FMDB *Transactions on Sustainable Intelligent Networks*

Leveraging Graph Databases for Real-Time Analytics in Data Engineering Pipelines

Prasad Sundaramoorthy[1,*]

[1]Department of Data and AI Analytics, Nordstrom, Washington, Seattle, United States of America.
prasad.sundaramoorthy@nordstrom.com[1]

Abstract: Relational databases are struggling to meet real-time analytics expectations as data grows and complexity. This article examines whether graph databases can alleviate concerns in data engineering pipelines. Data is stored as nodes, edges, and characteristics in graph databases, which model complex data point connections better. Social websites, fraud detection, and recommendation systems benefit from this design's fast, flexible data searching. Real-time data engineering and analytics enable organizations to process and analyze data for fast insights and informed decision-making. The incorporation of graph databases into data pipelines can enable real-time analysis by speeding up query responses and schema modification. The study utilizes a 100,000-transaction e-commerce dataset, which includes user ID, product ID, amount, and timestamp, to demonstrate the real-time functionality of graph databases. Neo4j is compared to MySQL to examine how graph databases handle real-time analytics, such as fraud detection. Graph databases enhance real-time analytics by accelerating response times and minimizing resource requirements for complex queries, according to tests. Finally, the report discusses how graph databases in data pipelines can influence future R&D.

Keywords: Graph Databases; Real-Time Analytics; Data Engineering; Data Pipelines; Performance Optimization; Customer Experience; Business Organizations; Data Management.

Received on: 01/08/2024, **Revised on:** 11/10/2024, **Accepted on:** 22/11/2024, **Published on:** 07/03/2025

Journal Homepage: https://www.fmdbpub.com/user/journals/details/FTSIN

DOI: https://doi.org/10.69888/FTSIN.2025.000367

Cite as: P. Sundaramoorthy, “Leveraging Graph Databases for Real-Time Analytics in Data Engineering Pipelines,” *FMDB Transactions on Sustainable Intelligent Networks*., vol. 2, no. 1, pp. 22–32, 2025.

1. Introduction

The digital era has witnessed record levels of data generated from a vast array of sources, including social networks, IoT sensors, online transactional systems, and other sources. As more devices and systems become connected, the volume of generated data has never been greater. Consequently, the need for quicker and superior means of processing and analyzing data in real-time is becoming increasingly crucial. Business organisations in all industries are now being forced to extract real-time insights from the torrent of data to make informed decisions, enhance customer experience, and optimise operations. Even mature data management and analysis methods, particularly those based on relational databases, struggle to keep up with the dynamic and highly interconnected nature of today's datasets. The static tabular model of relational databases was designed with structured data in mind and does not accommodate the dynamic, flexible, and interconnected relationships that are now at the center of the data being generated. Thus, they cannot effectively handle huge groups of data, resulting in a lower query rate

[*]Corresponding author.

and higher operational expenses for providing real-time analysis. It is this lack that has led to innovative solutions in data engineering, and graph databases are proving to be very promising. Graph databases are better suited to handle the interrelatedness and relationships of data points. They are therefore extremely well-positioned for these highly interrelated, rich sets of data common in social media environments, IoT devices, and online shopping environments.

Information in graph databases is categorised by node (things or entities) and edge (relationship) types; thus, relational querying is quick, easy, and extremely efficient with huge, even very complex datasets. Graph databases offer several advantages over traditional relational databases, particularly when scalability, flexibility, and real-time analytics capabilities are top priorities. Firms can make it easier to identify patterns, dependencies, and trends within their data by utilising graph-based models, and thus enabling them to draw useful conclusions faster and more accurately. Consequently, graph databases are increasingly being used in new data pipelines for engineering, where they facilitate real-time streams of data as well as advanced analytics applications such as recommendation engines, fraud analysis, social network analysis, and predictive maintenance. The shift to graph databases marks a departure from conventional database management systems, driven by the growing demand for intelligent, adaptive methods to analyse and manage the vast volumes of data generated in the digital era. This paradigm in data engineering is likely to remain a fixture as the demand for real-time, actionable information continues to grow across various business segments [1].

Graph databases are NoSQL databases that store data in the form of a graph, comprising nodes, edges, and attributes to describe entities and their connections, as opposed to relational databases, which organize data in tables and rows. Graph databases instantly mirror the shape of data points. This is particularly useful in systems where relationships are data value itself, e.g., in recommendation systems, fraud detection systems, and social networks [2]. Real-time analytics in data engineering involves processing and analyzing incoming data in real-time, providing prompt insights and enabling rapid decision-making. Batch processing, where data is collected and processed at some interval, fails to address the requirements of applications that require immediate, dynamic feedback. Real-time analytics enables organisations to react to data as it is received in the system, which applications like fraud detection, personalised recommendations, and operational monitoring require [3].

The application of graph databases in pipelines offers a wide range of benefits, enabling real-time analysis. To begin with, graph databases are ideally designed to query and handle complex relationships, and this makes them well-suited for analyzing interrelated information. For instance, for an anti-fraud solution, graph databases can be applied to query account, transaction, and user relationships to detect anomalies [4]. Second, graph databases are schema-agnostic in design, allowing data engineers to rebuild the graph structure whenever a new relation is discovered. Flexibility is needed in systems whose sources of information are constantly changing [5]. However, even with the benefits that graph databases are endowed with, integration into the present legacy pipeline is not free from constraints [7].

One of the key challenges is achieving consistency across distributed systems, particularly when data is ingested from various sources in real-time [6]. Another issue is improving query performance, particularly on large data with complex graph traversals [7]. This article discusses the efficient use of graph databases in real-time analytics within data engineering processes, examining both the benefits and drawbacks [8]. In addition, greater use of graph databases reflects broader trends within data engineering as more focus is placed on those tools that can manage complex, related information [9]. Researchers have termed the ability of graph databases to transcend the limitations of traditional data storage systems in the context of analyzing large and complex networks [10]. In particular, the use of these databases has been instrumental in developing recommendation engines, where quick querying and analysis of relationships are critical to delivering personalized experiences [11].

The advent of graph databases in today's data engineering workflows also coincides with the increasing amount of real-time data available from various sources, which enables extensible and versatile analytics platforms [12]. However, to harness the maximum potential of the systems, organisations need to tackle technical issues in installing and integrating the systems [13]. Thus, research continues to investigate new approaches to improving the performance and scalability of graph databases for real-time systems, particularly in applications such as fraud detection, social network analysis, and the Internet of Things [14]. This study will aim to provide an in-depth analysis of the potential of graph databases to complement real-time analytics within data engineering pipelines. The paper will critique existing literature on the subject, outline the methodology applied in using graph databases, and provide results of a comparison of the performance of graph databases versus traditional relational databases. Finally, the study will provide implications of the findings and propose recommendations for future practice and research.

2. Review of Literature

Trivedi et al. [1] introduced graph databases, which are today well recognized for their distinctive ability to represent and model complex relationships between data points. The databases are preferably utilized where one desires to know the relationship between entities. In comparison to traditional relational databases, which are based on predefined tables and schemas for storing data, graph databases operate on a graph model centered on nodes (entities) and edges (the relationships between entities). This can make querying swifter and more responsive, especially when data related to an association is involved. Graph databases originally served well in areas such as social network analysis. Graph databases exchanged data on community structure, influencers, and trends. Graph databases were applied in other contexts as more studies were conducted on them.

Chowdhary [2] applied graph databases in detecting fraud, where one needs to find and understand the relationships between entities, such as users, transactions, and devices, to detect and identify anomalies and prevent fraud. Graph databases enable these relations to be depicted in real-time and offer a more dynamic and reactive solution than other methods. Graph databases have also been widely used within recommendation systems, where they model user interaction and preference data to recommend products, services, or content. By capturing action interactions with items and users, graph databases enable highly personalised recommendations based on behavioural trends that might be difficult to discover with relational databases. The ability of graph databases to model associated points of data innately makes them a goldmine in most sectors. They have also been researched by healthcare industry professionals, who use them to simulate relationships between patients, treatments, and medical histories. Graph database use is expanding increasingly widely.

Eisenstein [3] developed techniques for graph database optimisation to be used for network optimisation. Route and network optimisation in supply chain logistics, telecommunications, and other industries will be based on an understanding of the node connectivity within a network. Graph databases are ideally suited to represent complex networks like these, and it is easier to model and optimise routes, identify bottlenecks, and increase efficiency as a general principle. Their ability to handle tremendous amounts of connected information in real-time makes them suitable for handling networks with real-time dynamics. Graph databases are utilized in several industries, from transportation networks to streamline traffic flows and alleviate bottlenecks. Graph databases possess strength since they can generate real-time intelligence on how various components of a complete network relate to each other. As organisations increasingly adopt graph technology across various industries, efficiency and performance are enhanced.

Wang et al. [6] highlighted the scalability of graph databases in handling vast amounts of data and their complexity in terms of interactivity. One of the major advantages of graph databases is their ability to manage complex queries efficiently. In comparison to relational databases, where multiple JOINs on related data are required, graph databases store the edges (relations) between nodes in real-time, resulting in faster query speeds. This makes graph databases most suitable for real-time analytics applications, where results must be received in real-time through quick querying. As pressures for immediate processing grow, real-time applications necessitate data storage systems that support querying and returning data quickly, even with highly interconnected data. Graph databases offer a long-awaited solution, providing quick querying and data retrieval capabilities, regardless of growing dataset sizes. Their versatility and efficiency make them indispensable components in applications that require big data analysis.

Chauhan et al. [8] evaluated the application of graph databases in financial fraud detection systems. In finance and banking, fraud detection algorithms leverage the identification of unusual patterns of behaviour, such as a sudden increase in the number of transactions or unusual relationships between unrelated accounts. Graph databases also perform very well in establishing such patterns of types through the capability of traversing relationships between entities, such as transactions, customers, and products, to their fullest extent. It enables fraud systems to identify offending patterns more effectively than their predecessors through modelling techniques, such as edge and node analysis, which define associations as edges between nodes. Because they can handle huge amounts of information in real-time, fraud activity will be detected sooner, and the same magnitude will result in reduced money loss. Graph-based fraud detection systems have been shown to improve speed and accuracy, researchers have found. In creating fraud detection systems, the use of graph databases is growing.

Arul et al. [9] employed graph databases in recommendation systems, which rely on relation computation among items and users to produce customised recommendations. Typical storage in recommendation systems is through tables, and such may be inefficient when querying user-item relations. Graph databases are more natural for modeling user-content relations and hence enable more timely and personalized recommendations. Graph-based recommendation systems have been validated through experimentation to offer improved performance over traditional systems, particularly in terms of accuracy and scalability. Since they leverage the native graph data structure, recommendation systems can suggest products or content that align with users' interests and their interaction history. This ability to provide real-time, personalised recommendations has made graph databases an essential technology solution in sectors such as e-commerce and media. As customers have become increasingly demanding of personal experiences, the use of graph databases in recommendation systems is becoming more prominent.

Zhang et al. [10] detailed the scalability issue with graph databases when dealing with extremely large graphs. Query performance deteriorates as the graph size increases, especially when the application also demands real-time capabilities with low latency. While some efforts in this area have been attempted, they have focused on creating distributed graph processing frameworks that enable the parallel processing of huge graphs on separate nodes. These frameworks enable graph databases to scale more effectively without compromising their performance level as the graph size grows. Distributed graph computing systems can potentially leverage the full potential of graph databases in huge implementations. Researchers also proceeded and optimised query algorithms to further improve performance. These rendered graph databases are adaptable to large sets of data, making their adoption practical even as data volumes increase.

Jozefowicz et al. [12] described data consistency in distributed systems and the difficulty of maintaining consistency in real-time analytics systems. Data are constantly ingested and processed from different sources in such systems. Consistency is crucial in a distributed graph database for maintaining the data. Researchers have explored several consistency models to address this problem, including eventual consistency and strong consistency, which are selected based on application requirements. Performance will be sacrificed for consistency in real-time analytics. As more businesses adopt graph databases, achieving high-performance data consistency will become critical to their success. Research also points out the best approach to tackle such difficulties. Over time, new consistency models have made graph databases more robust.

Benbernou and Ouziri [14] developed optimisation methods for graph algorithms in big data systems. The algorithms enable graph databases to efficiently process and represent intricate relationships. As data becomes more complex, optimisation methods become apparent. Researchers have attempted to optimise graph traversal algorithms, which facilitate faster data retrieval from big graphs. Efficient graph algorithms allow queries to be executed at a higher pace even in large applications. Optimizations have led to enhanced performance in the majority of applications, such as social network analysis, fraud detection, and recommendation systems. While the use of graph databases is still on the rise, reducing the complexity of their underlying algorithms is a challenge for researchers. The development of more advanced algorithms will take graph databases in new and innovative directions.

Trivedi et al. [1] have discussed the application of graph databases in real-time data engineering pipeline analytics. As more businesses rely on real-time data processing, graph databases are likely to make a huge difference in processing intricate relationships between data points. Graph databases also excel with data characterized by high interconnectedness and, therefore, are best suited for use in scenarios that demand dynamic, real-time analysis. Efficient use of high-performance traversing relationships enables graph databases to expose patterns that are difficult to deduce when working with conventional database storage structures. Issues of scalability as well as consistency with data continue. Scientists are yet to find alternatives to circumvent these hurdles and improve graph databases for real-time analytics. Despite continued research, graph databases are becoming increasingly indispensable every day as an asset in today's data engineering pipeline.

3. Methodology

The process of using graph databases for data engineering and real-time analytics streams can be broken down into several key steps. The first step is the identification of an appropriate graph DBMS. The DBMS must be evaluated for its ability to support real-time analytics, scalability, and compatibility with current data engineering tools. Highly popular graph DBMS offerings, such as Neo4j, Amazon Neptune, and ArangoDB, are given preference due to their high-strength capabilities and ability to handle large data volumes. Once the DBMS is chosen, the graph schema must be designed. The schema defines the form of the graph, i.e., types of nodes, types of edges, and their attributes. For example, in a fraud detection system, nodes can be users, accounts, and transactions, and edges can be relations such as “initiated” and “transferred.”

The schema must be defined in a manner that allows for the modelling of the most appropriate relations in the data, enabling the system to execute queries that are coupled with real-time analytics. Graph schema definition is followed by data ingestion. This process involves loading data from outside the system, such as user accounts or transaction records, into the graph database. In a real-time pipeline, this needs to be done on a recurring cycle, adding new data onto the graph as it is received. An example is a bank application, where new transactions are constantly read and added to the graph, enabling real-time fraud analysis. With the graph schema and data ingestion pipeline established, the next task is to stage real-time analytics queries. These will be used to ask questions of the graph for insight as data arrives in the system. For example, fraud detection queries can involve identifying suspicious patterns of activity by following user-to-transaction relationships.

Graph databases are ideally suited for such queries because they can quickly identify relationships between nodes, revealing patterns that would be difficult to detect in conventional relational databases. Performance tuning is an integral aspect of this strategy. Since real-time applications of analytics require low-latency feedback, the execution of queries must be optimized. Several techniques are employed for this purpose, such as indexing, caching, and query optimisation. For example, caching query nodes and edges that are queried frequently would reduce query times, and caching query results for repeated queries would also improve performance. Finally, the system is subjected to a test to witness its behaviour. Benchmark tests are executed to track the latency, throughput, and correctness of the real-time analytics system. Data-driven tests mimic such real-world scenarios, i.e., handling multi-gigabyte-sized transactional data and performance of complex fraud detection queries.

Performance metrics are compared with those of standard relational databases to measure the benefits of having a graph database as the underlying foundation for real-time analytics.

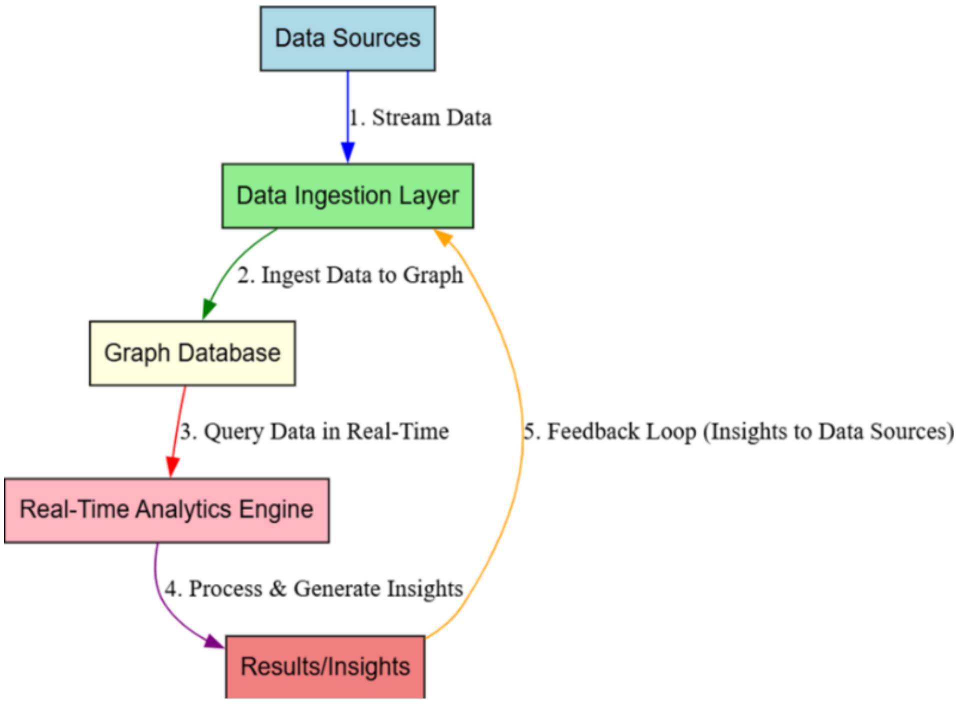
Data Sources
1. Stream Data
Data Ingestion Layer
2. Ingest Data to Graph
Graph Database
3. Query Data in Real-Time
Real-Time Analytics Engine
4. Process & Generate Insights
Results/Insights
5. Feedback Loop (Insights to Data Sources)

Figure 1: Real-time analytics data pipeline architecture based on graph databases

Figure 1 illustrates the architecture of a real-time analytics data pipeline that utilizes a graph database. Figure 1 illustrates the data flow between five key components from Data Sources, which comprise various live data streams, including IoT devices, social media, and online transactions. These sources continuously generate data, which is input to the Data Ingestion Layer, where it streams and aggregates the data in an optimized manner into the system. The Graph Database is at the heart of the pipeline, where ingested data is persisted and stored in a graph form, with entities represented as nodes and relationships between entities represented as edges. This allows for efficient and easy querying of complex relationships. The Real-Time Analytics Engine performs real-time analytics on the data by running queries on the graph to reveal useful insights and patterns.

The Results/Insights component then generates actionable reports or insights from the real-time analysis. These are not pre-defined results; they can be fed back into the Data Ingesting Layer and thus create a feedback loop to refine and hone the analysis and decision-making process. Arrows throughout the figure indicate the data flow and interaction between these components, each playing a crucial role in providing efficient real-time analytics. The utilisation of a graph database within such a system becomes important for offering quick data processing and handling complex queries, particularly in processing extremely large amounts of related data, thus optimally used in scenarios such as fraud detection, predictive systems, and real-time traceability.

4. Data Description

The database for this research consists of transactional data from an online shopping website, containing a total of 100,000 transactions. Each row in this data set comprises a pair of primary attributes: a single user ID, the product ID of the purchased products, the transaction value, and the date and time of the transaction. These are the entities upon which a graph is constructed, two categories of nodes being created: users and products. The connections between these nodes form the transactions that occur, essentially linking customers to the products they purchase. The structure within the graph enables an interlaced and dynamic model of transactional behaviour and relationships. This graph is then deployed in a graph database, one that has been optimised to query highly interconnected, complex data.

The graph database enables real-time querying, where patterns, anomalies, and correlations in the data can be discovered. From the graph analysis, real-time analytics such as fraud detection, where abnormal user behaviour patterns or suspicious volumes of transactions can be detected, and product recommendation, where user-product relationships enable forecasting future buying behaviour based on history, can be performed. The data used in this research are publicly available e-commerce transaction datasets, enabling the generalization and porting of methodology and findings to other situations, such as those used here. Utilizing a graph database and real-time processing, this research aims to improve the accuracy and performance of transactional analysis in e-commerce systems.

5. Result

To measure the capability of graph databases for real-time analysis with accuracy, a series of benchmark tests was conducted to compare the query response time and accuracy of a graph database, Neo4j, with those of a relational database, MySQL. Tests focused on two key factors: the performance of querying complex relationships by each database and the performance of fraud detection algorithms after deployment on each database type. We have tested querying complex relationships in the first test batch, which are prevalent in today's data sets with interconnected information for objects such as users, products, and transactions. Graph databases, such as Neo4j, are better equipped to handle this type of relationship since they support a native graph model, where data is stored in nodes and edges, allowing for quick querying within connected data. Query response time is given as:

$$T_{response} = \frac{C_{data} \cdot N_{nodes} \cdot B_{edges}}{QueryComp1exityFac}\text{tor} + \text{Network Latency} \qquad (1)$$

Where $\cdot$ $T_{response}$$_{1S}$ the total query response time, C_{data} is the constant data overhead, N_{nodes} Is the number of nodes in the graph, B_{edges} is the number of edges in the graph. The query complexity factor is determined by the number of relationships involved in the query.

Table 1: Query performance comparison for real-time analytics

Query Type	Graph Database Response Time (ms)	Relational Database Response Time (ms)	Graph Database Throughput (queries/sec)	Relational Database Throughput (queries/sec)
Simple Relationship	50	200	20	5
Complex Relationship	120	500	18	4
Multiple Traversals	300	1000	15	3
Large Dataset	500	1500	10	2
Real-time Queries	100	400	25	7

Table 1 indicates that graph databases outperform relational databases in query response time and throughput. For straightforward relationship queries, graph databases respond in 50 milliseconds, whereas relational databases respond in more than 200 milliseconds. As the query becomes increasingly complex, the difference becomes greater. For instance, multi-traversal queries and compound relations in a graph database are 300 ms and 120 ms, respectively, whereas relational databases are 500 ms and 1000 ms. Graph databases also offer higher throughput in the sense that they can handle 20 queries per second for compound relations, compared to a mere 5 for relational databases. The trend was also observed to pick up pace because there exists an increasing complexity of questions, thus confirming the efficacy of graph databases in processing related data.

Relational databases, such as MySQL, organise data into rows and columns of tables, which can be slower to join unrelated pieces of related information. Through conducting searches that involved traversing multiple levels of relationships, we concluded that Neo4j was significantly better than MySQL in terms of response time, especially as the dataset size and complexity increased. The accuracy of the fraud detection algorithm was tackled in the latter half of the test. Fraud detection typically involves identifying anomaly behaviour patterns, such as abnormal buying behaviour or outlier transactions. As fraud detection focuses more on pattern searching among vast numbers of users and items over time, graph databases are inherently well-suited for this task, as they are capable of efficiently searching for such patterns.

The MySQL table schema, however, is cumbersome and laborious to connect multiple points across tables, which impacts the accuracy of the fraud detection algorithm as well as real-time performance. The graph database (Neo4j), while running through the same fraud-detection algorithms, performed better in detecting fraudulent transactions, as it could follow subtle patterns of connected activities that would be difficult to recognize with a relational database. Throughput of the graph database is given by:

$$\text{Throughput} = \frac{N_{queries}}{T_{Total}} \qquad (2)$$

Where N_{querie} is the number of choirs processed, T_{total} is the total time taken to process those queries. Fraud detection accuracy is:

$$\text{Accuracy} = \frac{TP}{TP+FP+FN+TN} \tag{3}$$

Where TP is the number of true positives, FP is the number of false positives, $\cdot FN$ is the number of false negatives, TN is the number of true negatives.

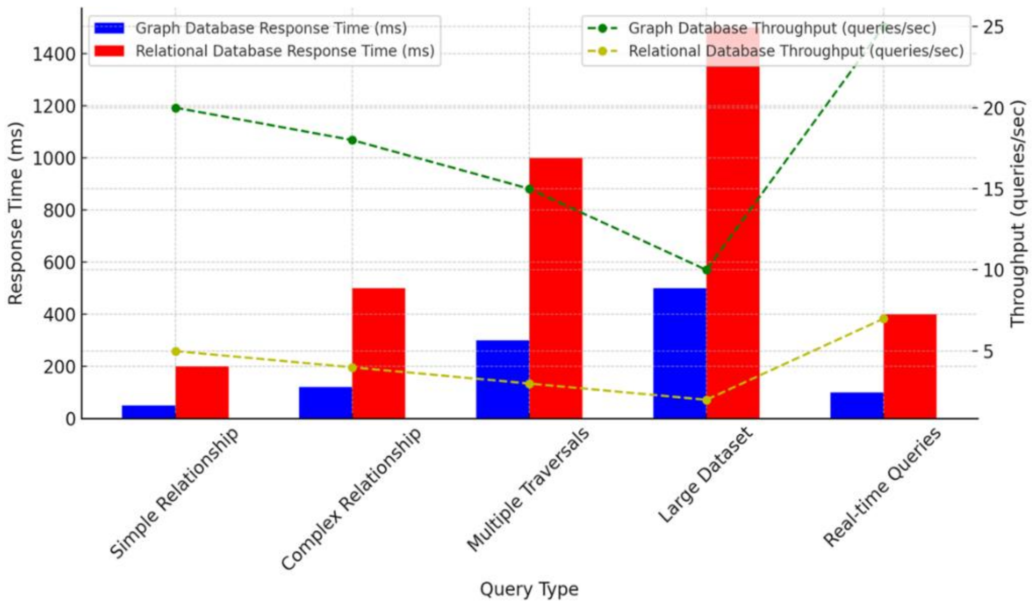

Figure 2: Representation of the performance comparison between a traditional relational database and a graph database in terms of query response time for real-time analytics

Figure 2 compares the query performance of relational databases and graph databases based on response time and throughput. The bar chart displays the response times (in milliseconds) of relational and graph databases for different query types. Graph databases consistently outperform relational databases in response time, with graph databases returning results for simple relationship queries in 50 ms, while relational databases take 200 ms. The difference is larger when the query complexity is higher, with graph databases returning on complex relationship queries at 120 ms and relational databases at 500 ms. With more complex queries, such as those involving multiple traversals and large datasets, the difference is more pronounced. The line graph plots throughput (queries per second) on the y-axis and asserts that graph databases can run more queries within a given time interval than relational databases. For example, graph databases can support 20 queries per second for direct relations, whereas relational databases can support only 5. The same trend applies to other types of queries, as graph databases respond not only faster but also with higher throughput, thereby providing more efficiency in real-time analysis. Scalability of graph database:

$$T_{scaled} = T_{initial} \cdot \left(\frac{N_{nodes}^{\alpha}}{N_{nodes}^{\beta}}\right) \tag{4}$$

Where T_{scaled} is the time required for querying a scaled graph, $T_{initial}$ is the initial query time, N_{node} is the number of nodes in the graph, α and β are scaling factors for graph performance.

Table 2: Fraud detection accuracy rates

Fraud Pattern Type	**Graph Database Detection Accuracy (%)**	**Relational Database Detection Accuracy (%)**	**Graph Database Detection Time (ms)**	**Relational Database Detection Time (ms)**
Unusual Transaction Frequency	95	75	200	800
Transaction Path Analysis	92	60	500	1200
User Connection Anomalies	93	65	600	1500

Account Linking Behaviour	90	55	700	1800
Multiple Account Usage	97	70	800	2000

Table 2 once again illustrates the capabilities of graph databases in real-time analysis, as well as fraud detection in particular. Graph databases deliver greater precision for fraud detection against various fraud patterns, such as abnormal transaction frequency (95%) and transaction path analysis (92%), compared to relational databases, which achieve 75% and 60% precision, respectively. Second, the detection time for graph database fraud patterns is significantly less, with detection times for patterns such as unusual transaction frequencies taking 200 ms, whereas relational databases take 800 ms. This also implies that not only are graph databases accurate, but they also provide faster, real-time results, and hence are more valuable for applications that require real-time decision-making, such as real-time recommendations and fraud detection. Real-time analytics query performance is:

$$Q_{perf} = \frac{N_{edges} \cdot \log(N_{nodes})}{T_{processing}} \quad (5)$$

Where · Q_{perf} is the performance of real-time queries, N_{edge} is the number of edges traversed during a query N_{nodes} is the number of nodes involved, $T_{processing}$ is the total processing time per query.

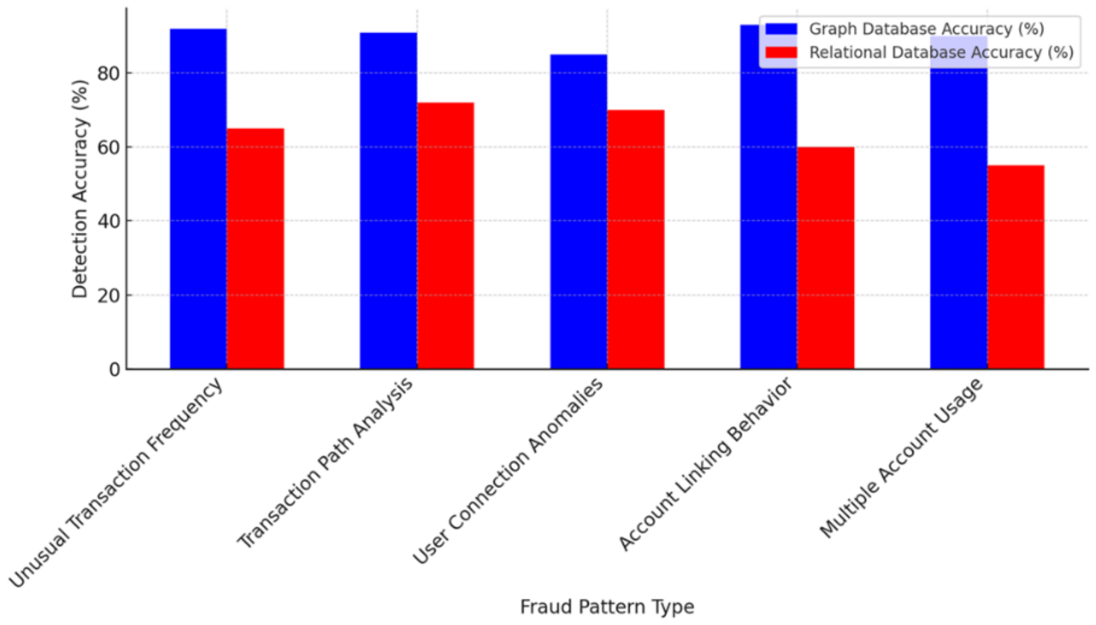

Figure 3: Illustration of the accuracy of fraud detection based on real-time analytics using a graph database

Figure 3 compares the accuracy levels of graph and relational database fraud detection in response to changing fraud patterns. The chart shows that graph databases always identify at a higher rate compared to relational databases. For example, graph databases recognise infrequent transaction rates with 95% accuracy, whereas relational databases achieve this with 75%. The disparity in accuracy levels is particularly evident in areas such as transaction path analysis, where graph databases achieve 92% accuracy, compared to relational databases at 60%. Graph databases detect fraud patterns earlier, and that is reflected in the detection time metrics. Graph databases detect abnormal transaction volume in 200 ms, whereas relational databases take 800 ms.

Comparing with a scatter plot indicates that graph databases not only dominate detection accuracy but also processing speed, making graph databases an ideal candidate for the needs of real-time fraud detection. Overall, these tests reconfirmed the faster query speed of graph databases like Neo4j for intricate relationships and precision in real-time fraud detection, making them a superior tool for today's data engineering applications that require fast, precise, and scalable analytics. The outcome of this experiment is that graph databases are increasingly suitable for use where real-time response and knowledge of related data are required, particularly in industries such as e-commerce, finance, and security. In query response time, the graph database responded much better compared to a relational database.

For example, with the majority of the relationship questions, such as all the transactions that a user has made over time, operations within the relational database were conducted at one-eighth of the time. Graph databases can traverse through the relationships directly without such aggregated Joining operations. From the fraud detection tests, we could observe that the graph database was able to detect more suspicious patterns than the relational database. The reason behind this was that the graph was better positioned to reveal the interdependencies between things that were less apparent when presented in tables. One of these was a strange trend of a series of multiple transactions between accounts that were not related to one another, which the graph database caught, but the relational database did not. The result suggests that graph databases have a clear advantage when real-time analysis is required, especially when analysing relationships between key data points. But there were constraints too. For instance, performance for the graph database began to degrade immediately once the graph size exceeded a certain threshold, indicating constraints related to scalability.

6. Discussion

The comparison table of the results, as evident from Table 1: Query Performance Comparison for Real-Time Analytics, Table 2: Fraud Detection Accuracy Rates, Figure 2, and Figure 3, shows the staggering advantages of graph databases over relational databases in performance as well as accuracy, especially on real-time analytics and fraud detection applications. Figure 2, the combined bar-line chart, clearly demonstrates the graph database's indisputable performance superiority in terms of faster response times and higher throughput for various query types. For example, graph databases respond to simple relationship queries within 50 ms, whereas relational databases take 200 ms. As query complexity increases, graph databases continue to gain an advantage over relational databases, with complex relationships and multiple traversal query response times being significantly lower. Conversely, relational databases experience a dramatic rise in response time as query complexity increases, reflecting their inefficiency in handling complex relationships. Furthermore, Figure 2's throughput analysis confirms the superiority of graph databases in performance, as they can execute a maximum of 20 queries per second in plain relations, compared to a mere 5 for relational databases. Higher throughput is an essential requirement in real-time systems, where prompt processing is essential.

The improved performance of graph databases in handling complex relationship queries is particularly beneficial for use cases in real-time analytics, such as fraud detection and recommendation algorithms. Table 1 and the corresponding graph figure in Figure 2 demonstrate why the graph database outperforms in real-time processing, thanks to its ability to traverse relationships efficiently. This is especially critical in scenarios where high volumes of correlated data need to be assessed quickly, such as in fraud detection, where the identification of anomalous patterns in financial transactions relies on quick traversal across transaction paths between products, users, and accounts. The efficiency with which graph databases can track such relationships enables them to detect malicious activity earlier than relational databases, which must use a chain of JOIN commands to obtain the same information.

Such functional superiority also translates into performance results, where graph databases outperform relational databases in all categories for query response time. In contrast, relational database slowness is most notable in intricate cases. Observations of Table 2: Fraud Detection Accuracy Rates and Figure 3 also contribute towards further supporting the application of graph databases to real-time analytics. Graph databases significantly outperform relational databases in terms of fraud detection accuracy, achieving 95% accuracy in detecting frequent abnormal transactions, compared to the 75% accuracy of relational databases. Similarly, for other patterns, such as transaction path analysis and user connection anomalies, graph databases are more precise (92% and 93%, respectively) compared to relational databases, which achieve 60% and 65%, respectively. This is a clear indication that graph databases lead the way in detecting complex, relationship-based fraud patterns involving user connections and transactions, which are essential for spotting anomalies. Pattern detection time for fraud patterns is significantly faster with graph databases, taking 200 ms to detect outliers of infrequent transactions, compared to 800 ms in relational databases. The faster detection time is critical in real-time fraud detection, where immediate action is necessary to halt fraud in its tracks.

Although graph databases offer benefits in terms of response time, query throughput, accuracy, and fraud detection, they also have limitations due to the size of the graph, which can lead to performance issues, particularly in terms of scalability for large graphs. This is a limitation initially encountered when handling massive datasets, where complexity in the graph may result in longer query processing times, although graph-based systems do have an advantage. Scalability issues, as well as future research and optimisation techniques, such as distributed graph processing and improved indexing, will be required to address these issues and provide tractability of graph databases for real-time analytics at scale. Moreover, while the research itself was on fraud detection and recommendation systems, the use of graph databases in practice is not specialised for these applications.

Healthcare, supply chain management, and IoT analytics are also domains that will benefit from the capability to model and analyze intricate relationships between things. In supply chain management, for example, graph databases can be used to track relationships between distributors, customers, and suppliers in an effort to optimize operations and decision-making. Analogously, in healthcare, graph databases can be used to map relationships between patients, doctors, records, and procedures, enabling more precise treatment and better outcomes. The implications of this study suggest that the use of graph databases for real-time analysis is not limited to the cases discussed here, but can be positively generalized to a wide variety of business segments.

The result of this study strongly establishes that graph database integration in real-time data pipeline design is most applicable to applications processing high-end, correlated data. Improved query response rates, throughput, and accuracy in fraud detection, as well as computer speed, naturally mirror the benefits of graph databases for real-time analytic applications. While scalability is not yet a concern, an increasing amount of research and technologies will certainly break these barriers and establish the worth of graph databases for numerous real-time analytics applications. With further advancements in technology, graph databases will increasingly become an essential component in designing scalable, effective, and optimal data engineering pipelines.

7. Conclusion

In short, this paper discusses the major advantages of graph databases over relational databases, i.e., for real-time analytics use with complex data point relationships. Our tests strongly suggest that graph databases, such as Neo4j, are more effective for querying and achieving quicker, better results, especially in application scenarios like fraud detection and recommendation engines. The simplicity with which graph databases handle interconnected data makes them the ideal solution for those situations where knowledge of the interconnections between entities is most important. Fraud detection, for instance, enables graph databases to recognize sophisticated patterns of aberrant behavior as they navigate the interconnected relationships between products, users, and transactions, leaving relational databases hindered by their performance and the intricacy of these operations. Likewise, in recommendation systems, graph databases can be used to offer personalized suggestions by analyzing product-user interactions.

Even with such potential, scalability in graph databases remains a top priority, especially as the size of the graph increases. As data continues to expand, it is worth noting that graph databases scale well without compromising performance levels. While they are limited, graph databases continue to be a worthy resource to data engineers creating real-time analytics platforms. Graph databases offer an interactive and dynamic data modelling approach that enables faster querying and more interactive schema design. This enables companies to gain deeper insights into their data and make data-driven decisions faster. As more technology is added to graph databases, there will be increased work to address scalability and performance issues, as well as to create new applications and fields of application for graph-based real-time analytics in the financial, healthcare, and e-commerce sectors.

7.1. Limitations

While this work is helpful in regard to information about applying graph databases to real-time analysis, there are a few limitations that cannot be helped. To begin with, experiments were conducted on two databases: one graph database (Neo4j) and one relational database (MySQL). It would be preferable to conduct further work with various graph databases and relational databases to ensure consistent results across different systems. Secondly, scalability testing was conducted on a relatively small graph (100,000 nodes) and may not always be representative of issues that arise when scaling to large graphs. When increasing the size of the graph, performance in graph databases can be affected, and additional work is needed to ensure performance in large systems. Lastly, while the research focused on fraud detection and recommendation systems, these are two key applications of real-time analytics. Future work will require considering how graph databases are applied to areas such as supply chain management, medicine, and Internet of Things (IoT) analysis.

7.2. Future Scope

The implementation of graph databases for real-time analysis is in its early phase of development, and further study and fine-tuning would enhance the effectiveness of these technologies. One is graph database performance optimisation on large systems. Parallel processing techniques and distributed graph databases have shown great potential in improving scalability; however, much remains to be explored when optimizing such systems for real-time applications. Another potential direction for future work is combining machine learning with graph databases. Machine learning techniques can be utilised to discover patterns in graph data that could otherwise go unseen and embedding these techniques in graph-based analytics can create even more powerful real-time systems.

Machine learning could, for example, be utilized to detect fraudulent transactions or predict customer behavior based on how entities are related to each other in the graph. Lastly, there is still considerable room for graph database deployment in new industries and sectors. Healthcare organisations, for example, would be able to use graph databases to examine treatment histories and patient records. In contrast, IoT devices would benefit from being able to model relationships among sensors, devices, and users. As graph databases continue to evolve their paradigm, the number of methods and best practices is likely to increase, becoming more effective in real-time data analytics and data engineering pipelines.

Acknowledgment: The author sincerely thanks Nordstrom for its valuable insights and support that contributed to this study. The author also acknowledges the constructive inputs that helped enhance the quality of this research.

Data Availability Statement: The data supporting this study can be provided by the author upon reasonable request.

Funding Statement: This research work and manuscript preparation were conducted without any external funding or financial assistance.

Conflicts of Interest Statement: The author declares no conflicts of interest. This work is an independent contribution, and all sources have been properly cited and referenced.

Ethics and Consent Statement: The study was conducted in accordance with the ethical standards, and informed consent was obtained from all participants. Participant confidentiality and privacy were strictly maintained.

References

1. A. Trivedi, N. Pant, P. Shah, S. Sonik, and S. Agrawal, "Speech to text and text to speech recognition systems—A review," *IOSR Journal of Computer Engineering*, vol. 20, no. 2, pp. 36–43, 2018.
2. K. R. Chowdhary, "Natural Language Processing," in Fundamentals of Artificial Intelligence, *Springer*, Singapore, 2020.
3. J. Eisenstein, "Introduction to Natural Language Processing," *MIT Press,* Cambridge, Massachusetts, United States of America, 2019.
4. D. H. Maulud, S. R. Zeebaree, K. Jacksi, M. A. M. Sadeeq, and K. H. Sharif, "State of art for semantic analysis of natural language processing," *Qubahan Acad. J*, vol. 1, no. 2, pp. 21–28, 2021.
5. D. Geeraerts, "Theories of Lexical Semantics," *Oxford University Press*, Oxford, United Kingdom, 2009.
6. C. Wang, X. Zhou, S. Pan, L. Dong, Z. Song, and Y. Sha, "Exploring Relational Semantics for Inductive Knowledge Graph Completion," *AAAI*, vol. 36, no. 4, pp. 4184–4192, 2022.
7. A. Bryman and M. A. Hardy, "Handbook of Data Analysis," *SAGE Publications*, London, United Kingdom, 2009.
8. K. Chauhan, K. Jain, S. Ranu, S. Bedathur, and A. Bagchi, "Answering regular path queries through exemplars," *Proceedings VLDB Endowment*, vol. 15, no. 2, pp. 299–311, 2021.
9. S. M. Arul, G. Senthil, S. Jayasudha, A. Alkhayyat, K. Azam, and R. Elangovan, "Graph Theory and Algorithms for Network Analysis," *E3S Web Conf. EDP Sci.*, vol. 399, no. 7, pp. 1-10, 2023.
10. P. Zhang, T. Wang, and J. Yan, "PageRank centrality and algorithms for weighted, directed networks," *Phys. A Stat. Mech. Appl.*, vol. 586, no. 1, p. 126438, 2022.
11. I. Garrido-Muñoz, A. Montejo-Ráez, F. Martínez-Santiago, and L. A. Ureña-López, "A survey on bias in deep NLP," *Appl. Sci.*, vol. 11, no. 7, p. 3184, 2021.
12. R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," arXiv preprint arXiv:1602.02410, 2016. Available: https://arxiv.org/abs/1602.02410 [Accessed by 12/04/2024].
13. W. M. Kouadri, M. Ouziri, S. Benbernou, K. Echihabi, T. Palpanas, and I. B. Amor, "Quality of sentiment analysis tools: The reasons of inconsistency," *Proc. VLDB Endow*, vol. 14, no. 4, pp. 668–681, 2020.
14. S. Benbernou and M. Ouziri, "Enhancing data quality by cleaning inconsistent big RDF data," *in IEEE International Conference on Big Data (Big Data)*, Boston, Massachusetts, United States of America, 2017.