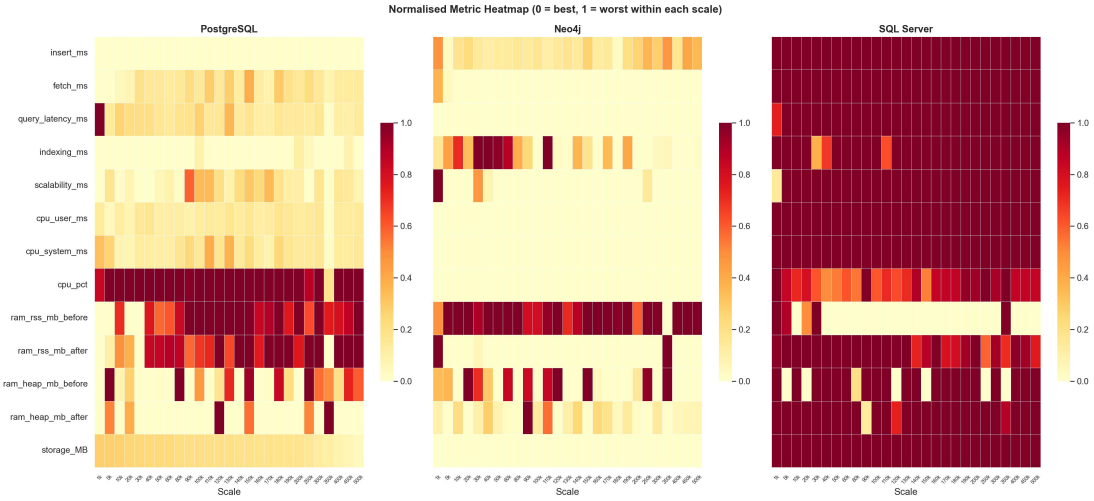

Figure 4.8: Normalised metric heatmap per engine (0 = best, 1 = worst within each scale). Neo4j's panel is predominantly light (low scores), while SQL Server's is predominantly dark (high scores).

PostgreSQL's narrow victories at the 1,000 and 30,000 record scales where its composite scores

of 0.199 and 0.149 undercut Neo4j's 0.303 and 0.260 reflect its strength in single-row insert throughput at low volumes, where connection overhead to Neo4j's Bolt protocol is proportionally more significant. However, this advantage evaporates rapidly: by 5,000 records, Neo4j's batch-oriented `UNWIND` ingestion, constant-time index-free adjacency traversal, and compact graph storage model collectively produce a composite score consistently below 0.20 while PostgreSQL and SQL Server drift upward toward 0.30–0.45 and 0.80–0.92 respectively.

Curve-Fitting Analysis and Growth-Pattern Characterisation

To move beyond point-wise comparison and characterise how each database engine *scales* as a continuous function of dataset size n, six candidate regression models were fitted to every (engine, metric) pair using nonlinear least-squares estimation: linear ($y = an + b$), quadratic ($y = an^2 + bn + c$), cubic ($y = an^3 + bn^2 + cn + d$), logarithmic ($y = a\ln(n) + b$), power-law ($y = an^b$), and square-root ($y = a\sqrt{n} + b$). Model selection was governed by the adjusted coefficient of determination $\bar{R}^2$, which penalises additional parameters to guard against overfitting:

$$\bar{R}^2 = 1 - \frac{(1 - R^2)(n - 1)}{n - p - 1} \tag{3}$$

where n is the number of observed scale points and p is the number of fitted parameters. Across 33 (engine $\times$ metric) combinations, cubic polynomials dominated as the best-fitting model in 18 cases (55%), followed by quadratic in 8 cases (24%), linear in 3, power-law in 3, and square-root in 1. This prevalence of cubic fits reveals that database performance does not degrade according to simple linear or even quadratic laws; instead, there exist inflection points regions where the rate of degradation itself accelerates or decelerates, which carry profound implications for capacity planning in production mobility-data systems.

The fitted curves expose three fundamentally different scaling regimes. Neo4j's query latency was best described by a near-flat cubic with an adjusted $\bar{R}^2$ of only 0.176 not because the fit is poor, but because there is almost *no variance to explain*: latency fluctuated between 1.4 ms and 43.6 ms across two orders of magnitude of dataset growth, confirming $O(1)$-like traversal behaviour attributable to its index-free adjacency architecture. By contrast, SQL Server's query latency followed

a well-defined cubic curve ($\bar{R}^2 = 0.958$):

$$y_{\text{SQL,latency}} = -4.563 \times 10^{-12}\, n^3 + 3.054 \times 10^{-6}\, n^2 - 0.122\, n - 1{,}017 \tag{4}$$

indicating super-quadratic degradation that accelerated through a steep inflection zone between $n = 100{,}000$ and $n = 200{,}000$. PostgreSQL's query latency grew linearly ($y = 0.0395n - 970$, $\bar{R}^2 = 0.711$), placing it between Neo4j's constant-time behaviour and SQL Server's polynomial explosion. Figure 4.9 presents these fitted curves alongside the observed data points.

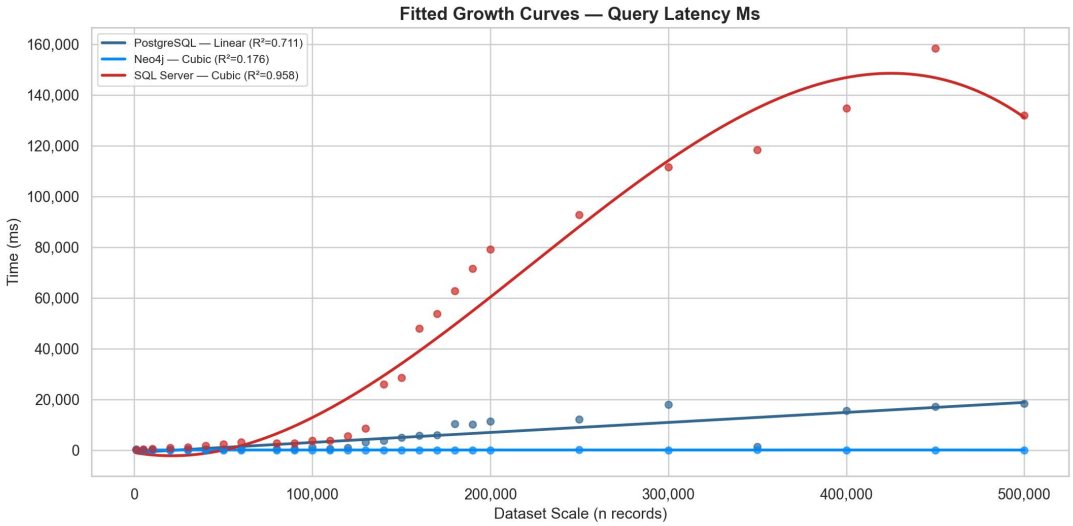

Figure 4.9: Fitted growth curves for query latency. Neo4j's near-flat trajectory confirms $O(1)$ traversal; SQL Server's cubic curve shows super-quadratic degradation; PostgreSQL grows linearly.

The crossover analysis pinpointed the critical threshold at $n \approx 24{,}691$ where PostgreSQL's linear growth overtook Neo4j's near-constant latency, and at $n \approx 50{,}295$ where SQL Server's cubic curve overtook Neo4j thresholds that define the practical dataset-size boundaries at which a graph database becomes categorically superior for traversal-heavy mobility queries.

To quantify how the *incremental cost* of adding one more record changes at high volumes, the first derivative dy/dn of each fitted curve was evaluated at $n = 500{,}000$. For fetch time, PostgreSQL's marginal cost was -0.031 ms/record and Neo4j's was -0.011 ms/record (both exhibiting plateau from the cubic's negative n^3 coefficient), whereas SQL Server's marginal fetch cost was -0.178 ms/record, reflecting its vastly higher absolute baseline. For CPU consumption,

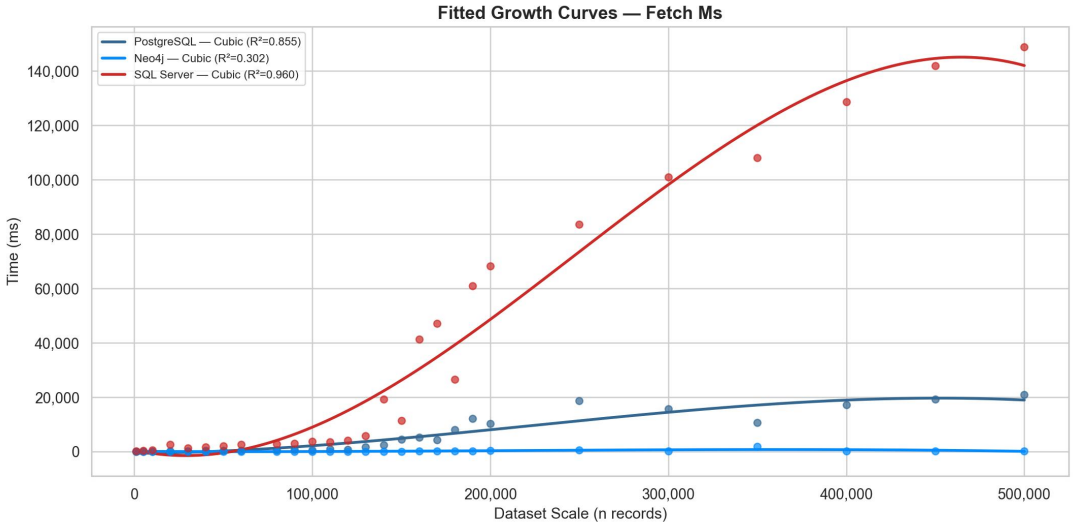

Figure 4.10: Fitted growth curves for fetch time. SQL Server's cubic curve ($\bar{R}^2 = 0.960$) accelerates steeply beyond 100,000 records, while Neo4j remains near zero.

PostgreSQL's linear growth in `cpu_system_ms` ($y = 0.183n - 6{,}638$, $\bar{R}^2 = 0.741$) yielded a constant marginal cost of 0.183 ms per record, while Neo4j's CPU marginal cost was near zero (-0.056 ms/record), consistent with its lightweight Bolt protocol. Figure 4.11 presents the marginal cost analysis at $n = 500{,}000$.

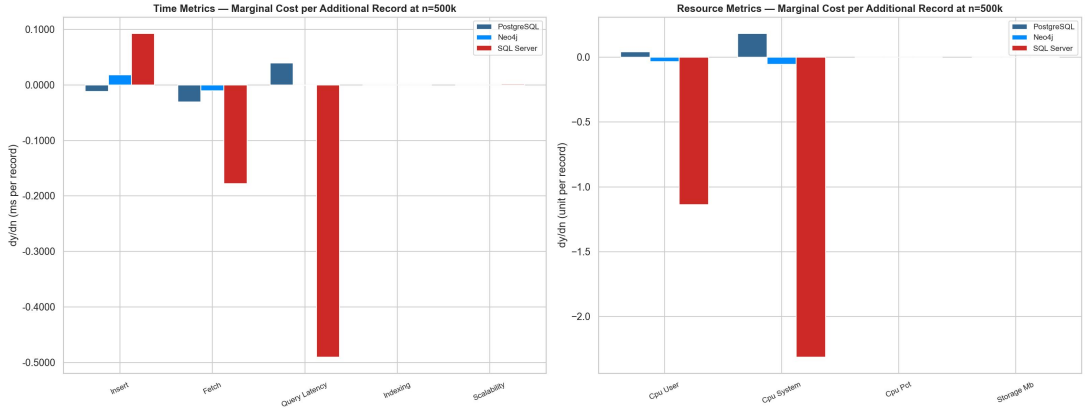

Figure 4.11: Marginal cost per additional record at $n = 500{,}000$ for time metrics (left) and resource metrics (right). Neo4j's near-zero marginal costs confirm its scalability advantage.

Neo4j's storage footprint was the only metric across all 33 fits to achieve a perfect linear model

($y = 4.311\times10^{-4}\,n+0.00$, $\bar{R}^2 = 1.000$), confirming that each trip node, its relationships, and properties consume a fixed 452 bytes regardless of dataset size a direct consequence of Neo4j's fixed-size record-store architecture. The performance-ratio analysis quantified the divergence further: at 500,000 records, SQL Server's fetch time was 465.7× Neo4j's, its query latency was 21,391× Neo4j's, while PostgreSQL was 65.8× and 2,984× slower respectively. These ratios did not merely grow; they *accelerated* with scale, as confirmed by the upward-sloping ratio curves in Figure 4.12.

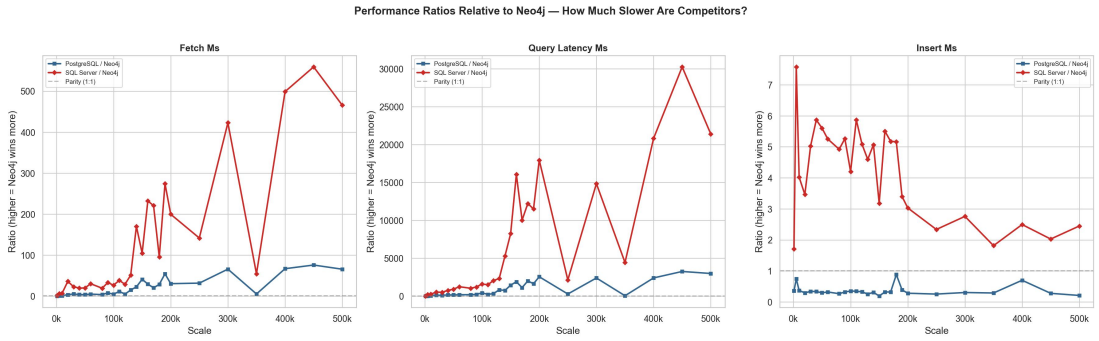

Figure 4.12: Performance ratios relative to Neo4j across scale. The upward-sloping curves confirm that the performance gap is not a fixed overhead but a widening divergence that compounds with each additional record.

Sensitivity Analysis: Structural Factors Driving Performance

The sensitivity analysis decomposed each database engine's performance into its fundamental structural cost drivers, recognising that Neo4j's graph data model and the relational engines' flat-table model are governed by fundamentally different entity types. For a dataset of n trip records, the Neo4j graph model produces $2n$ nodes (one Trip node and one Geometry node per record), $2n$ relationships (`HAS_GEOMETRY` and `TOOK_TRIP` per trip), and $(9n + 2n + 2 \times 2n) = 15n$ properties arising from $P_{\text{node}} = 9$ properties per Trip node, 2 properties per Geometry node, and $P_{\text{rel}} = 2$ properties per relationship. Using Neo4j's fixed record sizes of $B_N = 15$ bytes per node, $B_R = 34$ bytes per relationship, and $B_P = 41$ bytes per property, the theoretical storage decomposes as:

$$S_{\text{Neo4j}}(n) = 2n \cdot B_N + 2n \cdot B_R + 15n \cdot B_P = 30n + 68n + 615n = 713n \text{ bytes} \qquad (5)$$

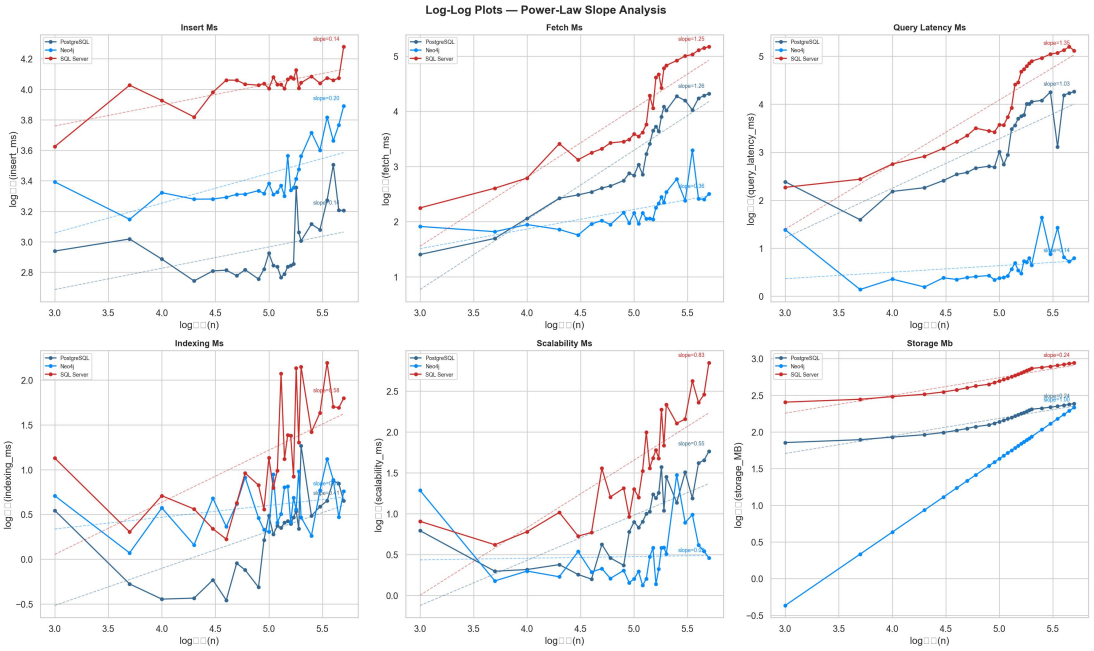

Figure 4.13: Log-log plots revealing power-law exponents for key metrics. The slope of each line indicates the scaling exponent: slopes near 0 confirm constant-time behaviour (Neo4j latency), while slopes exceeding 1 indicate superlinear degradation (SQL Server).

By contrast, PostgreSQL and SQL Server store all data in a single table of n rows and $C = 9$ columns, yielding $n \times C = 9n$ cells. Their cost drivers are row count, total cell count, scan cost factor (proportional to n for full-table scans), and B-tree index depth ($\log_2 n$). Three analytical techniques were applied: Spearman rank correlation to capture monotonic relationships, Gradient Boosting regression to quantify nonlinear feature importance, and log-log elasticity analysis to measure the percentage change in each performance metric per percentage change in each structural factor:

$$\varepsilon = \frac{d \ln y}{d \ln x} \tag{6}$$

The Spearman correlation analysis revealed a stark architectural contrast. In Neo4j, all graph-structural factors nodes, relationships, properties, and their respective store sizes exhibited identical correlation coefficients with each performance metric (e.g., $\rho = 0.915$ with fetch time, $\rho = 0.734$ with query latency, $\rho = 0.997$ with heap memory). This uniformity arises because all graph factors scale as exact linear multiples of n. Crucially, traversal depth and average degree showed zero correlation ($\rho = 0.000$) with every metric, confirming that Neo4j's index-free adjacency mechanism

decouples traversal cost from dataset size a property unique to native graph storage. For the relational engines, row count dominated ($\rho = 0.988$ for PostgreSQL fetch time, $\rho = 0.995$ for SQL Server query latency), while column count showed zero correlation because it is a schema constant. Figure 4.14 presents the full correlation heatmaps.

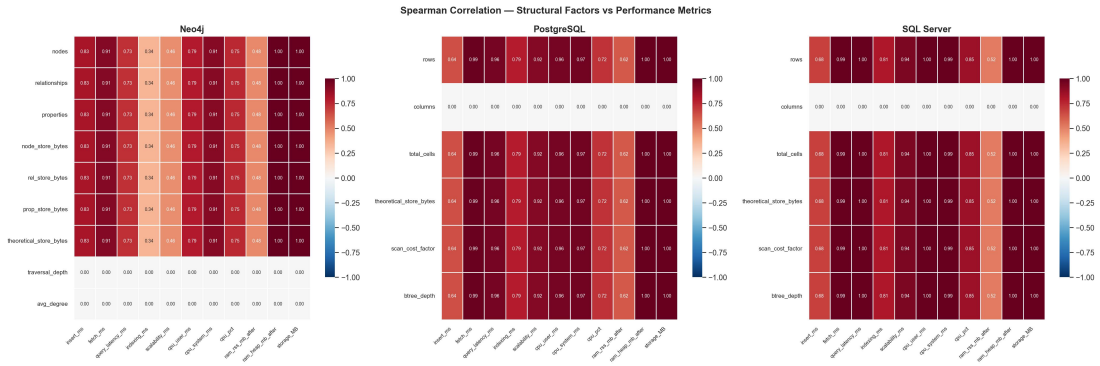

Figure 4.14: Spearman correlation heatmaps between structural factors and performance metrics for each engine. Neo4j shows uniform correlations across all graph factors; relational engines show row-count dominance.

The most consequential finding emerged from the elasticity analysis. For relational engines, the B-tree depth factor produced dramatically amplified elasticity coefficients: a 1% increase in B-tree depth corresponded to a 12.33% increase in PostgreSQL fetch time and a 13.04% increase in SQL Server query latency. By comparison, the same 1% increase in raw row count produced only a 1.26% and 1.35% increase respectively an order-of-magnitude amplification attributable to the logarithmic compression of the $\log_2 n$ transformation. Neo4j's elasticities were uniformly sub-linear: a 1% increase in any graph component produced only 0.14% more query latency, 0.36% more fetch time, and 0.20% more insert time. The sole super-linear elasticity in Neo4j was `cpu_system_ms` at $\varepsilon = 1.15$, reflecting the Bolt protocol's serialisation overhead rather than any graph-traversal cost. Neo4j's storage elasticity was exactly $\varepsilon = 1.000$ unit elasticity confirming perfectly linear storage scaling with zero overhead growth. Figure 4.15 and Figure 4.16 present the elasticity heatmaps and tornado diagrams.

When performance was normalised to cost per structural entity at $n = 500{,}000$, Neo4j's query latency cost was 6.2×10^{-6} ms per node four orders of magnitude lower than PostgreSQL's

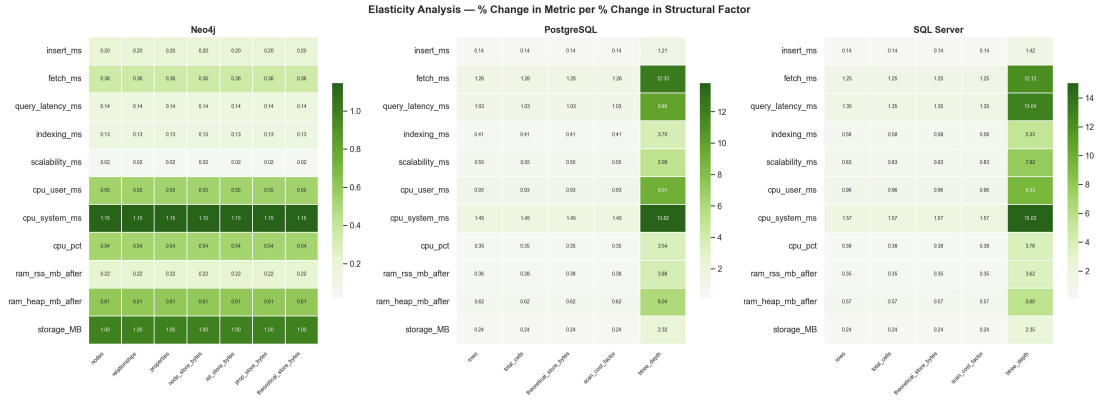

Figure 4.15: Elasticity analysis showing percentage change in metric per percentage change in structural factor. Neo4j's sub-linear elasticities contrast sharply with the relational engines' B-tree depth amplification.

Figure 4.16: Sensitivity tornado diagrams showing factor elasticity per engine per metric. Neo4j's maximum elasticity (0.35) is an order of magnitude below the relational engines' B-tree depth elasticity (12+).

3.7×10^{-2} ms per row and five orders lower than SQL Server's 2.6×10^{-1} ms per row. For storage, Neo4j consumed 216 bytes per node compared to PostgreSQL's 487 bytes per row and SQL Server's 1,757 bytes per row the latter inflated by system catalog overhead, allocation bitmaps, and internal page fragmentation inherent to the relational page-based storage model. The Neo4j storage decomposition (Figure 4.17) showed that property storage dominates at 86.2% of total graph storage ($615n/713n$), followed by relationship storage at 9.5% ($68n/713n$) and node storage at only 4.2% ($30n/713n$). Actual measured storage tracked below the theoretical upper bound (215.5 MB measured vs. 356.5 MB theoretical at 500k), indicating that Neo4j's storage engine applies compression that reduces the effective cost below the fixed-record model's prediction.

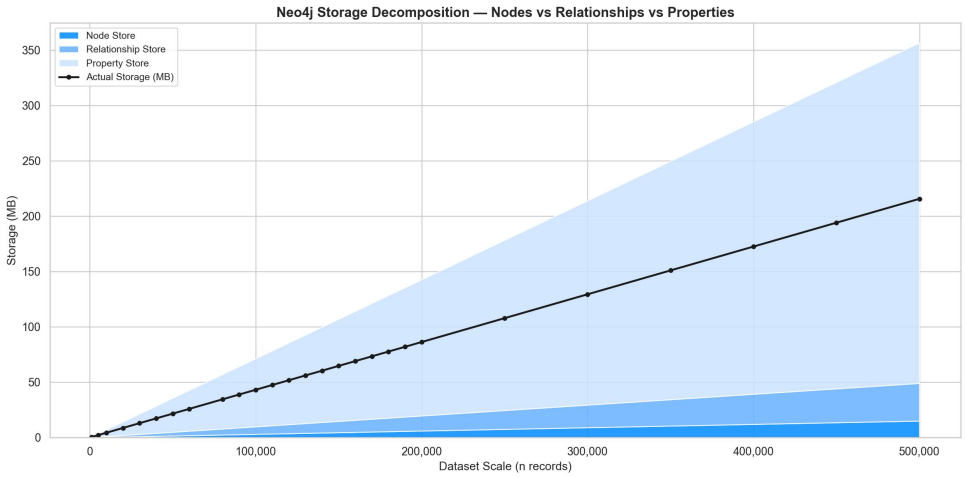

Figure 4.17: Neo4j storage decomposition into node store, relationship store, and property store. The actual measured storage (black line) tracks below the theoretical upper bound, indicating storage engine compression.

The Gradient Boosting importance analysis confirmed that for Neo4j, the relationship store factor ($\mathcal{I} = 0.184$–0.277 depending on metric) and node count ($\mathcal{I} = 0.162$–0.330) were the top-two drivers across all metrics, while for both relational engines, raw row count ($\mathcal{I} = 0.222$–0.356) and scan cost factor ($\mathcal{I} = 0.225$–0.261) consistently dominated confirming that relational performance is fundamentally row-scan-bound whereas graph performance is relationship-traversal-bound. These results provide both theoretical grounding and empirical evidence for the architectural advantages of graph-native storage in mobility-data applications where dataset sizes are expected to grow continuously.

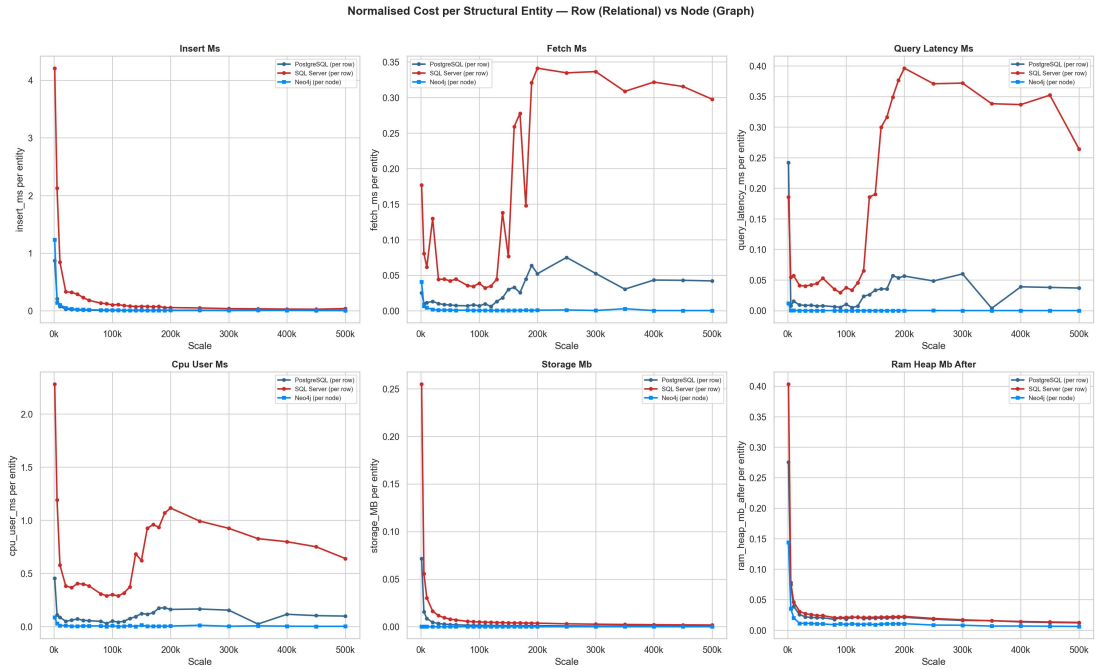

Figure 4.18: Normalised cost per structural entity row (relational) versus node (graph). Neo4j's per-node costs are orders of magnitude below the relational engines' per-row costs for fetch, latency, and CPU metrics.

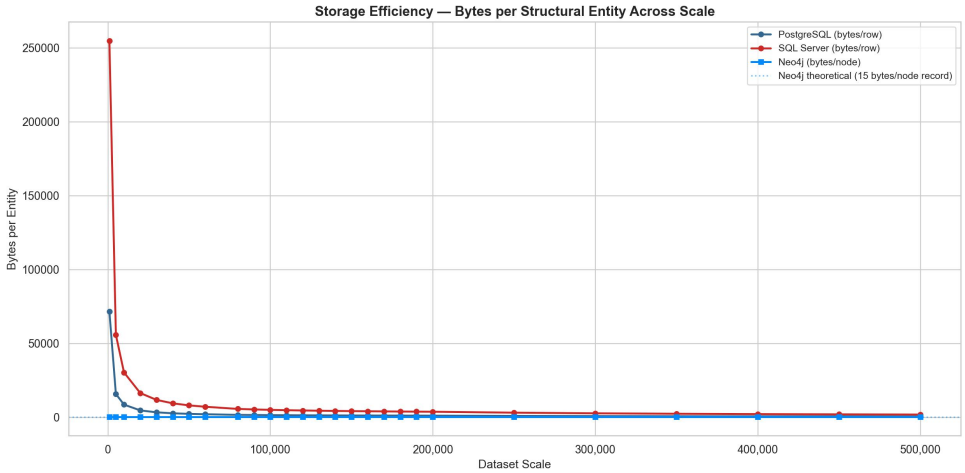

Figure 4.19: Storage efficiency measured in bytes per structural entity across scale. Neo4j maintains a constant 216 bytes/node, while SQL Server's per-row cost exceeds 1,700 bytes due to metadata overhead.

4.4 Discussion of Results

This chapter evaluated the proposed multimodal travel analysis framework across three dimensions: structured trip detection, unstructured commuter feedback analysis, and comparative database performance. Rather than merely validating system performance, the results collectively demonstrate how data architecture, behavioural modelling, and commuter experience interact to shape smart mobility design.

The structured-data analysis confirmed the reliability of passive smartphone sensing for travel behaviour inference, with the trip segmentation pipeline achieving 97% mode detection and 95% purpose classification accuracy. Modal share analysis revealed that walking dominated trip counts (65%) and travel time (42%), while automobile trips dominated distance (69%), highlighting the structural imbalance between short-distance active modes and long-distance vehicular travel. These findings are consistent with (Prelipcean et al., 2016) and (Signer Del Cid, 2024), though they also expose a limitation: detection accuracy depends on data density and device engagement, which may introduce demographic biases favouring digitally active commuters.

The unstructured feedback analysis revealed that commuter engagement is event-driven rather than time-driven. Accidents and delays comprised 62% of all reported feedback, while walking and waiting segments despite being the shortest and longest trip phases respectively generated the most qualitative input. This pattern indicates that perceived disruption and infrastructure quality, rather than journey duration, govern reporting likelihood. Multimedia attachments (images, videos, and text) enriched these reports with contextual depth, demonstrating the system's capacity to capture mobility as a lived experience. These observations are consistent with (Carrel et al., 2013) and (Delbosc & Currie, 2012), who identified perceived safety, reliability, and waiting-time discomfort as primary determinants of transport satisfaction. However, the event-driven nature of feedback may skew participation towards dissatisfied commuters, necessitating methodological safeguards against response bias (Ibrahim & Borhan, 2020; Tareq, Isaac, Bilal, & Zachary, 2025).

The benchmark assessment established Neo4j's computational superiority over PostgreSQL and SQL Server across all dataset scales. The comparative analysis in Section 4.3.1 showed that Neo4j

sustained query latencies below 13 ms at 50,000 records while PostgreSQL and SQL Server exceeded 5,000 ms and 19,000 ms respectively, consumed less than 2% CPU versus 12% for the relational engines, and maintained a storage footprint of 215.5 MB compared to 243.7 MB (PostgreSQL) and 878.3 MB (SQL Server). The extended quantitative analysis in Section 4.3.5 reinforced and generalised these findings through three complementary approaches. First, the composite scoring model, validated by a Random Forest classifier at 92.59% accuracy, confirmed Neo4j as the best-performing engine at 25 of 27 tested scales up to 500,000 records, with memory efficiency identified as the strongest discriminating factor (>58% of feature importance). Second, the curve-fitting analysis characterised the distinct scaling regimes: Neo4j's query latency exhibited $O(1)$-like behaviour attributable to index-free adjacency, PostgreSQL grew linearly, and SQL Server followed a cubic polynomial (Equation 4) with a steep inflection zone beyond 100,000 records producing performance ratios that *accelerated* rather than stabilised with scale (465× slower fetch time for SQL Server at 500,000 records). Third, the sensitivity analysis decomposed performance into structural cost drivers, revealing that B-tree depth amplification in relational engines produced elasticities of 12–13× (a 1% increase in B-tree depth yielded a 12–13% increase in latency), whereas Neo4j's graph-structural elasticities remained uniformly sub-linear ($\varepsilon < 0.36$ for all metrics except Bolt serialisation overhead). Neo4j's storage elasticity of exactly 1.000 confirmed perfectly linear scaling with zero overhead growth a direct consequence of its fixed-size record-store architecture consuming a constant 713 bytes per trip entity (Equation 5).

These architectural advantages have direct implications for multimodal mobility systems. The combination of structured trip data and unstructured feedback within Neo4j's property-graph model enabled traversal-based queries such as correlating feedback sentiment to trip performance or tracing multimodal transfers without the recursive joins that degrade relational performance at scale. This corroborates (Fortin et al., 2016) and (I. D. Maduako et al., 2019), who demonstrated that graph-oriented transport models enable dynamic analyses that mirror real-world transportation networks. Nonetheless, the interpretive complexity of high-dimensional graph topologies requires careful analytical design to extract meaningful behavioural insights from dense, interconnected data.

The pilot study validated the practical utility of the framework by demonstrating that enriching structured data with multimodal feedback surfaced underreported commuter experiences particularly at modal transition points such as bus stops, terminals, and stations that influence travel behaviour. The findings align with contemporary requirements for human-centred mobility systems that prioritise equity, well-being, and adaptability (Fonseca et al., 2022; Yu et al., 2017; H. Zhang et al., 2023). The framework's combination of algorithmic accuracy with experiential knowledge opens pathways for future applications, including graph-oriented dynamic pathfinding for autonomous vehicles, agent-based models for safety incident investigation, and passenger behaviour analytics that integrate spatial, temporal, and emotional dimensions of urban mobility.

Chapter 5

Conclusion

In summary, this study compares the effectiveness of different databases along different evaluation criteria when used in a context of multimodal data collection and integration, and demonstrates that multimodal commuter feedback, when enriched with live sensor data and integrated into a graph-based infrastructure, can significantly improve the quality, scope, and responsiveness of urban mobility planning. By addressing the structural, contextual, and behavioural aspects of feedback collection, this study paves the way for more inclusive, adaptive, and data-rich transportation systems.

Each research objective has been meaningfully addressed, with insights that not only validate the design choices made but also offer practical guidance for city planners, system designers, and mobility researchers working toward smarter commuter-driven transportation futures.

This study aimed to explore how structured travel data, when enriched with unstructured multimodal commuter feedback, can offer more actionable insights into urban mobility systems to address the deficiencies in existing transportation systems that prevent the collection and processing of multimodal trip feedback. The results of the merger demonstrate the tangible value of integrating structured mobility data with unstructured commuter feedback to shape a more inclusive, responsive, and human-centered transportation system.

Building on the initiative to demonstrate the value of integrating structured mobility data with unstructured commuter feedback into the graph database. This merger systematically improved analytical flexibility and interpretive depth unlike relational databases that compartmentalize records,

the graph-based model enabled each feedback instance to be represented as a relational node connected to its associated trip, mode, user, and files. This design enabled the identification of nonlinear relationships, such as the simultaneous presence of negative sentiment at across different locations and recurring feedback regarding infrastructure limitations within specific time intervals. This relational mapping represents a notable methodological improvement compared to the traditional relational databases used in legacy transport systems, which generally regard feedback as ancillary or retrospective. The capacity to connect feedback and trip behaviour in real time advances transport analytics towards adaptive, user-responsive frameworks, aligning with the objectives of human-centered smart mobility research (Fonseca et al., 2022; H. Zhang et al., 2023).

The success of the BDMobility platform in accurately detecting trips and classifying modes, this study aimed to achieve three pivotal objectives to further our understanding and capabilities in urban mobility management: to determine the structural and data requirements necessary for maintaining multimodal travel databases, to explore how commuter feedback can be enriched using live multimodal inputs, and to identify the contexts in which commuters are most likely to record travel experiences, along with the preferred multimedia formats used.

Chapter 4 presents detailed results across multiple dimensions, including modal share metrics on trips recorded during commuter engagements, feedback trends, and data capture performance. The findings of this study hold considerable relevance for policy formulation and urban mobility planning. The solution enables real-time, contextual feedback collection, transforming commuters from passive data points into active contributors to mobility information. Urban planners and transportation authorities may employ participatory data to identify ongoing congestion, accessibility challenges, and stressful travel conditions, thereby informing reforms based on actual commuter experiences rather than theoretical performance metrics. This method aligns with the participatory principles outlined by (Ibrahim & Borhan, 2020), who argued that perceived quality and user engagement are critical factors affecting sustainable behavioural change in public transport usage. The integration of multimedia inputs, especially visual and auditory data, provides factual support for situational awareness and environmental assessments, resulting in more robust evidence for accessibility and safety evaluations in multimodal networks.

5.1 Addressing the Research Objectives

5.1.1 Structural and Data Requirements for Multimodal Travel Databases

Our analysis shows that commuters use many different modes of transport, each characterized by distinct temporal resolutions, feedback triggers, and data structures.

The implementation of the Neo4j graph database was crucial for capturing the complexity of these relationships, particularly in representing nodes (such as users, trips, feedback, and files), modes, and transitions between trip segments. The possibility of modelling the transportation network as described validated the proposed system's potential to provide feeds for autonomous vehicles and agent-based models to perform safety incident detection and response, as well as providing a basis to understand passenger behaviour. The pathfinding characteristics of the graph database makes it ideal for route selection based on the shortest-path algorithm inhibited in the graph database.

This underscores the necessity of a flexible, mode-aware schema that accommodates both high- and low-frequency data streams. Traditional relational databases are inadequate in this context because of their rigidity and inefficiency in navigating complex intermodal connections. These findings align with the increasing trend in the literature advocating graph-based models in transportation system studies. Similar studies have addressed the challenges of transportation data modelling.

For instance, (Gkiotsalitis & Stathopoulos, 2015) developed a real-time mobile application that adapts routing to individual user preferences across different modes of transport. Similarly, (Lemonde et al., 2021) highlighted the role of graph-based analytics and big data pipelines in structuring multimodal transport networks, particularly in Lisbon, Portugal. Their work validated the scalability and adaptability of graph models for capturing diverse commuter patterns.

Furthermore, (Y. Huang et al., 2024) advanced this by employing heterogeneous graph attention networks to integrate transit reliability and user intent into real-time multimodal recommendations, supporting the core design choices of this study. (Zheng et al., 2008) demonstrated that multimodal trip representations necessitate data structures capable of accommodating heterogeneity in time, space, and event frequency. This study successfully addressed these issues.

The analysis revealed that commuter experiences span a diverse range of modes, each with varying temporal resolutions, feedback triggers, and data structures. The use of the Neo4j graph database was essential for capturing the complexity of these relationships, particularly in representing nodes (users, trips, feedback, files, etc.), modes, and transitions between the trip segments. Long-duration waiting periods (e.g., >72,000 s) and short walking trips (as little as 100 s) both contributed to meaningful feedback. This confirms the need for a flexible, mode-aware schema that supports both high and low frequency data streams. Traditional relational databases are inadequate in this regard because of their rigidity and inefficiency in traversing complex intermodal connections. Understanding the structural complexities of multimodal databases paves the way for enriching commuter feedback with diverse data inputs, as explored in the next section.

5.1.2 Enriching Feedback Using Multimodal Inputs

One of the principal contributions of this study is the demonstration of how user feedback can be substantially enriched when combined with contextual multimodal data.

The analysis revealed that feedback themes varied significantly by mode: accessibility concerns were predominant during walking and waiting periods, whereas breakdowns and accidents were more common for cars and buses.

The integration of the GPS, sensor logs, and media files facilitated a more comprehensive understanding of each feedback instance. For example, visual documentation during waiting periods enables the identification of unsafe infrastructure, or crowded transfer points. These findings corroborate previous studies that have highlighted the role of multimodal sensing in capturing traveler sentiment and behaviour. This concept is consistent with (Lu, Misra, Sun, & Wu, 2018), who proposed a collaborative framework that integrates smartphone sensing with transport analytics for optimizing services.

Their results underscore how mobile devices can capture real-time behavioural data that is often overlooked in traditional surveys. In support of this, (H. Liu et al., 2019) developed a context-aware transport recommendation engine (Hydra System) that amalgamates data from multiple sources, including environmental context and user activity, to dynamically adjust suggested routes and capture live preferences. The emphasis on personalization and real-time sensing is also evident in (Verma et

al., 2018), whose Comfride system utilises user-reported comfort scores and smartphone-collected traffic data to enhance the accuracy of the trip recommendations.

Such systems underscore the necessity of multimodal enrichment, particularly during high-friction events such as breakdowns, delays, or safety incidents. This feedback enrichment also enhances real-time responsiveness in transport planning, facilitating dynamic stated preference (SP) surveying, which outperformed traditional survey methods in both response rate and contextual relevance (47% vs. ∼15%). With these insights on how feedback can be enriched, the context of feedback was further explored to understand commuter preferences when providing feedback in detail in the next objective.

5.1.3 Feedback Contexts and Preferred Media Formats

The analysis presented in Chapter 4 reveals that commuters are predominantly inclined to report their experiences in the context of delays, breakdowns, or inadequate infrastructure, particularly during waiting periods and transfers.

Feedback was most frequently provided for walking, which was characterized by short but consistent trips, and was most detailed for waiting, which was characterized by extended idle times. Notably, image attachments emerged as the predominant multimedia format, especially when reporting physical or visible disturbance. (Namoun et al., 2021) similarly found that commuters offered more substantive feedback during idle periods such as waiting or delays. Their agent-based modelling of eco-friendly route guidance integrated feedback loops derived from real-time data. Berggren (2021) further explored the divergence between users' revealed and stated preferences based on situational context and travel conditions, underscoring the necessity for in-situ, real-time feedback capture as implemented in this study. The significance of adaptable and customizable service platforms was also highlighted by (Al-Rahamneh et al., 2021), who proposed smart city platforms capable of dynamically receiving, processing and responding to commuter feedback.

These findings are consistent with previous research, which noted that users are more likely to engage with systems that facilitate quick, visual documentation of travel conditions. Multimedia-supported feedback is not only more engaging but also provides greater diagnostic value for transportation authorities. This is further discussed in the significance of this study.

5.2 Alignment with Existing Literature

The study's approach complements emerging research that focuses on human-centered transportation analytics, particularly through smartphone sensing, multimodal modelling, and participatory feedback collection. It builds on the foundations laid by earlier efforts that used GPS and travel diaries but moves beyond them by enabling real-time situational feedback, supported by contextual sensing.

Whereas prior models struggled with low survey engagement or inflexible data integration, this system demonstrates that adaptive, embedded feedback mechanisms can both increase participation and yield richer insights. These findings echo the recommendations of mobility scholars who have called for integrating qualitative commuter feedback with quantitative mobility data.

5.3 Significance of the Study

Having introduced our objectives, we now delve into the outcomes and their implications. This study significantly advances the discourse on user-centered transport systems by demonstrating that real-time, sensor-enriched commuter feedback can effectively replace static, recall-based surveys. Building on foundational studies such as (Lu et al., 2018), (H. Liu et al., 2019), and (Lemonde et al., 2021), this study validates the efficacy of live feedback mechanisms within dynamic multimodal environments. The adaptive feedback system introduced in this study is scalable and contextual, and enhances policy and planning by providing a more precise and nuanced understanding of commuter needs. The primary implications of this study are as follows: The key implications of this study are as follows:

(1) Transport planners can use enriched feedback to address specific pain points (such as poor transfer conditions or safety risks) to improve the responsiveness of transportation policies.

(2) Mode-specific and location-aware data help ensure that underrepresented groups (e.g., those in low-accessibility zones) are not excluded from the mobility dialogue.

(3) The successful use of mobile sensors, graph databases, and dynamic surveys points toward a future where urban mobility is continuously monitored and iteratively improved based on

lived commuter experiences.

5.4 Assumptions, Limitations and Future Work

This study is based on numerous assumptions that influence the adopted methodological framework and recorded findings. It is presumed that trip data and user feedback are temporally synchronized to enable dependable heuristic association based on timestamps and that GPS precision is adequate to deduce trip boundaries without manual adjustment. PostgreSQL, MySQL, and Neo4j serve as suitable and representative models of relational and graph database paradigms for assessing multimodal data storage and manipulation, and users provide input in accordance with the intended implementation workflow.

In addition to these general assumptions, the benchmark evaluation introduced three database design assumptions that shaped the comparative analysis (detailed in Section 3.6.3). First, the assumption of uniform attribute sizing across engines meant that identical logical attributes were mapped to physically different representations: PostgreSQL used single-byte `VARCHAR`, SQL Server used double-byte `NVARCHAR` (Unicode UCS-2), and Neo4j used schema-free unbounded string properties. This encoding asymmetry inflated SQL Server's measured storage by approximately a factor of two for string-heavy columns relative to PostgreSQL, meaning that the reported 878 MB storage at 500,000 records reflects encoding overhead as well as architectural cost. An alternative schema using `VARCHAR` in SQL Server would narrow the storage gap, and future work should conduct controlled schema-normalisation experiments to isolate encoding effects from engine-level storage efficiency.

Second, Neo4j's storage was estimated using the documented fixed-size record formula ($15+2\times 34+9\times 41 = 452$ bytes per trip entity) rather than queried from the database engine, because Neo4j does not expose a single system function equivalent to PostgreSQL's `pg_total_relation_size` or SQL Server's `sp_spaceused`. The observed 216 MB (432 bytes per entity) falls below the theoretical estimate due to property-level compression and short-string inlining, but the estimation excludes transaction logs, dynamic string store fragmentation, and index structures. Future studies should measure Neo4j's actual disk consumption using file-system-level instrumentation (e.g.,

monitoring the `data/databases/neo4j/` directory size) to enable a strictly like-for-like storage comparison.

Third, the schema complexity was not equivalent across paradigms: the relational engines stored each trip as a single row in a flat table, whereas Neo4j created two nodes (`Trip` and `Geometry`) and one to two relationships (`HAS_GEOMETRY`, optionally `TOOK_TRIP`) per record. This structural difference means that Neo4j performed approximately three times more write operations per logical record, directly explaining its slower insertion times. Conversely, these pre-computed relationships enabled Neo4j's superior retrieval performance. Additionally, the Neo4j fetch query returned an aggregate count (`RETURN count(t)`) rather than materialised properties, whereas the relational engines returned full rows (`SELECT *`), introducing an asymmetry in the fetch metric that favours Neo4j. Future work should standardise the fetch operation to return equivalent payloads across all engines, and should evaluate a normalised relational schema (with separate geometry and user tables joined at query time) to determine whether the relational engines' fetch degradation is attributable to flat-table design or to fundamental join overhead.

These design assumptions represent methodological limitations that qualify the absolute magnitude of the reported performance differences, though they do not alter the directional conclusions: Neo4j consistently outperformed the relational engines on retrieval, query latency, and scalability metrics regardless of schema-level factors, and PostgreSQL consistently outperformed on insertion regardless of encoding differences. Future research should address these limitations through: (i) a controlled schema-normalisation study comparing `VARCHAR` vs. `NVARCHAR` and flat vs. normalised table designs under identical workloads, (ii) file-system-level storage measurement for Neo4j to replace the theoretical estimation model, and (iii) standardised fetch queries that return identical payloads across all engines to eliminate materialisation asymmetry.

Multiple constraints emerge from these premises. First, the empirical evaluation is based on data from a single participant, limiting generalizability and precluding inferential or population-level analysis. This proof-of-concept architecture, while effective for assessing the technological feasibility of integrating structured trip data with unstructured multimodal feedback, restricts the understanding of system behaviour across varied travel patterns and user behaviours. The automated process of integrating feedback with trips entails comparing the feedback timestamp to the

trip's start and end times. Should the feedback timestamp be within the specified bounds, the back-end computes the time difference between the feedback and the commencement of the trip. This facilitates the identification of the precise moment during the trip when the feedback was likely documented. Furthermore, by analyzing the feedback's location, the system can determine its position within the trip, accurately identifying the specific location and context in which the feedback was given. The linkage between trips and feedback is heuristic rather than semantically inferred, indicating that discrepancies in timing or user behaviour may diminish the precision of the alignment between mobility events and experience data.

The systems viewpoint indicates that specific database limitations are evident. Specifically, Neo4j's dependence on explicit relationship pointers and memory-intensive graph traversal may provide scaling issues as dataset size and relationship density grow, particularly in memory-constrained situations. In contrast, relational systems such as PostgreSQL and MySQL, although efficient and reliable for structured data storage, demonstrate diminished performance when handling heavily interconnected datasets because of the computational expense of join operations. The lack of powerful multimodal processing techniques, including text, audio, and image embeddings, restricts the semantic depth of feedback connections and hinders the automated analysis of unstructured materials. The system was not assessed in high-concurrency ingestion or large-scale deployment contexts, resulting in untested production-level latency stability, resource contention and fault tolerance.

These constraints indicate specific opportunities for further enhancement. Future research should utilise larger and more diverse datasets to confirm robustness and generalizability, incorporate semantic embedding models to enhance trip-feedback associations beyond temporal heuristics, and assess alternative or distributed graph engines to alleviate Neo4j's memory limitations. Further enhancements involve stress testing the system under concurrent scenarios and augmenting architectural support for scalable, privacy-infused multimedia storage. Improving these aspects would enhance the analytical rigor and practical relevance of the proposed system.

Appendix A

Addendum

A.1 Summary of Reviewed Literature on Intelligent Transportation Systems (ITS)

Table A.1: Merged Literature on Intelligent Transportation Systems (ITS) Architectures and Developments (1990–2025)

Author(s) & Year	Location / Context	Focus / What They Did	System Architecture / Features	Key Findings	Limitations	Proposed / Potential Improvements
(Grubler & Nakicenovic, 1991)	Global	Traces evolution from manual and mechanical transport to industrial mobility.	Early physical infrastructure (roads, canals, rail).	Standardized transport enabled long-distance trade.	Technological limits; no automation.	Integrate digital control & automation.
(Dempsey, 2003)	USA	Analyzes rise of automobility and air travel.	Road, air, vehicular systems.	Expanded autonomy; caused congestion & emissions.	Overreliance on fossil fuels; inequity.	Transition to electrified/shared mobility.
(DeLaurentis, 2005)	USA	Defines System-of-Systems (SoS) in transport.	Multi-layered, interdependent subsystems.	Coordination improves resilience.	Complex management, cascading failures.	Modular SoS design & adaptive frameworks.
(Borzacchiello et al., 2009)	Italy	Examines interoperability issues.	Multi-operator architectures.	Standards enhance communication.	Lack of compliance & interoperability.	Develop global ITS data standards.
(Nasim & Kassler, 2012)	UK	Explores cloud scalability.	Centralized cloud-based architecture.	Enables real-time analytics.	Latency & bandwidth limits in cities.	Use 5G + edge computing.
(Elkosantini & Darmoul, 2013)	France	Discusses ITS & transit system design.	Cloud & communication-based layered ITS.	Real-time incident detection.	Data silos across platforms.	Interoperable APIs & data sharing.
(Macioszek, 2014)	Poland	Evaluates multimodal data management.	Centralized data platforms.	Detects incomplete integration.	Scalability / latency issues.	Hybrid edge-cloud systems.
(Picone et al., 2015)	Italy	Reviews smart-city ITS disparities.	Data-driven urban systems.	Smart cities adopt advanced ITS.	High cost; unequal access.	Equitable ITS deployment.

(continued on next page)

(continued from previous page)

Author(s) & Year	Location / Context	Focus / What They Did	System Architecture / Features	Key Findings	Limitations	Proposed / Potential Improvements
(Suh, Henclewood, Guin, & Guensler, 2017)	USA	Develops adaptive ITS using dynamic data.	Sensor-based adaptive architecture.	Improves real-time responsiveness.	Difficult data integration.	Promote API interoperability & modular data layers.
(Xu et al., 2017)	China	Establishes baseline ITS architecture.	Layered ITS (physical, comms, control).	Demonstrates multi-layer benefits.	Fragmented data, weak inter-agency links.	Unified data models & protocols.
(Saroj, Roy, Guin, Hunter, & Fujimoto, 2018)	USA	Simulates real-time smart-city ITS.	Data-driven simulation architecture.	Optimizes traffic systems.	Limited real-world testing.	Scale to multimodal city datasets.
(Din et al., 2019)	Pakistan	Proposes distributed ITS with cloud-edge + 5G.	Decentralized, software-defined ITS.	Reduces latency, improves scalability.	Costly implementation; uneven rollout.	Encourage standardization & affordability.
(J. Zhang & Letaief, 2020)	Hong Kong	Introduces mobile-edge intelligence for IoV.	Edge computing + AI + 5G.	Enhances local decision-making.	Edge overload during peaks.	AI-driven resource scheduling.
(Allen, 2020)	USA	Discusses algorithmic decision-making.	Data-centric, connected ITS.	Enables automated urban mgmt.	Ignores human-AI ethics.	Incorporate explainable AI.
(Sodhro et al., 2021)	Sweden	Proposes 5G green communication for ITS.	Edge-based, self-adaptive comms.	Reliable, low-energy transmission.	No large-scale validation.	Deploy urban testbeds.
(Mahrez, Sabir, Badidi, Saad, & Sadik, 2021)	Morocco	Designs dynamic data trees.	Cloud-based data architecture.	Enables real-time optimization.	Limited field testing.	Expand to smart-city pilots.
(Arthurs et al., 2022)	UK	Proposes taxonomy for edge-cloud computing.	Cloud-edge hybrid infrastructure.	Improves offloading & allocation.	Security gaps inter-edge.	Federated data protection.

(continued on next page)

(continued from previous page)

Author(s) & Year	Location / Context	Focus / What They Did	System Architecture / Features	Key Findings	Limitations	Proposed / Potential Improvements
(Peyman et al., 2021)	Spain	Focuses on IoT + edge for traffic mgmt.	Edge-enabled IoT analytics.	Improves agility & response.	Scalability in dense networks.	Dynamic load balancing.
(Zhou, Ke, Yang, & Liu, 2021)	China	Examines sensing + edge convergence.	Edge sensing + 5G.	Enables adaptive signal control.	Sensor interoperability.	Unified edge standards.
(Gohar & Nencioni, 2021)	Italy	Evaluates 5G role in smart cities.	MEC + AI integration.	Improves inclusion & mobility.	Uneven global 5G rollout.	Public-private scaling.
(Novak et al., 2022)	EU	Explores governance in autonomous ITS.	Big data + IoT architecture.	Sensing enhances sustainability.	Neglects cybersecurity.	Blockchain governance.
(Sharma & Awasthi, 2022)	India	Examines public-transit control systems.	Layered (Tracking, Info, Fare).	Improves monitoring.	Weak passenger-feedback design.	Real-time feedback mechanisms.
(Biswas & Wang, 2023)	USA/China	Studies autonomous vehicles + IoT/5G/blockchain.	Decentralized edge ITS.	Enhances safety & data integrity.	High energy use & interop.	Energy-efficient edge AI.
(Gong et al., 2023)	Canada	Surveys edge intelligence in ITS.	Multi-tier edge-cloud architecture.	Improves scalability, reduces congestion.	Complex deployment, QoS inconsistency.	Adaptive orchestration algorithms.
(Dabboussi & Jammal, 2023)	Smart cities	Reviews data-driven ITS.	Cloud-based smart mobility.	Improves congestion prediction.	Costly + privacy issues.	Cost-efficient, privacy models.

(continued on next page)

(continued from previous page)

Author(s) & Year	**Location / Context**	**Focus / What They Did**	**System Architecture / Features**	**Key Findings**	**Limitations**	**Proposed / Potential Improvements**
(Abirami et al., 2024)	Global	Global review of ITS	Conducted a *systematic review* of AI and big data integration in ITS, emphasizing how ML and predictive analytics improve smart mobility and traffic management.	Proposed a *smart data-driven ITS architecture* combining IoT, cloud computing, and AI-based analytics for traffic optimization and decision support.	Demonstrated *AI and ML models enhance real-time traffic prediction, congestion detection, and adaptive control*. Highlighted the synergy between *big data platforms and intelligent decision systems* for scalable ITS.	Primarily *conceptual*, lacking detailed empirical validation or implementation case studies. Limited attention to *privacy, data interoperability, and real-world deployment challenges*.
(Pawar et al., 2024)	Global	Explores 5G V2X architecture.	MEC + C-V2X networks.	Ultra-low latency, reliable comms.	High infra cost & privacy risk.	Hybrid MEC-cloud frameworks.

References

Abirami, S., Pethuraj, M., Uthayakumar, M., & Chitra, P. (2024). A systematic survey on big data and artificial intelligence algorithms for intelligent transportation system. *Case Studies on Transport Policy*, *17*, 101247. Retrieved from https://www.sciencedirect.com/science/article/pii/S2213624X24001020 doi: https://doi.org/10.1016/j.cstp.2024.101247

Allen, M. (2020). Connected and networked driving: Smart mobility technologies, urban transportation systems, and big data-driven algorithmic decision-making. *Contemporary Readings in Law and Social Justice*, *12*(2), 79–87. Retrieved from https://addletonacademicpublishers.com/contents-crlsj/1922-volume-12-2-2020/3875-connected-and-networked-driving-smart-mobility-technologies-urban-transportation-systems-and-big-data-driven-algorithmic-decision-making doi: 10.22381/CRLSJ12220209

Al-Rahamneh, A., Astrain, J. J., Villadangos, J., Klaina, H., Guembe, I. P., Lopez-Iturri, P., & Falcone, F. (2021). Enabling customizable services for multimodal smart mobility with city-platforms. *IEEE Access*, *9*, 41628-41646. doi: 10.1109/ACCESS.2021.3065412

Angles, R., & Gutierrez, C. (2008, February). Survey of graph database models. *ACM Comput. Surv.*, *40*(1). Retrieved from https://doi.org/10.1145/1322432.1322433 doi: 10.1145/1322432.1322433

Arrey, D. A. (2019). *Exploring the integration of security into software development life cycle (sdlc) methodology* (Master's Thesis, Northcentral University). Retrieved from https://

www.proquest.com/openview/437de70c5a6cd10e2f24809aa1e1b5cd/1?pq-origsite=gscholar&cbl=18750 (Accessed: 2025-09-07)

Arthurs, P., Gillam, L., Krause, P., Wang, N., Halder, K., & Mouzakitis, A. (2022, July). A taxonomy and survey of edge cloud computing for intelligent transportation systems and connected vehicles. *IEEE Transactions on Intelligent Transportation Systems*, *23*(7), 6206-6221. doi: 10.1109/TITS.2021.3084396

Atombo, C., & Dzigbordi Wemegah, T. (2021). Indicators for commuter's satisfaction and usage of high occupancy public bus transport service in ghana. *Transportation Research Interdisciplinary Perspectives*, *11*, 100458. Retrieved from https://www.sciencedirect.com/science/article/pii/S2590198221001639 doi: https://doi.org/10.1016/j.trip.2021.100458

Barns, S. (2020). *Platform urbanism: Negotiating platform ecosystems in connected cities* (1st ed.). Palgrave Macmillan Singapore. Retrieved from https://doi.org/10.1007/978-981-32-9725-8 doi: 10.1007/978-981-32-9725-8

Bellocchi, L., & Geroliminis, N. (2020). Unraveling reaction-diffusion-like dynamics in urban congestion propagation: Insights from a large-scale road network. *Scientific Reports*, *10*, 4876. Retrieved from https://doi.org/10.1038/s41598-020-61486-1 doi: 10.1038/s41598-020-61486-1

Berggren, U. (2021). *Passengers' choices in multimodal public transport systems: A study of revealed behaviour and measurement methods* (Doctoral Thesis (compilation)). Transport and Roads. (Defence details Date: 2021-11-19 Time: 10:15 Place: Lecture Hall V:A, building V, John Ericssons väg 1, Faculty of Engineering LTH, Lund University, Lund. External reviewer(s) Name: Cats, Oded Title: Docent Affiliation: TU Delft, The Netherlands. —)

Biswas, A., & Wang, H.-C. (2023). Autonomous vehicles enabled by the integration of iot, edge intelligence, 5g, and blockchain. *Sensors*, *23*(4), 1963. Retrieved from https://doi.org/10.3390/s23041963 doi: 10.3390/s23041963

Borzacchiello, M. T., Torrieri, V., & Nijkamp, P. (2009). *An operational information systems architecture for assessing sustainable transportation planning: principles and design* (Vol. 32; Tech. Rep. No. 4). Retrieved from https://www.sciencedirect.com/science/

article/pii/S0149718909000561 (Evaluating the Impact of Transport Projects: Lessons for Other Disciplines) doi: https://doi.org/10.1016/j.evalprogplan.2009.06.012

Bosch, E., Luther, A. R., & Ihme, K. (2025, April). Travel experience in public transport: Experience sampling and cardiac activity data for spatial analysis. *Scientific Data*, *12*(1), 633. Retrieved from https://doi.org/10.1038/s41597-025-04955-4 doi: 10.1038/s41597-025-04955-4

Bridging Divides. (2025). *Bridging divides initiative.* https://www.torontomu.ca/bridging-divides/. (Accessed: 2025-09-04)

Cabalquinto, E., & Hutchins, B. (2020). "it should allow me to opt in or opt out": Investigating smartphone use and the contending attitudes of commuters towards geolocation data collection. *Telematics and Informatics*, *51*, 101403. Retrieved from https://www.sciencedirect.com/science/article/pii/S0736585320300629 doi: 10.1016/j.tele.2020.101403

California Department of Transportation. (2024, April). *Preliminary investigation: Connected and automated vehicle data management and governance* (Tech. Rep. No. PI-0369). Sacramento, CA, USA: California Department of Transportation (Caltrans), Division of Research, Innovation and System Information. Retrieved from https://dot.ca.gov/-/media/dot-media/programs/research-innovation-system-information/documents/preliminary-investigations/pi-0369-final-a11y.pdf (Accessed October 2025)

Carrel, A., Halvorsen, A., & Walker, J. L. (2013). Passengers' perception of and behavioral adaptation to unreliability in public transportation. *Transportation Research Record: Journal of the Transportation Research Board*, *2351*(1), 153–162. Retrieved from https://doi.org/10.3141/2351-17 doi: 10.3141/2351-17

Carrel, A., Sengupta, R., & Walker, J. L. (2017). The san francisco travel quality study: tracking trials and tribulations of a transit taker. *Transportation*, *44*(3), 643–679. Retrieved from https://doi.org/10.1007/s11116-016-9732-4 doi: 10.1007/s11116-016-9732-4

Chen, C., Gong, H., Lawson, C., & Bialostozky, E. (2010). Evaluating the feasibility of a passive travel survey collection in a complex urban environment: Lessons learned from the new york city case study. *Transportation Research Part A: Policy and Practice*, *44*(10), 830-840. Retrieved from https://www.sciencedirect.com/science/article/pii/S0965856410001254 doi: https://doi.org/10.1016/j.tra.2010.08.004

Chen, J., Song, Q., Zhao, C., & Li, Z. (2020). Graph database and relational database performance comparison on a transportation network. In M. Singh, P. K. Gupta, V. Tyagi, J. Flusser, T. Ören, & G. Valentino (Eds.), *Advances in computing and data sciences* (pp. 407–418). Singapore: Springer Singapore. Retrieved from https://link.springer.com/chapter/10.1007/978-981-15-6634-9_37

Chen, X., & Tong, Z. (2025). A new approach for constructing heterogeneous graph databases for public transportation networks. *International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA)*. (Key Laboratory of Urban AI and Green Built Environment of Provincial Higher Education Institutes, Nanjing University)

Chondrogiannis, T., Kalogeraki, V., & Gunopulos, D. (2016). A time-dependent model for road and bus networks using postgis. *Transportation Research Procedia*, *19*, 93–106. doi: 10.1016/j.trpro.2016.12.004

Clifton, K., & Muhs, C. D. (2012). Capturing and representing multimodal trips in travel surveys: Review of the practice. *Transportation Research Record*, *2285*(1), 74–83. Retrieved from https://doi.org/10.3141/2285-09 (Original work published 2012) doi: 10.3141/2285-09

Czerepicki, A. (2016). Timetable-based graph databases for public transport systems. *Transportation Research Procedia*, *14*, 3682–3691. doi: 10.1016/j.trpro.2016.05.444

Dabboussi, A. H., & Jammal, M. (2023, December). Data-driven methods and challenges for intelligent transportation systems in smart cities. *IEEE Internet of Things Magazine*, *6*(4), 68-72. doi: 10.1109/IOTM.001.2300004

Dabiri, S., & Heaslip, K. (2018). Inferring transportation modes from gps trajectories using a convolutional neural network. *Transportation Research Part C: Emerging Technologies*, *86*, 360-371. Retrieved from https://www.sciencedirect.com/science/article/

pii/S0968090X17303509 doi: https://doi.org/10.1016/j.trc.2017.11.021

DeLaurentis, D. (2005). Understanding transportation as a system-of-systems design problem. *43rd AIAA Aerospace Sciences Meeting and Exhibit*. Retrieved from https://arc.aiaa.org/doi/abs/10.2514/6.2005-123 doi: 10.2514/6.2005-123

Delbosc, A., & Currie, G. (2012). Modelling the causes and impacts of personal safety perceptions on public transport ridership. *Transport Policy*, *24*, 302-309. Retrieved from https://www.sciencedirect.com/science/article/pii/S0967070X12001576 doi: https://doi.org/10.1016/j.tranpol.2012.09.009

Dempsey, P. S. (2003). Transportation: A legal history. *Transportation Law Journal*, *30*, 1–38. Retrieved from https://heinonline.org/hol-cgi-bin/get_pdf.cgi?handle=hein.journals/tportl30&section=14

Din, S., Paul, A., & Rehman, A. (2019). 5g-enabled hierarchical architecture for software-defined intelligent transportation system. *Computer Networks*, *150*, 81-89. Retrieved from https://www.sciencedirect.com/science/article/pii/S1389128618312969 doi: https://doi.org/10.1016/j.comnet.2018.11.035

Díaz Erazo, A. D., Raúl Morales Morales, M., Pineda Chávez, V. K., & Leonardo Morales Cardoso, S. (2022, June). Comparative analysis of performance for sql and nosql databases. In *2022 17th iberian conference on information systems and technologies (cisti)* (p. 1-14). doi: 10.23919/CISTI54924.2022.9820292

Eboli, L., & Mazzulla, G. (2009, 05). An ordinal logistic regression model for analysing airport passenger satisfaction. *EuroMed Journal of Business*, *4*(1), 40-57. Retrieved from https://doi.org/10.1108/14502190910956684 doi: 10.1108/14502190910956684

Elkosantini, S., & Darmoul, S. (2013, May). Intelligent public transportation systems: A review of architectures and enabling technologies. In *2013 international conference on advanced logistics and transport* (p. 233-238). doi: 10.1109/ICAdLT.2013.6568465

Fernandes, D., & Bernardino, J. (2018). Graph databases comparison: Allegrograph, arangodb, infinitegraph, neo4j, and orientdb. In *Proceedings of the 7th international conference on data science, technology and applications - data* (p. 373-380). SciTePress. Retrieved from https://www.scitepress.org/PublishedPapers/

2018/69102/69102.pdf doi: 10.5220/0006910203730380

Fonseca, F., Papageorgiou, G., Tondelli, S., Ribeiro, P., Conticelli, E., Jabbari, M., & Ramos, R. (2022). Perceived walkability and respective urban determinants: Insights from bologna and porto. *Sustainability*, *14*(15), 9089. Retrieved from https://doi.org/10.3390/su14159089 doi: 10.3390/su14159089

Fortin, P., Trépanier, M., & Morency, C. (2016). Innovative gtfs data application for transit network analysis using a graph-oriented method. *Journal of Public Transportation*, *19*(4), 18–37. Retrieved from https://doi.org/10.5038/2375-0901.19.4.2 doi: 10.5038/2375-0901.19.4.2

Gatersleben, B., & Uzzell, D. (2007). Affective appraisals of the daily commute: Comparing perceptions of drivers, cyclists, walkers, and users of public transport. *Environment and Behavior*, *39*(3), 416-431. Retrieved from https://doi.org/10.1177/0013916506294032 doi: 10.1177/0013916506294032

Gkiotsalitis, K., & Stathopoulos, A. (2015). A mobile application for real-time multimodal routing under a set of users' preferences. *Journal of Intelligent Transportation Systems*, *19*(2), 149–166. Retrieved from https://doi.org/10.1080/15472450.2013.856712 doi: 10.1080/15472450.2013.856712

Gohar, A., & Nencioni, G. (2021). The role of 5g technologies in a smart city: The case for intelligent transportation system. *Sustainability*, *13*(9), 5188. Retrieved from https://doi.org/10.3390/su13095188 doi: 10.3390/su13095188

Gong, T., Zhu, L., Yu, F. R., & Tang, T. (2023, Sep.). Edge intelligence in intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, *24*(9), 8919-8944. doi: 10.1109/TITS.2023.3275741

Gonzalez, M. C., Hidalgo, C. A., & Barabási, A.-L. (2008). Understanding individual human mobility patterns. *Nature*, *453*(7196), 779–782. doi: 10.1038/nature06958

Goodchild, M. F. (2007). Citizens as sensors: the world of volunteered geography. *GeoJournal*, *69*(4), 211–221. doi: 10.1007/s10708-007-9111-y

Graells-Garrido, E., Opitz, D., Rowe, F., & Arriagada, J. (2023). A data fusion approach with mobile phone data for updating travel survey-based mode split estimates. *Transportation*

Research Part C: Emerging Technologies, *155*, 104285. Retrieved from https://www.sciencedirect.com/science/article/pii/S0968090X23002747 doi: https://doi.org/10.1016/j.trc.2023.104285

Grubler, A. (1990). *The rise and fall of infrastructures: Dynamics of evolution and technological change in transport*. Physica-Verlag. Retrieved from https://pure.iiasa.ac.at/id/eprint/3351/1/XB-90-704.pdf

Grubler, A., & Nakicenovic, N. (1991). *The evolution of transport systems: Past and future* (Tech. Rep. No. RR-91-008). International Institute for Applied Systems Analysis (IIASA). Retrieved from https://pure.iiasa.ac.at/id/eprint/3486/1/RR-91-008.pdf (Research Report)

Guan, H., Xue, J., Zhang, Y., Chang, F., & Liu, W. (2025). Examining the relationship between commuting time, academic achievement, and mental health in rural china: a cross-sectional analysis. *BMC Public Health*, *25*(1), 1616. Retrieved from https://doi.org/10.1186/s12889-025-22861-7 doi: 10.1186/s12889-025-22861-7

Harding, C. (2019). *From smartphone apps to in-person data collection: Modern and cost-effective multimodal travel data collection for evidence-based planning* (Doctoral dissertation, University of Toronto). Retrieved from https://search.proquest.com/openview/57f112edbe28563ff61da7df8707c43c/1?pq-origsite=gscholar&cbl=18750&diss=y

Harding, C., Faghih Imani, A., Srikukenthiran, S., & Miller, E. J. (2021). Are we there yet? assessing smartphone apps as full-fledged tools for activity-travel surveys. *Transportation*. Retrieved from https://link.springer.com/article/10.1007/s11116-020-10135-7 doi: 10.1007/s11116-020-10135-7

Hosseini, S. H. (2025). *Trip phase recognition and transport mode classification through mobile sensing technologies* (Doctoral dissertation, sapienza university of rome). Retrieved from https://tesidottorato.depositolegale.it/bitstream/20.500.14242/188589/1/Tesi_dottorato_Hosseini.pdf (PhD Thesis, accessed 2025-09-04)

Hrnčíř, J., & Jakob, M. (2013, Oct). Generalised time-dependent graphs for fully multimodal

journey planning. In *16th international ieee conference on intelligent transportation systems (itsc 2013)* (p. 2138-2145). doi: 10.1109/ITSC.2013.6728545

Huang, H., Bucher, D., Kissling, J., Weibel, R., & Raubal, M. (2019, Sep.). Multimodal route planning with public transport and carpooling. *IEEE Transactions on Intelligent Transportation Systems*, *20*(9), 3513-3525. doi: 10.1109/TITS.2018.2876570

Huang, Y., Li, D., Han, B., Xu, E., & Medeossi, G. (2024). Multimodal transportation recommendation: Embedding travel intention and transit reliability by heterogeneous graph attention network. *Expert Systems with Applications*, *255*, 124579. Retrieved from https://www.sciencedirect.com/science/article/pii/S0957417424014465 doi: https://doi.org/10.1016/j.eswa.2024.124579

Huang, Y., Xiao, Z., Wang, D., Jiang, H., & Wu, D. (2020, Dec). Exploring individual travel patterns across private car trajectory data. *IEEE Transactions on Intelligent Transportation Systems*, *21*(12), 5036-5050. doi: 10.1109/TITS.2019.2948188

Ibrahim, A. N. H., & Borhan, M. N. (2020, October). The interrelationship between perceived quality, perceived value and user satisfaction towards behavioral intention in public transportation: A review of the evidence. *International Journal of Advanced Science, Engineering and Information Technology*, *10*(5), 2048–2056. Retrieved from https://doi.org/10.18517/ijaseit.10.5.12818 doi: 10.18517/ijaseit.10.5.12818

Jain, A., & Handy, S. (2025). How does commute well-being affect life satisfaction? evidence from uc davis. *Transportation Research Part F: Traffic Psychology and Behaviour*, *115*, 103340. Retrieved from https://www.sciencedirect.com/science/article/pii/S1369847825002955 doi: https://doi.org/10.1016/j.trf.2025.103340

Jiang, H., Zhang, Y., Xiao, Z., Zhao, P., & Iyengar, A. (2021, Jan). An empirical study of travel behavior using private car trajectory data. *IEEE Transactions on Network Science and Engineering*, *8*(1), 53-64. doi: 10.1109/TNSE.2020.3025529

Joshi, V. D., Agarwal, P., Kumar, A., Dogra, N., & Nandan, D. (2024). Urban odyssey: "pioneering multimodal routes for tomorrow's smart cities". *Measurement: Sensors*, *36*, 101301. Retrieved from https://www.sciencedirect.com/science/article/pii/S2665917424002770 doi: https://doi.org/10.1016/j.measen.2024.101301

Kaasinen, E. (2003). User needs for location-aware mobile services. *Personal and Ubiquitous Computing*, *7*(1), 70–79. Retrieved from https://doi.org/10.1007/s00779-002-0214-7 doi: 10.1007/s00779-002-0214-7

Kamargiannis, K., Pappelis, D., Stebbins, S., Song, F., Tsouros, I., Fermi, F., ... Kamargianni, M. (2023). Advancing travel demand surveys using a new generation smartphone-based platform: Applications and users' experience. *Transportation Research Procedia*, *null*, null. Retrieved from https://www.semanticscholar.org/paper/9131faa7a679a532da9af8bccb081f94101c39d4 doi: 10.1016/j.trpro.2023.11.822

Khan, R. A., Khan, S. U., Khan, H. U., & Ilyas, M. (2021). Systematic mapping study on security approaches in secure software engineering. *IEEE Access*, *9*, 19139-19160. doi: 10.1109/ACCESS.2021.3052311

Kim, E. J., Won, J., & Kim, J. (2019). Is seoul walkable? assessing a walkability score and examining its relationship with pedestrian satisfaction in seoul, korea. *Sustainability*, *11*(24), 6915. Retrieved from https://doi.org/10.3390/su11246915 doi: 10.3390/su11246915

Klein, J., Gorton, I., Ernst, N., Donohoe, P., Pham, K., & Matser, C. (2015). Performance evaluation of nosql databases: A case study. In *Proceedings of the 1st workshop on performance analysis of big data systems* (p. 5–10). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/2694730.2694731 doi: 10.1145/2694730.2694731

Koelle, M., Ananthanarayan, S., & Boll, S. (2020). Social acceptability in hci: A survey of methods, measures, and design strategies. In *Proceedings of the 2020 chi conference on human factors in computing systems* (p. 1–19). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3313831.3376162 doi: 10.1145/3313831.3376162

Kousis, A. (2023). Managing smart city linked data: A graph databases integrative literature review. In *Managing and mining graph data.* CRC Press. Retrieved from https://www.taylorfrancis.com/chapters/edit/10.1201/9781003183532-6 doi: 10

.1201/9781003183532-6

Küçükkeçeci, C., & Yazıcı, A. (2017). A graph-based big data model for wireless multimedia sensor networks. In P. Angelov, Y. Manolopoulos, L. Iliadis, A. Roy, & M. Vellasco (Eds.), *Advances in big data* (pp. 205–215). Cham: Springer International Publishing. Retrieved from https://link.springer.com/chapter/10.1007/978-3-319-47898-2_22

Kulak, D., & Guiney, E. (2012). *Use cases: Requirements in context*. Pearson Education. Retrieved from https://books.google.ca/books?id=cQ-QmAnu0HUC

Küçükkeçeci, C., & Yazıcı, A. (2018). Big data model simulation on a graph database for surveillance in wireless multimedia sensor networks. *Big Data Research*, *11*, 33-43. Retrieved from https://www.sciencedirect.com/science/article/pii/S2214579617300102 (Selected papers from the 2nd INNS Conference on Big Data: Big Data & Neural Networks) doi: https://doi.org/10.1016/j.bdr.2017.09.003

Lane, N. D., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., & Campbell, A. T. (2010, 09). A survey of mobile phone sensing. *IEEE Communications Magazine*, *48*(9), 140-150. doi: 10.1109/MCOM.2010.5560598

Lemonde, C., Arsenio, E., & Henriques, R. (2021). Integrative analysis of multimodal traffic data: addressing open challenges using big data analytics in the city of lisbon. *European Transport Research Review*, *13*(64), 1–15. Retrieved from https://doi.org/10.1186/s12544-021-00520-3 doi: 10.1186/s12544-021-00520-3

Li, D., Houghton, J. D., Li, X., Peng, Q., Li, J., & Zou, W. (2025). The relationship between commuting stress and nurses' well-being: Considering gender differences. *Journal of Nursing Management*, *2025*, 4414417. Retrieved from https://doi.org/10.1155/jonm/4414417 doi: 10.1155/jonm/4414417

Liu, H., Tong, Y., Zhang, P., Lu, X., Duan, J., & Xiong, H. (2019). Hydra: A personalized and context-aware multi-modal transportation recommendation system. In *Proceedings of the 25th acm sigkdd international conference on knowledge discovery & data mining* (p. 2314–2324). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3292500.3330660 doi: 10.1145/3292500.3330660

Liu, Y., Kang, C., Gao, S., Xiao, Y., & Tian, Y. (2012). Understanding intra-urban trip patterns from taxi trajectory data. *Journal of geographical systems*, *14*(4), 463–483. Retrieved from https://link.springer.com/article/10.1007/s10109-012-0166 doi: https://doi.org/10.1007/s10109-012-0166-z

Lu, Y., Misra, A., Sun, w., & Wu, H. (2018, April). Smartphone sensing meets transport data: A collaborative framework for transportation service analytics. *IEEE Transactions on Mobile Computing*, *17*(4), 945-960. doi: 10.1109/TMC.2017.2743176

Macioszek, E. (2014). Integration of traffic management systems in urban transport. *Archives of Transport*. Retrieved from https://bibliotekanauki.pl/articles/393705.pdf

Maduako, I., Cavalheri, E., & Wachowicz, M. (2018). *Exploring the use of time-varying graphs for modelling transit networks.* Retrieved from https://arxiv.org/abs/1803.07610

Maduako, I. D., Wachowicz, M., & Hanson, T. (2019). Transit performance assessment based on graph analytics. *Transportmetrica A: Transport Science*, *15*(2), 1382–1401. Retrieved from https://doi.org/10.1080/23249935.2019.1596991 doi: 10.1080/23249935.2019.1596991

Maelstrom, D. (2021). *Digital maelstrom. (2021). digital maelstrom.* Retrieved from https://www.digitalmaelstrom.net/development/secure-software-development/

Mahrez, Z., Sabir, E., Badidi, E., Saad, W., & Sadik, M. (2021, 06). Smart urban mobility: When mobility systems meet smart data. *IEEE Transactions on Intelligent Transportation Systems*, *PP*, 1-18. doi: 10.1109/TITS.2021.3084907

Maslich, & Voronova. (2024). *Development of metro transport systems in megacities.* Retrieved from https://dspace.khadi.kharkov.ua/bitstreams/8d27bb43-5610-42c0-9f44-5fa2ec402816/download

Microsoft Corporation. (2025). *Graph and relational databases in microsoft fabric.* https://learn.microsoft.com/en-us/fabric/graph/graph-relational-databases. Microsoft Learn. Retrieved from https://learn.microsoft.com/en-us/fabric/graph/graph-relational-databases

Moleman, M. L., & Kroesen, M. (2025). Revealing accessibility disparities: A latent class analysis linking objective and subjective accessibility measures. *Transportation Research Part A: Policy and Practice*, *192*, 104341. Retrieved from https://www.sciencedirect.com/science/article/pii/S0965856424003896 doi: https://doi.org/10.1016/j.tra.2024.104341

Montoya-Torres, J. R., Moreno, S., Guerrero, W. J., & Mejía, G. (2021). Big data analytics and intelligent transportation systems**this work was supported by universidad de la sabana, colombia (grant ingphd-10-2019). *IFAC-PapersOnLine*, *54*(2), 216-220. Retrieved from https://www.sciencedirect.com/science/article/pii/S2405896321004614 (16th IFAC Symposium on Control in Transportation Systems CTS 2021) doi: https://doi.org/10.1016/j.ifacol.2021.06.025

MotionTag GmbH. (2025). *Technology and compliance.* https://www.motiontag.com/tech-compliance/technology-motiontag. (Accessed: 2025-09-04)

Mystakidis, A. (2023). Graph databases in smart city applications: Using neo4j and machine learning for energy load forecasting. In *Managing and mining graph data.* CRC Press. Retrieved from https://www.taylorfrancis.com/chapters/edit/10.1201/9781003183532-7 doi: 10.1201/9781003183532-7

Namoun, A., Tufail, A., Mehandjiev, N., Alrehaili, A., Akhlaghinia, J., & Peytchev, E. (2021). An eco-friendly multimodal route guidance system for urban areas using multi-agent technology. *Applied Sciences*, *11*(5), 2057. Retrieved from https://www.mdpi.com/2076-3417/11/5/2057 doi: 10.3390/app11052057

Nasim, R., & Kassler, A. (2012, Dec). Distributed architectures for intelligent transport systems: A survey. In *2012 second symposium on network cloud computing and applications* (p. 130-136). Retrieved from https://ieeexplore.ieee.org/abstract/document/6472469/ doi: 10.1109/NCCA.2012.15

Neo4j. (2024). *Graph database vs relational database: What's the difference?* Retrieved from https://neo4j.com/blog/graph-database/graph-database-vs-relational-database/ (Accessed: 2025-10-11)

Neo4j, Inc. (2025). *Neo4j graph database platform.* Retrieved from https://neo4j.com/

(Accessed: 2025-09-07)

Neo4j, Inc. (2026). *Understanding neo4j's data on disk.* Retrieved from https://neo4j.com/developer/kb/understanding-data-on-disk

Novak, A., Novak Sedlackova, A., Vochozka, M., & Popescu, G. H. (2022). Big data-driven governance of smart sustainable intelligent transportation systems: Autonomous driving behaviors, predictive modeling techniques, and sensing and computing technologies. *Contemporary Readings in Law and Social Justice*, *14*(2), 100–117. Retrieved from https://addletonacademicpublishers.com/contents-crlsj/2599-volume-14-2-2022/4358-big-data-driven-governance-of-smart-sustainable-intelligent-transportation-systems-autonomous-driving-behaviors-predictive-modeling-techniques-and-sensing-and-computing-technologies doi: 10.22381/CRLSJ14220226

Oka, D. K. (2021). *Building secure cars: Assuring the automotive software development lifecycle*. CRC Press. Retrieved from https://books.google.com/books?hl=en&lr=&id=TD8gEAAAQBAJ&oi=fnd&pg=PR11&dq=secure+software+development+life+cycle+ssdlc+transport+systems+mobility+feedback+security (Accessed: 2025-09-07)

Ondiek, R. (2024, 04). *Secure software development lifecycle (sdlc): Best practices by reagan ondiek.* https://www.researchgate.net/profile/Reagan-Ondiek/publication/391981745_Secure-Software-Development-Lifecycle-SDLC-Best-Practices_By_Reagan-Ondiek/links/682f48b9be1b507dce8df998/Secure-Software-Development-Lifecycle-SDLC-Best-Practices-By-Reagan-Ondiek.pdf. (Accessed: 2025-09-07)

OrientDB Community. (2025). *Orientdb: Multi-model database.* Retrieved from https://orientdb.dev/ (Accessed: 2025-09-08)

Park, S., & Cheng, T. (2023). Framework for constructing multimodal transport networks and

routing using a graph database: A case study in london. *Transactions in GIS*, *27*(5), 1391–1417. Retrieved from https://doi.org/10.1111/tgis.13071 doi: /10.1111/tgis.13071

Patterson, Z., Fitzsimmons, K., Jackson, S., & Mukai, T. (2019). Itinerum: The open smartphone travel survey platform. *SoftwareX*, *10*, 100230. Retrieved from https://www.semanticscholar.org/paper/8f594608d15f478daec99e376163382ae0b04a11 doi: 10.1016/J.SOFTX.2019.04.002

Pawar, V., Zade, N., Vora, D., Khairnar, V., Oliveira, A., Kotecha, K., & Kulkarni, A. (2024). Intelligent transportation system with 5g vehicle-to-everything (v2x): Architectures, vehicular use cases, emergency vehicles, current challenges, and future directions. *IEEE Access*, *12*, 183937-183960. doi: 10.1109/ACCESS.2024.3506815

Peyman, M., Copado, P. J., Tordecilla, R. D., Martins, L. d. C., Xhafa, F., & Juan, A. A. (2021). Edge computing and iot analytics for agile optimization in intelligent transportation systems. *Energies*, *14*(19), 6309. Retrieved from https://doi.org/10.3390/en14196309 doi: 10.3390/en14196309

Picone, M., Busanelli, S., Amoretti, M., Zanichelli, F., & Ferrari, G. (2015). Advanced technologies for intelligent transportation systems. *Intelligent systems reference library*. Retrieved from https://doi.org/10.1007/978-3-319-10668-7 doi: 10.1007/978-3-319-10668-7

Prelipcean, A. C., Gidófalvi, G., & Susilo, Y. O. (2016). Transportation mode detection – an in-depth review of applicability and reliability. *Transport Reviews*, *37*(4), 442–464. Retrieved from https://doi.org/10.1080/01441647.2016.1246489 doi: 10.1080/01441647.2016.1246489

Sandberg, B., Hurmerinta, L., Helminen, V., Leino, H., & Vasankari, T. (2024, 07). A new tramway and the formation of emotional commuting experiences. *Transportation Research Part F: Traffic Psychology and Behaviour*, *104*, 31-41. doi: 10.1016/j.trf.2024.05.018

Sandell, J., Asplund, E., Ayele, W. Y., & Duneld, M. (2024). *Performance comparison analysis of arangodb, mysql, and neo4j: An experimental study of querying connected data.* Retrieved

from https://arxiv.org/abs/2401.17482

Santos López, E. G., Félix Melchor Santos De La Cruz. (2015). Literature review about neo4j graph database as a feasible alternative for replacing rdbms. *Industrial Data*. Retrieved from https://www.redalyc.org/articulo.oa?id=81643819017

Saroj, A., Roy, S., Guin, A., Hunter, M., & Fujimoto, R. (2018, July 2). Smart city real-time data-driven transportation simulation. In *Wsc 2018 - 2018 winter simulation conference* (pp. 857–868). Institute of Electrical and Electronics Engineers Inc. (Publisher Copyright: © 2018 IEEE; 2018 Winter Simulation Conference, WSC 2018 ; Conference date: 09-12-2018 Through 12-12-2018) doi: 10.1109/WSC.2018.8632198

Schema App. (2023). *Relational databases vs graph databases: What's the difference?* Retrieved from https://www.schemaapp.com/schema-markup/relational-databases-vs-graph-databases/ (Accessed: 2025-10-11)

Schuessler, N., & Axhausen, K. W. (2009). Processing raw data from global positioning systems without additional information. *Transportation Research Record*, *2105*(1), 28–36. doi: 10.3141/2105-04

Serin, F., Mete, S., Gül, M., & Celik, E. (2018, 09). Mapping between relational database management systems and graph database for public transportation network.. Retrieved from https://www.researchgate.net/publication/332057324_Mapping_Between_Relational_Database_Management_Systems_And_Graph_Database_For_Public_Transportation_Network

Servizi, V., Pereira, F. C., Anderson, M. K., & Nielsen, O. A. (2021). Transport behavior-mining from smartphones: a review. *European Transport Research Review*. Retrieved from https://link.springer.com/article/10.1186/s12544-021-00516-z doi: 10.1186/s12544-021-00516-z

Shaheen, S. (2016). *Mobile apps and transportation: A review of smartphone apps and a study of user response to multimodal traveler information* (Tech. Rep.). University of California, Berkeley, Transportation Sustainability Research Center. Retrieved from https://escholarship.org/content/qt6m332192/qt6m332192.pdf (eScholarship, University of California)

Sharma, S., & Awasthi, A. (2022). Introduction to intelligent transportation system: Overview, classification based on physical architecture and challenges. *ResearchGate Preprint*. Retrieved from https://www.researchgate.net/profile/Shivani_Sharma144/publication/360541670

Shen, L., & Stopher, P. R. (2014). Review of gps travel survey and gps data-processing methods. *Transport Reviews*, *34*(3), 316–334. doi: 10.1080/01441647.2014.903530

Shibanova, D. A., Stroganov, I. V., & Rudakov, I. V. (2021, Jan). Data formalization in transport system modeling using a graph database. In *2021 ieee conference of russian young researchers in electrical and electronic engineering (elconrus)* (p. 2245-2251). doi: 10.1109/ElConRus51938.2021.9396137

Signer Del Cid, A. M. (2024). *Exploring mobile map app usage: A geospatial study of touch-screen interactions and contextual influences* (Master's thesis, University of Zurich, Faculty of Science). (Masters Thesis) doi: 10.5167/uzh-275941

Singleton, P. A. (2020). Multimodal travel-based multitasking during the commute: Who does what? *International Journal of Sustainable Transportation*. Retrieved from https://www.tandfonline.com/doi/abs/10.1080/15568318.2018.1536237 doi: 10.1080/15568318.2018.1536237

Smarzaro, R., Davis, C. A., & Quintanilha, J. A. (2021). Creation of a multimodal urban transportation network through spatial data integration from authoritative and crowdsourced data. *ISPRS International Journal of Geo-Information*, *10*(7). Retrieved from https://www.mdpi.com/2220-9964/10/7/470 doi: 10.3390/ijgi10070470

Sodhro, A. H., Pirbhulal, S., Sodhro, G. H., Muzammal, M., Zongwei, L., Gurtov, A., ... de Albuquerque, V. H. C. (2021, Aug). Towards 5g-enabled self adaptive green and reliable communication in intelligent transportation system. *IEEE Transactions on Intelligent Transportation Systems*, *22*(8), 5223-5231. doi: 10.1109/TITS.2020.3019227

Sousa, S., Santos, A., Costa, A., Dias, B., Ribeiro, B., Gonçalves, F., ... Óscar Gama (2017). A new approach on communications architectures for intelligent transportation systems. *Procedia Computer Science*, *110*, 320-327. Retrieved from https://www.sciencedirect

.com/science/article/pii/S1877050917312796 (14th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2017) / 12th International Conference on Future Networks and Communications (FNC 2017) / Affiliated Workshops) doi: https://doi.org/10.1016/j.procs.2017.06.101

Stonebraker, M., & Çetintemel, U. (2005). "one size fits all": An idea whose time has come and gone. In *Proceedings of the 21st international conference on data engineering (icde)* (pp. 2–11). doi: 10.1109/ICDE.2005.1

Stopher, P. R., Daigler, V., & Griffith, S. (2018). Smartphone app versus gps logger: A comparative study. *Transportation Research Procedia*, *32*, 135–145. (Transport Survey Methods in the Era of Big Data: Facing the Challenges) doi: 10.1016/j.trpro.2018.10.026

Suh, W., Henclewood, D., Guin, A., & Guensler, R. (2017). Dynamic data driven transportation systems. *Multimedia Tools and Applications*, *76*, 25253–25269. Retrieved from https://doi.org/10.1007/s11042-016-4318-x doi: 10.1007/s11042-016-4318-x

Sukhov, A., Olsson, L. E., & Friman, M. (2022). Necessary and sufficient conditions for attractive public transport: Combined use of pls-sem and nca. *Transportation Research Part A: Policy and Practice*, *158*, 239-250. Retrieved from https://www.sciencedirect.com/science/article/pii/S0965856422000593 doi: https://doi.org/10.1016/j.tra.2022.03.012

Sumicad, R., Palanas, K. V. G., Ebesa, A. B., Sesante, M. A., Nunez, K. A., Pacaide, A., & Navarro, H. L. D. (2024). Student commuters' perceived level of stress. *Journal of Psychology and Behavior Studies*, *4*(2), 58–92. Retrieved from https://www.proquest.com/openview/1420b093ab61d81d16dc4fd51732286d/1?pq-origsite=gscholar&cbl=7226557# doi: 10.32996/jpbs.2024.4.2.6

Tareq, A., Isaac, O., Bilal, F., & Zachary, P. (2025). An investigation using automated travel scenarios from semi-passive travel diary recording. *Transportation Research Procedia*.

The PostgreSQL Global Development Group. (2024). *Postgresql documentation: Indexes.* Retrieved from https://www.postgresql.org/docs/current/indexes.html (Accessed: 2025-10-15)

Tjortjis, C. (2021). *Managing and mining graph data*. CRC Press, Taylor & Francis. Retrieved from https://api.taylorfrancis.com/content/books/mono/download?identifierName=doi&identifierValue=10.1201/9781003183532&type=googlepdf doi: 10.1201/9781003183532

Vereycken, J. (2016). *Investigating graph-based storage backends for hypermedia link services* (Master's Thesis, Vrije Universiteit Brussel). Retrieved from https://wise.vub.ac.be/sites/default/files/theses/ThesisJonasVereycken.pdf

Verma, R., Ghosh, S., Saketh, M., Ganguly, N., Mitra, B., & Chakraborty, S. (2018). Comfride: a smartphone based system for comfortable public transport recommendation. In *Proceedings of the 12th acm conference on recommender systems* (p. 181–189). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3240323.3240359 doi: 10.1145/3240323.3240359

Virić, D., & Pantelić, O. (2023). Comparative analysis of graph-based nosql databases. *Deutsche Internationale Zeitschrift für Zeitgenössische Wissenschaft*, *67*, 77–88. Retrieved from https://dizzw.com/wp-content/uploads/2023/11/Deutsche-internationale-Zeitschrift-f%C3%BCr-zeitgen%C3%B6ssische-Wissenschaft-%E2%84%9667-2023.pdf#page=77

Vyawahare, H. (2024). Exploring the hybrid approach: Integrating relational and graph databases for enhanced data management. In *Lecture notes in computer science*. Springer. Retrieved from https://link.springer.com/chapter/10.1007/978-3-031-74701-4_13

Webber, J., Robinson, I., & Eifrem, E. (2012). *Graph databases*. O'Reilly Media. Retrieved from https://neo4j.com/graph-databases-book/

Wirawan, P., Riyanto, D., Nugraheni, D., & Yasmin, Y. (2019). Graph database schema for multimodal transportation in semarang. *Journal of Information Systems Engineering and Business Intelligence*, *5*(2), 163–170. Retrieved from https://doi.org/10.20473/jisebi.5.2.163-170 doi: 10.20473/jisebi.5.2.163-170

Xu, H., Lin, J., & Yu, W. (2017). Smart transportation systems: Architecture, enabling technologies,

and open issues. In Y. Sun & H. Song (Eds.), *Secure and trustworthy transportation cyber-physical systems* (pp. 23–49). Singapore: Springer Singapore. Retrieved from https://doi.org/10.1007/978-981-10-3892-1_2 doi: 10.1007/978-981-10-3892-1_2

Xue, J., Tan, R., Ma, J., Li, Z., & Wang, C. (2025). Data science in transportation networks with graph neural networks: A review and outlook. *Data Science in Transportation*, *7*(1), 10. Retrieved from https://doi.org/10.1007/s42421-025-00124-6 doi: 10.1007/s42421-025-00124-6

Yang, J., He, M., He, M., & Peng, L. (2025). Investigating the associations of commuting behavior, perception, satisfaction, and subjective well-being: An analytical framework from the perspective of commuters' preferences and tolerance toward commuting time. *Journal of Transport and Health*, *44*, 102140. Retrieved from https://www.sciencedirect.com/science/article/pii/S2214140525001604 doi: https://doi.org/10.1016/j.jth.2025.102140

Yu, R., Cheung, O., Lau, K., & Woo, J. (2017). Associations between perceived neighborhood walkability and walking time, wellbeing, and loneliness in community-dwelling older chinese people in hong kong. *International Journal of Environmental Research and Public Health*, *14*(10), 1199. Retrieved from https://doi.org/10.3390/ijerph14101199 doi: 10.3390/ijerph14101199

Zhang, H., Zhang, L., Liu, Y., & Zhang, L. (2023). Understanding travel mode choice behavior: Influencing factors analysis and prediction with machine learning method. *Sustainability*, *15*(14), 11414. Retrieved from https://doi.org/10.3390/su151411414 doi: 10.3390/su151411414

Zhang, J., & Letaief, K. B. (2020, Feb). Mobile edge intelligence and computing for the internet of vehicles. *Proceedings of the IEEE*, *108*(2), 246-261. doi: 10.1109/JPROC.2019.2947490

Zhang, Q., Ma, Z., Zhang, P., & Jenelius, E. (2025). Mobility knowledge graph: review and its application in public transport. *Transportation*, *52*(3), 1119–1145. Retrieved from https://link.springer.com/article/10.1007/s11116-023-10451-8 doi: https://doi.org/10.1007/s11116-023-10451-8

Zhang, X., & Ma, L. (2024). Impact of commuting on mental well-being: Using time-stamped subjective and objective data. *Transportation Research Part F: Traffic Psychology and Behaviour*, *107*, 395-412. Retrieved from https://www.sciencedirect.com/science/article/pii/S1369847824002602 doi: https://doi.org/10.1016/j.trf.2024.09.009

Zheng, Y., Li, Q., Chen, Y., Xie, X., & Ma, W.-Y. (2008). Understanding mobility based on gps data. In *Proceedings of the 10th international conference on ubiquitous computing* (pp. 312–321). doi: 10.1145/1409635.1409677

Zhou, X., Ke, R., Yang, H., & Liu, C. (2021). When intelligent transportation systems sensing meets edge computing: Vision and challenges. *Applied Sciences*, *11*(20), 9680. Retrieved from https://doi.org/10.3390/app11209680 doi: 10.3390/app11209680

Zhou, X., Liang, W., Wang, K. I.-K., Yada, K., Yang, L. T., Ma, J., & Jin, Q. (2025, June). Decentralized federated graph learning with lightweight zero trust architecture for next-generation networking security. *IEEE Journal on Selected Areas in Communications*, *43*(6), 1908-1922. doi: 10.1109/JSAC.2025.3560012