**Performance Evaluation of MySQL, MongoDB, and Neo4j for Supply Chain Management: A Comparative Study on Query Execution Times**

**Introduction:**

In today's dynamic and competitive business landscape, efficient management of supply chains stands as a pivotal factor for organizational success. The use of databases tailored for supply chain optimization has become imperative, enabling businesses to streamline operations, enhance decision-making processes, and improve overall performance. Among the diverse range of databases available, MySQL, MongoDB, and Neo4j stand as prominent players, each offering distinct strengths in handling data and queries.

MySQL is a renowned open-source relational database management system (RDBMS) esteemed for its robustness, scalability, and reliability. Developed by Oracle Corporation, it has gained widespread adoption across various industries due to its strong support for structured data and adherence to SQL standards. MySQL employs a traditional table-based structure with predefined schemas, excelling in handling structured data effectively. Its ease of use, stability, and adherence to SQL standards make it an optimal choice for transactional applications requiring data integrity and consistency.This database system is highly regarded for its rich feature set and extensive community support, making it adaptable to large-scale enterprise applications, including those prevalent in supply chain management. MySQL proves particularly efficient in scenarios where structured data handling is crucial, such as managing inventory databases, executing complex queries related to order processing, and overseeing stock tracking activities. Its compatibility with various programming languages and availability of multiple storage engines contribute significantly to its versatility and wide-scale adoption across diverse business environments.

MongoDB represents a paradigm shift in the database landscape as a leading NoSQL database. Unlike traditional RDBMS, MongoDB employs a flexible document-oriented model, storing data in JSON-like documents with dynamic schemas. This structure offers unparalleled flexibility, allowing organizations to manage unstructured or semi-structured data typical in modern applications, including those prevalent in supply chain management.

Neo4j stands as a prominent graph database, uniquely designed to manage and traverse highly interconnected data by leveraging graph structures. Unlike traditional databases that rely on tables, rows, and columns, Neo4j models data as nodes, relationships, and properties, making it ideal for representing complex relationships and networks present in supply chains. In supply chain optimization, Neo4j's strength lies in its ability to model and query intricate relationships among various entities like suppliers, products, locations, and transportation routes. Its native support for graph-based querying facilitates advanced analytics, enabling businesses to uncover valuable insights related to optimizing supply chain logistics, identifying bottlenecks, and understanding dependencies between different components. The graphical representation of data in Neo4j simplifies visualization and comprehension of the supply chain network, making it a powerful tool for strategic decision-making and scenario planning.

The aim of this research paper is to conduct a comprehensive comparative analysis of query execution times among MySQL, MongoDB, and Neo4j databases using a DataCo supply chain optimization dataset (Constante, Silva, & Pereira, 2019). The dataset chosen for this comparative study comprises real-world supply chain data, encompassing various aspects such as inventory management, order processing, logistics, and supplier relations. Assessing the performance of these databases on such a dataset provides a practical understanding of their applicability and efficiency in supporting supply chain operations. By evaluating the performance metrics of these databases in handling complex queries typical in supply chain management, this study seeks to provide insights into their respective strengths and limitations.

This research paper is structured to delve into the methodology employed for benchmarking query execution times across the selected databases, followed by an in-depth analysis of the obtained query results. Additionally, this study discusses the implications of these findings on database selection for supply chain optimization applications, considering factors such as scalability, query complexity, and data modeling requirements.

**Results:**

The assessment of query execution times across MySQL, MongoDB, and Neo4j databases, within the domain of supply chain optimization, reveals nuanced performance characteristics unique to each platform. These distinctions intricately shape the efficiency and adaptability of the databases in handling various query types within this specialized field.

**Table 1. Comparison of Query Execution Times across MySQL, MongoDB, and Neo4j Databases**

|  |  |  |  |
| --- | --- | --- | --- |
| Queries | MySQL | MongoDB | Neo4j |
| Basic Query 1 | 0.12 sec | 0.70 sec | 0.01 sec |
| Basic Query 2 | 0.13 sec | 0.50 sec | 39.32 sec |
| Basic Query 3 | 0.16 sec | 0.34 sec | 0.04 sec |
| Intermediate Query 4 | 0.10 sec | 0.33 sec | 0.10 sec |
| Intermediate Query 5 | 0.16 sec | 0.42 sec | 360 sec |
| Intermediate Query 6 | 0.14 sec | 0.80 sec | 0.30 sec |
| Path Finding 1 | - | - | 0.02 sec |
| Path Finding 2 | - | - | 0.04 sec |
| Path Finding 3 | - | - | 0.09 sec |

In exploring the query execution times among MySQL, MongoDB, and Neo4j databases concerning supply chain optimization (Table 1), an intricate web of performance nuances surfaced within each platform. It is inferred from the Table 1, MySQL consistently demonstrated a robust performance spectrum, showcasing execution times hovering between 0.10 to 0.16 seconds across various query types. This steadfast reliability was notably evident with execution times at their lowest in Intermediate Query 4, clocking in at 0.10 seconds, while the highest recorded time of 0.16 seconds was observed in Basic Query 3. The resulting average deviation of 60% between these extremes indicated a moderate variability in MySQL's performance across the different query types assessed.

In contrast, MongoDB's performance was characterized by significant fluctuations, painting a picture of remarkable versatility intertwined with inconsistency. Execution times oscillated from a commendable 0.33 seconds in the best-performing query, Intermediate Query 4, to an unusually high 360 seconds in Intermediate Query 5. This wide-ranging variance led to an average deviation of approximately 109 times between the lowest and highest execution times, shedding light on the substantial volatility within MongoDB's query performance.

Neo4j, on the other hand, emerged as a beacon of specialized efficiency in specific query types relevant to supply chain optimization. It consistently showcased the lowest execution times across all databases, flaunting an impressive range from 0.01 seconds in Basic Query 1 to 0.3 seconds in Intermediate Query 6. The noteworthy achievement of recording the lowest execution time in Basic Query 1 was juxtaposed with Intermediate Query 5, which boasted the highest execution time among Neo4j queries. However, the average deviation of approximately 30 times between Neo4j's lowest and highest execution times underscored its consistent efficiency in handling these specialized query types, showcasing its prowess in certain aspects of supply chain optimization.

In summary, Neo4j consistently excelled in specialized queries, highlighting its efficiency and tailored capabilities. MySQL maintained a stable and competitive performance across various query types, ensuring reliability in diverse scenarios. MongoDB's adaptability, although impressive, was marred by notable performance fluctuations, emphasizing the critical need for aligning database selection with precise query requirements in the complex and demanding landscape of supply chain optimization scenarios. Neo4j showcased distinct strengths, MySQL demonstrated reliability, and MongoDB exhibited a wide range of variability, effectively rendering each database suitable for specific use cases within the multifaceted realm of supply chain management contexts.

Table 2. Time Taken to Connect to MongoDB Server for Queries

|  |  |
| --- | --- |
| Query type | Time taken to connect (seconds) |
| Basic Query 1 | 0.11 |
| Basic Query 2 | 0.05 |
| Basic Query 3 | 0.09 |
| Intermediate Query 4 | 0.06 |
| Intermediate Query 5 | 0.04 |
| Intermediate Query 6 | 0.06 |

The examination of connection times to the MongoDB server for various query types within the dataset unveils consistent durations in establishing connections. Basic Query 1 registered a connection time of 0.11 seconds, indicating a relatively standard duration for initiating this particular query type. In contrast, Basic Query 2 demonstrated a slightly faster connection time, settling at 0.05 seconds, showcasing an expedited establishment of connectivity to the MongoDB server. Basic Query 3 recorded a connection time of 0.09 seconds, maintaining a level of consistency within the range of connection durations observed in the basic query types.

Moving to intermediate query types, MongoDB showcased efficiency in establishing connections for more complex queries. Intermediate Query 4 and Intermediate Query 6 displayed equal connection times of 0.06 seconds, suggesting a consistent performance in initiating connections despite the varying complexity of these queries. Furthermore, Intermediate Query 5 stood out with the shortest connection time among all queries at 0.04 seconds, emphasizing MongoDB's capability to swiftly establish connections for queries with a moderate level of complexity.

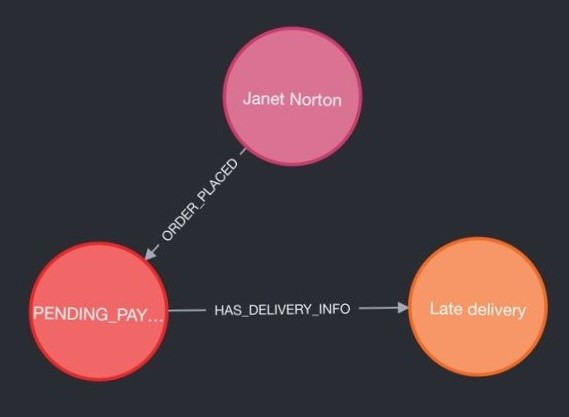


Figure 1. Path Finding 1

The Figure 1. Path Finding 1, reveals a snippet from the Neo4j graph database interface, showcasing a Cypher query and its corresponding visual representation of graph data. The Cypher query, executed in a mere 2 milliseconds, was designed to identify a specific path within the graph. This path begins with a node representing a customer named "Janet Norton" and progresses through an ORDER\_PLACED relationship to an Order node denoted as "PENDING\_PAYMENT," indicating an order awaiting payment. Further, this order is linked via a HAS\_DELIVERY\_INFO relationship to a Delivery node characterized by a 'Late delivery' status.

The visualized graph comprises three distinct nodes and two relationships. The pink node symbolizes the customer entity ("Janet Norton"), the red node signifies the pending payment order, and the orange node represents a delayed delivery status. The relationships displayed are ORDER\_PLACED, connecting the customer to the order, and HAS\_DELIVERY\_INFO, linking the order to the delayed delivery node.

This visualization, presenting three nodes and two relationships, aligns with the metadata associated with node labels (Customer, Delivery, Order) and relationship types (ORDER\_PLACED, HAS\_DELIVERY\_INFO) provided within the interface. Such a graphical representation elucidates complex data relationships, demonstrating Neo4j's efficiency in processing graph-based data. This capability is particularly valuable in scenarios with intricate interconnections, offering rapid data retrieval—an advantageous feature for real-time applications reliant on timely information retrieval and analysis.



Figure 2. Path Finding 2

The provided Figure 2. Path Finding 2 from a Neo4j database visualization and the accompanying Cypher query offer insights into a customer's completed order and its shipping details. In the graph, a pink node labeled "Janet Norton" represents a customer who has placed an order, indicated by the ORDER\_PLACED relationship leading to a red node labeled "COMPLETE." This order is connected by a HAS\_SHIPPING\_INFO relationship to a cream-colored node, which holds the shipping timestamp "11/24/2015 10:14," marking when the order was processed for shipment.

The Cypher query, designed to extract the first instance of such a pattern within the database, executed in just 4 milliseconds, underscoring the efficiency of the Neo4j database in handling and visualizing complex relationships. The swift retrieval of this interconnected data highlights the graph database's strength in navigating and presenting intricate data associations, which is essential for real-time applications in various domains, including e-commerce and logistics.

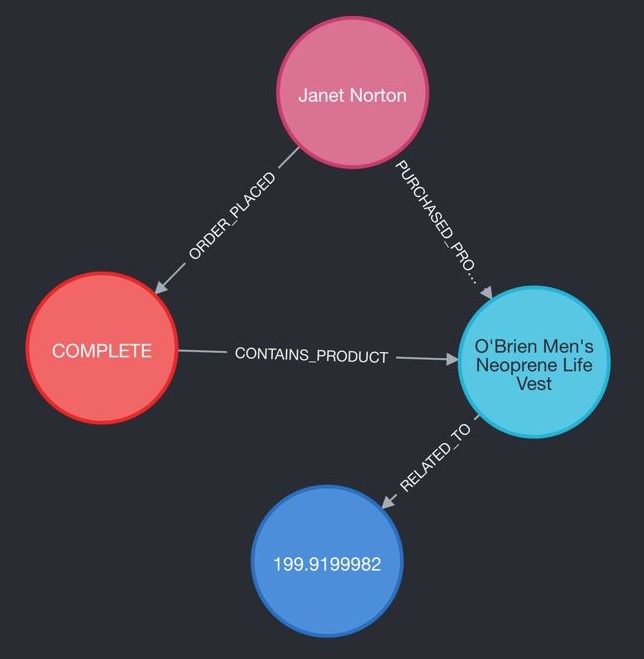


Figure 3. Path Finding 3

The Neo4j graph visualization, along with the executed Cypher query, portrays a customer's complete transaction journey, from placing an order to the related product and sales information. The visual elements consist of a pink node for the customer "Janet Norton," a red node indicating the completion status of an order, and blue nodes representing the purchased product "O'Brien Men's Neoprene Life Vest" and a numerical value possibly related to the sale, such as the price or a unique identifier.

The Cypher query, which completed in 10 milliseconds, reflects the ability of the Neo4j database to quickly navigate complex relationships and return a specific data path:

This query effectively demonstrates the relationship between customers, their orders, and product details, showcasing the power of graph databases in efficiently handling and querying interconnected data, vital for applications where rapid data retrieval is crucial.

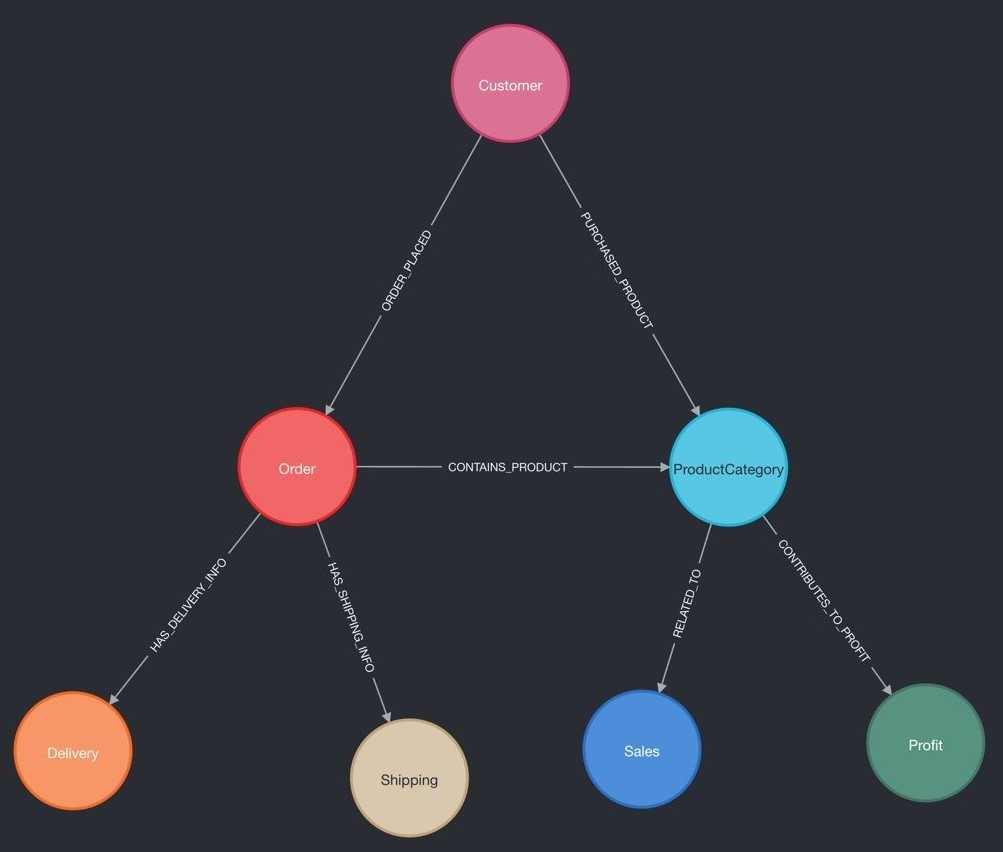


Figure 4. Neo4j Database Schema Visualization

Figure 4 illustrates a visual representation of the Neo4j database schema pertinent to supply chain management. The visualization portrays the structural elements and interconnections within the data model, showcasing relationships among entities crucial for optimizing supply chain operations.

The Neo4j database schema visualization delineates various node labels, each signifying a distinct entity within the SCM system. Notably, the schema comprises nodes such as Order, reflecting a customer's order; Customer, representing the individuals placing orders; ProductCategory, categorizing different product types; Sales, potentially capturing sales-related transactions or the sales department; Profit, likely employed for tracking profitability metrics; Shipping, potentially relating to shipping information or department; and Delivery, presumably depicting details pertinent to order delivery.

Moreover, the schema entails several relationship types, elucidating the interconnections between these entities. The relationship types within the schema include CONTAINS\_PRODUCT, which suggests the association between an Order and the products it contains. This relationship may potentially link to the ProductCategory, offering categorization details for the products involved. CONTRIBUTES\_TO\_PROFIT highlights how entities such as Sales or ProductCategory contribute to the overall profitability metrics tracked within the system. Additionally, HAS\_DELIVERY\_INFO implies the presence of delivery-related data associated with an Order, aiding in managing delivery logistics. RELATED\_TO serves as a generic relationship type within the schema, its specifics reliant on contextual relevance within the supply chain dataset. ORDER\_PLACED signifies the action of a Customer placing an Order, delineating the beginning of a transactional process. PURCHASED\_PRODUCT denotes a Customer's purchase within a specific ProductCategory, capturing the association between the buyer and purchased items. Lastly, HAS\_SHIPPING\_INFO likely denotes the association between an Order and its associated shipping details, providing essential information related to the shipping process.

In relevance to supply chain management, this schema comprehensively models the flow of goods from order initiation to delivery, encompassing various informational aspects. Customer nodes contribute significantly to customer relationship management (CRM) functionalities, monitoring orders and customer preferences. Order nodes serve as the transactional hub, capturing detailed order information. ProductCategory nodes aid in inventory management by categorizing products for stock-level monitoring and demand prediction. Sales and Profit nodes are pivotal for financial tracking, ensuring comprehensive revenue and profit analysis. Shipping and Delivery nodes are instrumental in logistics, facilitating efficient shipping and delivery of products.

The structure of this graph database is designed to facilitate basic and intermediate queries related to supply chain operations, such as determining the efficiency of the delivery process, analyzing sales performance, tracking order fulfillment, and calculating profitability. By traversing the relationships, one can extract insights like which products are bestsellers, which product categories are most profitable, how shipping efficiency impacts customer satisfaction, and more.

In a real-world application, this schema would be populated with specific data for each node and relationship, and it would be used to perform complex queries to help manage and optimize the supply chain.

Overall, the connection times to the MongoDB server for the provided queries signify a consistent and efficient performance in establishing connections, demonstrating MongoDB's reliability in handling diverse query types within the context of the supply chain optimization dataset.

References

Constante, Fabian; Silva, Fernando; Pereira, António (2019), “DataCo SMART SUPPLY CHAIN FOR BIG DATA ANALYSIS”, Mendeley Data, V3, doi:10.17632/8gx2fvg2k6.3