Master's in Big Data

Master's Thesis

Improving Supply Chain Explainability Using Graph Retrieval-Augmented Generation

Author
Gonzalo Bobillo Rincón

Supervised by
Caleb Vicente Moreno

Madrid
June 19, 2025

Gonzalo Bobillo Rincón, declara bajo su responsabilidad, que el Proyecto con título **Improving Supply Chain Explainability Using Graph Retrieval-Augmented Generation** presentado en la ETS de Ingeniería (ICAI) de la Universidad Pontificia Comillas en el curso académico 2024/25 es de su autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Gonzalo Bobillo Rincón Fecha: 19 / 06 / 2025

Autoriza la entrega:

Caleb Vicente Moreno

Nombre del Director

Fdo.: Caleb Vicente Moreno Fecha: 19 / 06 / 2025

V. B. del Coordinador de Proyectos

Nombre del Coordinador

Fdo.: ..... Fecha: ..... / ..... / .....

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. Gonzalo Bobillo Rincón **DECLARA** ser el titular de los derechos de propiedad intelectual de la obra: Improving Supply Chain Explainability Using Graph Retrieval-Augmented Generation, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, los derechos de digitalización, de archivo, de reproducción, de distribución y de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

(a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.

(b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.

(c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.

(d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.

(e) Asignar por defecto a estos trabajos una licencia Creative Commons.

(f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente*).

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

(a) Que la Universidad identifique claramente su nombre como autor de la misma

(b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.

(c) Solicitar la retirada de la obra del repositorio por causa justificada.

(d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

(a) El autor se compromete a:

(b) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.

(c) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.

(d) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

(e) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.

- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.

- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.

- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 19 de junio de 2025

ACEPTA

Fdo.: Gonzalo Bobillo Rincón

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

Abstract

In the context of increasingly complex and interconnected global supply chains, timely access to accurate and explainable insights is essential for operational efficiency and risk management. Traditional analytical tools struggle to address the relational intricacies of supply chain data, often leaving domain experts dependent on technical specialists to extract actionable knowledge. This thesis presents the development and evaluation of a conversational system based on Graph Retrieval-Augmented Generation (Graph-RAG), designed to bridge this gap by integrating a large language model (LLM) with a graph-structured supply chain knowledge base.

The proposed prototype receives user queries in natural language, translates them into Cypher graph queries, retrieves relevant data from a Neo4j database, and generates contextually grounded answers in plain English. The solution was developed and tested using an anonymized dataset representative of real-world supply chain networks and evaluated on a suite of practical use cases. Quantitative and qualitative analyses demonstrate that state-of-the-art LLMs can generate accurate graph queries and coherent responses for complex supply chain scenarios. The system significantly lowers the barrier for non-technical users to interact with large, heterogeneous supply chain datasets, enabling more efficient and transparent decision-making.

Resumen

En el contexto de cadenas de suministro globales cada vez más complejas e interconectadas, el rápido acceso a información precisa y explicable es esencial para la eficiencia operativa y la gestión de riesgos. Las herramientas analíticas tradicionales tienen dificultades para abordar las relacionales complejas de los datos de la cadena de suministro, lo que a menudo obliga a los expertos en la materia a depender de especialistas técnicos para extraer conocimiento. Esta tesis presenta el desarrollo y la evaluación de un sistema conversacional basado en Graph Retrieval-Augmented Generation (Graph-RAG), diseñado para reducir esta brecha mediante la integración de un gran modelo de lenguaje (LLM) y una base de conocimiento sobre la cadena de suministro modelada mediante grafos.

El prototipo propuesto recibe consultas de los usuarios en lenguaje natural, las traduce en consultas Cypher sobre el grafo, recupera los datos relevantes de una base de datos Neo4j y genera respuestas fundamentadas en el contexto en inglés sencillo. La solución se desarrolló y probó utilizando un conjunto de datos anonimizado representativo de redes reales de cadenas de suministro y fue evaluada en un conjunto de casos de uso prácticos. Los análisis cuantitativos y cualitativos demuestran que los LLM de última generación pueden generar consultas sobre grafos precisas y respuestas coherentes incluso en escenarios complejos de cadena de suministro. El sistema reduce significativamente la barrera de entrada para que los usuarios no técnicos interactúen con grandes volúmenes de datos heterogéneos de la cadena de suministro, permitiendo así una toma de decisiones más eficiente y transparente.

Acknowledgements

I would like to thank my supervisor Caleb for his guidance and being there in all the countless daily meetings.

Special thanks to the Supply Chain Planning team at Accenture for providing this interesting project and an amazing environment to make this possible.

I also want to thank the classmates I have met this year at ICAI. I hope that the friendships we have built will last forever.

Finally, i wish to thank my parents, who are always there supporting me, looking forward to me coming home a few days and giving me lots of tuppers to make my life easier in Madrid.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the scope and structure of the thesis. First, we provide context on the critical role and inherent challenges of global supply chains and present the technologies that motivate our work. Next, we articulate the motivations that drive this project. Finally, we outline the primary objective and specific sub-objectives that will guide the design, implementation and evaluation of a conversational Graph RAG prototype for supply chain planning.

1.1 Context

In today's highly interconnected economy, global supply chains play a critical role in almost every industry. According to some estimates, 80% of world trade flows through multinational supply chain networks, and one in five jobs worldwide is tied to these supply chains [1].

However, recent disruptions, from geopolitical tensions and pandemics to natural disasters, have exposed the vulnerability of these complex networks. Companies and governments are increasingly concerned with supply chain transparency and resilience, especially as new regulations (e.g., in the US and EU) demand deeper traceability of suppliers to address sustainability and human rights issues. The core problem is that supply chain data is vast, fragmented, and often incomplete beyond a company's immediate suppliers. A manufacturer may be familiar with its direct suppliers, known as tier-1 suppliers, but often has limited or no visibility into the suppliers that provide goods to those suppliers, referred to as tier-2 or tier-3. This lack of insight can create significant blind spots in managing supply chain risks.

Traditional tools struggle to capture the multi-tier, multi-entity relationships (spanning materials, facilities, shipments, and finance) that characterize an international supply network. As a result, analysts find it difficult not only to gather all relevant information, but also to explain complex chain dynamics in an intuitive way to decision-makers. Even when advanced analytics are applied, their results can appear as black boxes without clear explanations, eroding trust in AI-driven insights.

In this context, there is a pressing need for new approaches that make sense of supply chain complexity and present insights in an explainable, user-friendly manner. One promising direction is the use of knowledge graphs. They can integrate data from disparate sources such as supplier databases, logistics records or market data, representing the supply chain as a network of nodes and relationships. Thanks to this representation, we can easily answer questions like which supplier provides which component, or how a delay at one port affects downstream factories. By connecting different data sources into a unified graph, researchers have shown it is possible to achieve visibility up to multiple tiers of the supply network [2]. Moreover, such graphs enable the discovery of hidden patterns and bottlenecks that would be hard to see in siloed tables or spreadsheets. However, extracting insights from a graph still traditionally requires technical skills like writing graph queries in Cypher or SQL and expert interpretation.

The rise of powerful large language models (LLM) has opened the door for systems that allow users to pose questions in everyday language and receive coherent, context-aware answers. Instead of manually querying databases or combing

through dashboards, a supply chain manager could ask "Which suppliers would be most affected if Factory X in region Y shuts down?", and receive an explanation drawing on the data in the graph. Such natural language querying of a graph could dramatically lower the barrier for non-technical users to gain insights from big supply chain data.

1.1.1 Problem statement

Despite advances in analytics, two core challenges persist in global supply chain management:

1. **Data complexity and fragmentation.** Global supply chains generate "big data" (high-volume, high-variety information on shipments, inventories, supplier performance, etc.) but this data is often underutilized because it is scattered across systems and difficult to query holistically. Conventional data analysis methods, like static reports or BI tools, are insufficient for capturing the dynamic, multi-relational nature of supply chain problems, yet remains siloed and difficult to query as a whole.

2. **High barrier to timely insights.** Extracting and interpreting supply chain relationships currently requires proficiency in graph or SQL query languages and extensive manual exploration. This constitutes an obstacle for domain experts and decision-makers who may possess deep knowledge of supply chain operations but lack proficiency in graph query languages or data science techniques, thereby limiting their ability to directly access and interpret complex relational data embedded within the graph. Moreover, even when analysis is performed (e.g. a simulation identifies a vulnerable supplier), explaining why that supplier is critical (perhaps due to being a single source for a key material, connected to many products) is non-trivial.

The development of a Graph RAG (Retrieval-Augmented Generation) system seeks to address the twin challenges of data complexity and explainability in global supply chains.

The proposed system posits that by integrating an LLM with a graph-based supply chain knowledge base, we can enable interactive Q&A in natural language that is both rich in context and grounded in actual data. The LLM provides natural language understanding and generation capability, allowing users to simply ask questions or seek explanations in plain English. The graph provides the factual backbone: a retrieval mechanism can pull relevant nodes, relationships, or subgraphs from the knowledge base to feed into the LLMs response. In essence,

the system will augment the language model with supply chain knowledge on-the-fly, ensuring that answers cite real data. This combination aims to overcome the limitations of black-box AI by delivering explainable insights.

Natural language interaction is a key part of the solution because it aligns with how humans reason and inquire. Supply chain domain experts may not know how to code or query databases, but they know the questions they need answered. Early indications of industry interest in this direction are evident: major companies like Caterpillar have explored natural language dialogue systems for maintenance and supply chain data [3]. Similarly, enterprises are looking to back their LLMs with knowledge graphs to get more reliable business AI solutions, rather than using language models in isolation.

1.2 Motivation

This project is motivated by three main factors: business needs, technological advancements, and academic interest.

During my internship on the Supply Chain Planning team at Accenture, I witnessed how frequently operational questions emerged and how much time was required to obtain clear answers. Planners need rapid insights into bottlenecks, alternative sourcing options and risk mitigation, but must often rely on specialized data analysts to query disparate systems, compile reports and interpret results. This dependency introduces delays that can hinder timely decision-making.

From a business standpoint, Accenture recognized the potential value of a conversational tool that would allow logistics managers to pose questions in natural language and receive immediate, data-driven responses. Such a prototype could reduce analysis costs, accelerate planning cycles and offer a competitive advantage by enabling clients to interact directly with their control-tower data without writing queries or awaiting analyst support.

Technologically, integrating Retrieval-Augmented Generation with a graph-based knowledge store presents an underexplored opportunity in global supply chains. While Graph RAG has shown promise in domains such as biomedicine [4], improving factual accuracy and explainability, few implementations address the challenges of multi-tier logistics data. Developing a working prototype will validate methods for ingesting data into a graph database, retrieving relevant subgraphs and using an LLM to generate coherent explanations. Additionally, measuring the end-to-end latency of retrieval, inference and presentation will establish performance metrics essential for enterprise adoption.

Academically, this thesis represents the capstone of my Big Data studies. It brings together skills in graph modeling, data pipeline orchestration, generative-model APIs and user experience evaluation. The results will not only enrich my portfolio but also provide empirical evidence on the practical benefits and limitations of Graph RAG architectures in industrial settings.

In summary, the motivation for this project is threefold: to meet Accenture's business need for faster, self-service analytics; to explore a novel technological application of Graph RAG in supply chain planning; and to fulfill academic objectives by rigorously evaluating an emerging conversational framework. These driving factors converge on the goal of demonstrating that a graph-powered RAG system can transform how supply chain teams access, interpret and act upon their own data.

1.3 Objectives

This project pursues a set of goals aligned with the motivations already mentioned in the previous section. A primary objective is defined alongside several sub-objectives.

1.3.1 Primary Objective

To develop a conversational prototype based on Graph Retrieval-Augmented Generation that enables non technical users to obtain rapid, explainable answers to operational questions.

1.3.2 Sub-objectives

1. Conduct a systematic review of the state of the art and existing approaches, identifying the most relevant advances and predominant techniques in retrieval-augmented generation.

2. Select a graph database and design a coherent schema representing supply chain entities and relationships. Load a test dataset to serve as the basis for subsequent development phases.

3. Develop the functional core of the Graph RAG system, establishing the mechanisms required to link natural-language queries with the underlying graph data.

4. Evaluate the prototype using a set of representative questions, measuring response accuracy with different metrics, time-to-insight and the degree of explanatory clarity for the end user.

5. Produce comprehensive, structured documentation describing the theoretical foundations, development process, evaluation results and guidelines for potential extension or deployment.

Through this work, the goal is to demonstrate the feasibility and added value of combining graph technologies with advanced language models for supply chain analysis and knowledge extraction.

Chapter 2

State of art

In this chapter, we review the scientific and technological underpinnings relevant to integrating graph-augmented retrieval and large language models within supply chain data systems. We begin by examining modern techniques for supply chain data analysis, we then explore large language models, tracing their evolution from Transformer-based architectures to their deployment in conversational question-answering systems, with attention to strengths and limitations. Building on this, we present Retrieval-Augmented Generation (RAG) as a method to enhance factual accuracy by combining generative models with external memory. Then, we extend RAG into Graph RAG, highlighting how knowledge graphs can improve multi-hop reasoning and transparency. The last section introduces the core concepts of graph databases, emphasizing their suitability for representing and querying networked supply chain data, and concludes with a focused overview of Neo4j as a mature platform that supports both graph storage and integration with AI systems. Collectively, this chapter lays the conceptual groundwork for the proposed Graph RAG+LLM system in subsequent chapters.

2.1 Supply Chain Data Analysis: Methods and Challenges

Global supply chains generate vast and complex data that require sophisticated analysis to ensure efficiency and resilience. Recent global crises and regulatory pressures have highlighted the need for greater supply chain transparency and risk management. For example, the European Union's Raw Materials Initiative (RMI) underscores the importance of mapping critical raw material supply chains and diversifying sources to reduce dependency [5]. However, obtaining end-to-end visibility in supply networks remains difficult. Information beyond first-tier suppliers is often opaque or incomplete, making it hard to identify vulnerabilities deep in the supply chain. In one survey, nearly 80% of companies could not name the number of their second-tier or deeper suppliers, indicating prevalent data blind spots in multi-tier networks [2]. This lack of visibility hampers precise risk forecasting and proactive mitigation. Traditional supply chain data analysis has relied on enterprise resource planning (ERP) systems, spreadsheets, and manual reporting, but these tools struggle with the scale and interconnectedness of modern supply chain data. In the era of Industry 4.0, the volume and velocity of supply chain data have increased dramatically, exacerbating the challenges of data integration and analysis [6]. The relationships in a supply network form a graph (or "supply web") structure with multiple tiers of suppliers and customers. Conventional relational databases are often ill-suited for capturing such multi-hop relationships and may lead to poor performance or even fragile analytics when dealing with deeply linked data [6]. For instance, recursive queries to trace a part through several supplier tiers can become computationally expensive in a relational model. To address these challenges, recent approaches have adopted graph-based data models and advanced analytics. Modeling a supply network as a knowledge graph enables the integration of heterogeneous data sources and facilitates reasoning over complex relationships [2]. Liu *et al.* (2023) present a framework for supply chain resilience using knowledge graphs: by connecting disparate supplier and procurement datasets, they achieved visibility up to tier-3 suppliers and applied graph algorithms to identify critical nodes (e.g., single points of failure in the network) [2]. They also demonstrated the use of machine learning for link prediction in the supply graph, inferring missing supplier relationships with reasonable accuracy. This knowledge-driven approach supports risk identification by uncovering, for example, that many suppliers might be concentrated in a single region or share a high-risk profile [2]. Such insights are difficult to obtain with traditional tabular data analysis. Another emerging method in supply chain analytics is the application of natural language processing (NLP) to unstructured data (e.g., maintenance logs, procurement documents, news feeds). Large manufactur-

ers have explored combining NLP with graph data models to capture and query domain knowledge. For example, Chandler (2018) describes an industrial system at Caterpillar that uses a graph database to store maintenance and supply chain information, allowing users to query it via natural language dialogues [3]. In this approach, free-form text data (work orders, incident reports, etc.) are parsed and mapped into a graph of entities (machines, parts, locations) and relationships. The graph representation provides a flexible yet structured way to link textual knowledge (such as a description of a part failure) to the broader supply network context. This enables powerful queries like, "Which critical components sourced from Supplier X have experienced failures in the last year?" to be answered by traversing the graph. The use case demonstrates that graphs can serve as a backbone for NLP systems at scale, bridging the gap between unstructured text and structured supply chain data [3]. In summary, supply chain data analysis is evolving from siloed, manual approaches toward more automated, intelligence-driven methods. Key challenges include data silos and heterogeneity (structured ERP data vs. unstructured text), limited visibility across multiple tiers, and the need for real-time risk assessment. Contemporary methods address these challenges by integrating data into knowledge graphs and applying advanced analytics: graph algorithms to analyze network connectivity, machine learning to predict disruptions, and NLP to extract insights from text. These methods come with their own limitations, such as the effort required to build and maintain a comprehensive knowledge graph and potential data quality issues (incomplete or inconsistent data from suppliers). Nonetheless, they lay the groundwork for more resilient and data-informed supply chain management, which is crucial in an increasingly volatile global environment.

2.2 Large Language Models (LLMs)

Large Language Models (LLMs) are a class of AI systems distinguished by their size (hundreds of millions to hundreds of billions of parameters) and their ability to learn from massive text corpora. By leveraging deep neural architectures, particularly the Transformer [7], LLMs have achieved remarkable proficiency in natural language tasks. Notable examples include BERT [8]) for language understanding and GPT-3 [9] for text generation. GPT-3, with 175 billion parameters, demonstrated that scaling up model size and training data can yield emergent capabilities: it can perform tasks such as question answering, translation, and summarization with little or no task-specific training (so-called few-shot learning). This marked a turning point in NLP, as LLMs began to serve as general-purpose language tools or foundation models, applicable to a wide range of domains.

LLMs operate by predicting the next word in a sequence, but their massive training on diverse internet text allows them to capture a broad spectrum of factual and linguistic knowledge implicitly. In the context of supply chain data systems, LLMs offer potential benefits. They could, for instance, interpret and generate reports from logistics data, understand complex supply chain documents, or answer questions posed in natural language by synthesizing information from multiple sources. Early research has explored fine-tuning LLMs on domain-specific corpora (such as maintenance records or procurement contracts) to adapt their knowledge to supply chain contexts. The result would be AI assistants capable of analyzing unstructured supply chain information at scale and aiding decision-making with human-like language explanations.

Despite their impressive capabilities, LLMs also have significant limitations. Firstly, they store knowledge implicitly in model weights, which means their knowledge is static up to the time of training and not easily updated. They have broad but shallow knowledge in specialized fields, often lacking the depth or the most up-to-date facts needed for expert tasks [10]. For example, an LLM trained on general web data might not know the latest disruptions in a specific supply chain or the intricate details of a company's supplier network. Secondly, LLMs can produce information that appears fluent and convincing but is factually incorrect or fabricated – a phenomenon known as hallucination. Because they lack direct access to ground truth databases when generating answers, they may fill gaps with plausible-sounding but erroneous statements [10]. This is particularly problematic in a supply chain scenario, where decisions based on incorrect information (e.g., a wrong lead time or supplier location) could have serious consequences. Another concern is that LLMs do not naturally provide provenance for their outputs. They cannot explain why they gave a certain answer or cite a source, which undermines trust in critical applications. Finally, the computational requirements of training

and deploying LLMs are immense, raising practical issues of cost and integration into existing systems.

Researchers and practitioners are actively seeking ways to mitigate these issues while harnessing LLMs' strengths. One promising direction is to augment LLMs with external knowledge sources so that the model can retrieve up-to-date, relevant information when needed, rather than relying solely on its internal memory. By doing so, we can enhance the factual accuracy of LLM outputs and enable the model to cite evidence for its statements. The next section discusses such an approach, Retrieval-Augmented Generation, which combines an LLM with a retrieval mechanism to improve performance on knowledge-intensive tasks.

2.3 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) is a technique that integrates large language models with external knowledge retrieval to overcome some of the limitations of standalone LLMs. In a RAG architecture, the language model is coupled with a retriever module that can access a large collection of documents or a knowledge base. For a given query or prompt, the system first retrieves relevant information (e.g. text passages, articles, or facts) from the external source, and then the language model conditions its generation on the retrieved evidence. This process allows the model to inject up-to-date and detailed knowledge into its responses, rather than relying purely on what was seen during pre-training [10].

The RAG framework was introduced by Lewis *et al.* (2020) for knowledge-intensive NLP tasks [10]. In their seminal work, a pre-trained sequence-to-sequence model (BART) was augmented with a dense vector retriever that indexes the entire Wikipedia corpus. Given a question, the retriever finds the most relevant passages from Wikipedia, and the generator model then produces an answer using not only its parametric knowledge but also the retrieved text. Two variants were explored: one where a fixed set of passages is retrieved once for the whole answer, and another where new passages can be fetched at each decoding step. The results were compelling: RAG achieved state-of-the-art performance on open-domain question answering benchmarks, outperforming models that had to rely only on encoded knowledge in their parameters [10]. Moreover, the answers generated by RAG were observed to be more specific and factual, closely referencing the retrieved documents, as opposed to the more generic outputs of parametric-only models.

Strengths of RAG: The primary advantage of RAG is that it marries the flexibility and fluency of LLMs with the precision of information retrieval. The external knowledge base serves as a dynamic memory. It can be updated independently of the model (for example, by adding new documents), immediately injecting new information without retraining the language model. This makes RAG systems adaptable in fast-changing domains like supply chain management, where new supplier information or regulatory updates can be incorporated on the fly. RAG also provides a form of transparency: since the model's output is grounded in retrieved passages, a user can be shown the sources that the model used (addressing the provenance issue to some extent). In knowledge-intensive scenarios, RAG significantly reduces hallucinations, because the model has factual text to draw from and is implicitly encouraged to stay faithful to that evidence. Lewis et al. report that their RAG models produced substantially more accurate statements on factual QA tasks compared to a purely generative baseline, thanks to grounding in external text [10].

Despite these benefits, RAG is not without challenges. The effectiveness of a

RAG system hinges on the quality of the retrieval component. If the retriever fails to fetch the relevant documents for a query, the generator may still produce an incorrect or unsupported answer (and perhaps with misplaced confidence). Therefore, RAG inherits the classic information retrieval problems of recall and precision. Another issue is that splitting knowledge into discrete documents or passages (as is typically required for indexing) may lead to fragmentation of information. A single fact might be spread across multiple documents, requiring multi-hop reasoning. Traditional RAG, which usually retrieves a few top documents based on similarity to the query, might miss information that is not explicitly mentioned in any single document but can be inferred by connecting pieces from several sources. Handling complex queries that involve multiple entities or relationships (for instance, "Which suppliers of Supplier X also experienced delays due to the same hurricane in 2023?") remains challenging for text-only RAG. The model might need to retrieve separate passages about Supplier X's suppliers and about the hurricane's impact, then connect them—something beyond the capability of naive retrieval algorithms that treat each query independently.

There is ongoing research to address these limitations, and one notable direction is the use of structured knowledge and graphs within the RAG paradigm. By incorporating graph-based retrieval or knowledge graph information, it is possible to better handle multi-hop queries and exploit relationships between pieces of data that a flat document index would overlook [11]. In the next section, we discuss Graph RAG, which extends the RAG concept by using graph-structured knowledge bases and has shown promise in tasks requiring complex reasoning and domain-specific knowledge integration.

2.4 Graph RAG

Graph Retrieval-Augmented Generation (Graph RAG) is an emerging paradigm that combines large language models with graph-structured knowledge sources to enhance information retrieval and reasoning. In Graph RAG, the external memory accessed by the language model is not a corpus of independent text documents, but rather a knowledge graph or a graph-based index of the knowledge. This approach retains the core idea of RAG—grounding LLM outputs in external data—but leverages the rich relational structure of graphs to overcome some limitations of text-based retrieval. A recent survey by Procko and Ochoa (2024) provides a comprehensive overview of Graph RAG techniques, highlighting their benefits in complex query scenarios and multi-relational data environments [11].

By using graphs, Graph RAG can capture complex relationships between entities, enabling more nuanced understanding and multi-hop reasoning. For example, consider a supply chain knowledge graph where nodes represent organizations, facilities, and products, and edges represent supplier relationships or material flows. A Graph RAG system could retrieve not just isolated facts about one supplier, but an entire subgraph connecting a manufacturer to its tier-2 and tier-3 suppliers when answering a query about supply chain risk. This subgraph retrieval provides the language model with context that preserves the relationships (e.g., the chain of dependencies or geographic co-locations) inherent in the query. Traditional RAG would require stitching together multiple documents to get the same connected context, whereas Graph RAG can directly navigate the knowledge graph to gather relevant interconnected information.

The advantages of Graph RAG over text-only RAG have been noted in the literature [11] and can be summarized as follows:

- Enhanced Knowledge Representation: Graphs store knowledge in a structured form, with nodes and edges encoding entities and their relations. This enables the system to consider not just keyword overlap, but the actual semantic relationships when retrieving information. A graph can represent hierarchies, causality, and other relationships (e.g., part-of, located-in) that are crucial for complex reasoning but might be implicit or scattered across texts. By traversing the graph, Graph RAG can discover non-obvious connections and answer queries that require joining facts from different parts of the knowledge base.

- Better Multi-hop Reasoning: Because of the explicit links in a knowledge graph, Graph RAG can naturally perform multi-hop retrieval. It can follow a chain of edges to gather evidence that spans multiple hops (for instance, finding all suppliers indirectly affected by a raw material shortage via their

supply network). This logical traversal goes beyond what a vector-similarity search can do. Empirical studies indicate that Graph RAG can handle complex queries with multiple intent more effectively, reducing the risk that the model misses a critical intermediate piece of information.

- Efficiency and Context Preservation: Graph databases are optimized for relationship-centric queries, which means a Graph RAG system can retrieve relevant knowledge subgraphs quickly even in a large dataset. By focusing on the subgraph that is relevant to a query, the language model does not need to process as much extraneous text. This can lead to efficiency gains; in some cases Graph RAG responses were generated with a fraction of the tokens needed by a traditional RAG approach, due to more focused context. Additionally, graphs can be updated incrementally (adding or removing nodes/edges) without reprocessing an entire corpus, allowing real-time knowledge updates.

- Interpretability and Trust: The use of a knowledge graph offers a transparent window into the model's reasoning process. Users or auditors can visualize the retrieved subgraph and see the chain of relationships that led to a given answer. For high-stakes domains like healthcare, finance, or supply chain, this interpretability is valuable. It provides an explanation of which facts and connections the model considered, thereby building trust. In contrast, a pure text RAG might retrieve a document, but it's less clear which part of the text or which combination of texts led to the conclusion. Graphs naturally document the reasoning path (e.g., Company A $\rightarrow$ Company B $\rightarrow$ Factory C for a supply chain risk path).

Early use cases of Graph RAG underscore these advantages. In the medical domain, for instance, Graph RAG has been applied to question answering where the knowledge source is a biomedical knowledge graph (genes, diseases, drugs, etc.). The LLM can retrieve a biomedical subgraph that links a gene to a pathway to a disease, providing grounded answers to complex clinical queries. In the legal domain, researchers have used Graph RAG to navigate legislative documents: a knowledge graph of laws and regulations allows an LLM to pinpoint relevant statutes across a large body of text when answering legal questions. In the context of supply chains, one can envisage Graph RAG being used to query a supply chain knowledge graph for resilience analysis—for example, identifying how a disruption in one region might propagate through suppliers and which products would be affected. By leveraging the structured relationships (supplier-of, located-in, uses-material) in the graph, the LLM can generate a report that is both factually accurate and tailored to the specific query, with the backing of the graph data.

Despite its promise, Graph RAG is a nascent field and comes with limitations and challenges. One major challenge is the requirement of a high-quality, up-to-date knowledge graph. Many domains do not have readily available knowledge graphs, and building one (via knowledge graph construction or manual curation) can be labor-intensive and error-prone. If the graph is incomplete or contains erroneous relationships, the Graph RAG system's performance will suffer; it might overlook answers that lie outside the known graph or propagate the graph's errors. This issue is analogous to the knowledge base quality problem in traditional expert systems. Another limitation involves knowledge integration: how to effectively encode the retrieved subgraph into a form that the language model can understand. Graph data might need to be linearized (turned into text or a list of triples) or embedded, and there is a risk of losing some structural information in that conversion. Furthermore, large knowledge graphs pose scaling issues. While graph databases handle big data well in terms of storage and retrieval, the language model still has a finite context window. If a query requires a very large subgraph as context, it might exceed the token limit of the model. Research is ongoing into techniques for summarizing or chunking graphs to fit within model constraints without sacrificing critical information.

Additionally, Graph RAG inherits concerns about knowledge conflicts and consistency. When multiple sources or multiple paths in a graph provide conflicting information (for example, two different data sources give different delivery lead times for the same supplier), resolving these conflicts is non-trivial. The LLM might need to decide which source to trust or how to reconcile differences, which is an open problem. Privacy is another consideration: supply chain knowledge graphs for a company might contain sensitive business information, so any Graph RAG deployment needs to ensure secure handling of data and adherence to privacy regulations.

In summary, Graph RAG extends retrieval-augmented generation by using graphs to organize and supply knowledge to LLMs. It shows clear strengths in representing and reasoning about connected data, making it highly relevant for domains like supply chain management where relationships are paramount. Successful early applications in various fields suggest that Graph RAG can improve accuracy and interpretability of LLM-driven systems. At the same time, constructing and maintaining the graph knowledge base, as well as integrating it with LLMs efficiently, remain active areas of research. As Graph RAG techniques mature, we expect them to play a significant role in enterprise AI systems, including those aimed at analyzing and optimizing supply chain networks. The approach naturally dovetails with graph database technology, which we explore next.

2.5 Graph Databases: Concepts and Applications

Graph databases are specialized data management systems designed to store and query data as a network of nodes and edges. Unlike traditional relational databases that arrange data into tables, graph databases use a flexible schema-less structure: entities are nodes, relationships between entities are edges, and both can have properties (key-value pairs) describing them. This model—often called the labeled property graph model—is intuitive for representing connected information. As Robinson et al. (2013) famously note, with a graph database "what you sketch on the whiteboard is typically what you store in the database" [12]. In other words, the graph model allows data to be stored very closely to how we naturally think about relationships, whether it's social networks, transportation routes, or supply chain linkages.

In a graph database, queries are usually expressed in terms of patterns to match in the graph (for example, find a path of length 3 between Node A and Node B with certain properties). This contrasts with SQL queries on relational databases, which require multiple JOIN operations across tables to traverse relationships. Graph query languages like Cypher (originally developed by Neo4j) or SPARQL (for RDF graph stores) provide a high-level syntax to specify these patterns. For instance, a Cypher query might look for all suppliers of a given company that are located in a certain country by traversing "SUPPLIES" relationships and filtering on a location property. Under the hood, graph databases index these relationships, making such multi-hop traversals efficient. Research and benchmarks have shown that for highly interconnected data, graph databases can outperform relational databases by orders of magnitude on queries that involve traversing many relationships [12]. A classic example is the "friends-of-friends" query in a social network: finding friends-of-friends three levels deep can be extremely slow in SQL if not carefully indexed, but graph databases handle it readily since they can explore the graph neighborhood in linear time with respect to the number of relevant edges. Robinson et al. report that a 3-hop path query was over 10,000 times faster on a native graph database compared to a relational system in certain scenarios without specialized indexing [12]. Even with indexing in the RDBMS, graph systems tend to maintain an edge in such traversal-heavy queries.

Applications of Graph Databases. The strengths of graph databases make them well-suited for a variety of applications.

- In knowledge graph management, graph databases are used to store large knowledge bases (like DBpedia or enterprise knowledge graphs), supporting

semantic queries and inference.

- In social networks and recommendation systems, graphs naturally model users, their connections, and their interests; companies like Facebook and LinkedIn have graph backends to compute things like "people you may know" or to detect communities.

- In fraud detection, financial institutions use graph databases to uncover fraud rings by seeing connections between accounts, transactions, and entities.

- Another pertinent application domain is supply chain and logistics. Supply chain data—encompassing suppliers, manufacturers, distribution centers, shipments, etc.—forms a complex network that can be efficiently managed in a graph database. Hong and Chen (2022) demonstrate a graph database solution for automotive supply chain resilience: they created a labeled property graph of their supply network and were able to run analytics like "time-to-stockout" computation and multi-tier supplier queries with high efficiency [13]. Their framework showed that recursive queries (e.g., finding all suppliers up to n-th tier or calculating the impact radius of a disruption) were handled elegantly by the graph model, which would have been cumbersome with a relational approach [13]. Moreover, graph algorithms such as centrality measures can be applied directly to identify key suppliers (nodes with high centrality) or vulnerable connections in the supply chain network.

Graph databases also play a critical role in retrieval-augmented pipelines as discussed in previous sections. When implementing a RAG or Graph RAG system, a graph database often serves as the underlying storage for the knowledge to be retrieved. For example, if a company constructs a knowledge graph of its supply chain (integrating data from procurement, logistics, and external sources), a graph database like Neo4j or Amazon Neptune could store this graph. An LLM-based assistant could then query the graph database (via Cypher or SPARQL queries) to fetch relevant subgraphs or facts in response to a user's question. This integration allows the pipeline to exploit the performance and query power of graph databases in retrieving relevant information, which is then fed into the language model for generation. Such a setup combines the best of both worlds: the rigor and connectivity of graph-stored knowledge with the fluent explanatory power of LLMs.

While graph databases are powerful, they are not a silver bullet for all data problems. One limitation is that they may not match the performance of relational databases for heavy analytical queries that involve scanning large portions of the data (e.g., aggregating a numerical column across millions of rows) – tasks where columnar SQL databases excel. Graph databases shine in traversal queries,

but pure aggregation or extremely large graph analytics might require additional tools or graph processing frameworks (like Apache Spark GraphX or Neo4j's own Graph Data Science library). Scalability can also be a concern: scaling a graph database horizontally (across many servers) is a complex task due to the interlinked nature of the data. Recent advancements and distributed graph systems (e.g., Neo4j Fabric, TigerGraph) have addressed this to some extent, but designing sharding or partitioning strategies that preserve graph locality is non-trivial. Another challenge is the relative novelty of graph querying for many developers and analysts — there is a learning curve to adopt a new query language and mental model, as opposed to the long-established SQL for relational databases. However, with growing interest in knowledge graphs and network analytics, graph databases are becoming more mainstream, aided by an expanding ecosystem of tools and standards (for instance, the ongoing development of GQL, a standardized graph query language by the ISO).

In conclusion, graph databases provide a robust platform for managing connected data. Their data model aligns well with real-world networks, and their performance advantages manifest in scenarios where relationships matter more than individual data points. In the realm of supply chain systems, graph databases enable the creation of a "digital twin" of the supply network, supporting advanced analytics and AI-driven insights. As we turn to a specific graph database in the next subsection, Neo4j, we will see how these concepts materialize in a concrete system that has been widely adopted in both academia and industry.

2.5.1 Neo4j

Neo4j is one of the leading graph database management systems and a prominent example of a property graph database. First released in the late 2000s, Neo4j was a pioneer in the NoSQL movement, offering an ACID-compliant transactional database entirely optimized for graph storage and traversal. In Neo4j's data model, nodes represent entities and can have one or more labels (types), while relationships (edges) connect nodes and are directed and named (e.g., `SUPPLIES`, `LOCATED_IN`). Both nodes and relationships can hold properties in the form of key-value pairs, allowing rich metadata to be stored directly in the graph structure. This flexible schema-less design means data can evolve without the need for costly migrations, new node types or relationship types can be added as needed, reflecting changes in the domain model.

One of Neo4j's most notable contributions is the development of the Cypher query language, which has a SQL-like syntax tailored to graphs. Cypher uses pattern matching with an ASCII-art style representation of nodes and relationships. For example, a Cypher query to find suppliers in Germany that provide parts to a company might look like:

```
MATCH (supp:Company)-[:SUPPLIES]->(part:Part)<-[:BUYS]-(buyer:Company)
WHERE buyer.name = "ACME Corp" AND supp.country = "Germany"
RETURN supp.name, part.id;
```

This query traverses the graph pattern of a supplier supplying a part that the buyer (ACME Corp) buys, filtering by supplier country. The readability of Cypher and its expressive power (including support for variable-length path patterns, aggregations, etc.) have made it a popular choice for graph querying. In fact, Cypher influenced the design of GQL, the forthcoming standard graph query language.

Applications and Strengths: Neo4j has been widely adopted in industries ranging from finance to telecommunications, and has also been used in numerous academic projects. In the supply chain context, as mentioned earlier, Neo4j has been employed to model supply networks. Its ability to quickly traverse multi-tier relationships, thanks to the set of native Cypher functions and syntax, is invaluable for questions like "find all indirect suppliers of a given component" or "identify the shortest supply path between two locations of the network.". Hong and Chen (2022) implemented their supply chain resilience graph model on Neo4j and reported significant performance gains: queries that involve exploring recursive supplier relationships ran efficiently, and the graph schema allowed them to naturally represent complex structures like alternative suppliers and bills-of-materials [13]. In another example, Caterpillar's maintenance and supply chain knowledge graph was built on Neo4j to enable interactive question-answering; Neo4j's fast graph traversal meant that even complex queries through the equipment-failure-service networks could be answered in near real-time [3]. These cases highlight Neo4j's strength in handling operational graph queries (many short transactions of traversals) as well as serving as a backbone for knowledge-driven applications.

Neo4j also provides an ecosystem of tools that enhance its applicability. The Neo4j Graph Data Science library offers algorithms for community detection, centrality, similarity, and more, which can run natively on the graph. This is useful for supply chain analytics—e.g., finding clusters of suppliers that are tightly interconnected, or key nodes whose removal would fragment the network (a measure of vulnerability). Neo4j Bloom, a visualization tool, allows interactive exploration of the graph, which can be helpful for analysts to inspect supply chain structures or for presenting insights to stakeholders in an intuitive graph format. Moreover, Neo4j supports integrations with various programming languages (via drivers) and can be deployed in cluster configurations for high availability and scalability.

Limitations: While Neo4j is powerful for many use cases, it does have limitations to be aware of. For extremely large graphs (billions of nodes/edges), Neo4j's performance can degrade if the hardware is insufficient, and sharding such a graph across multiple servers can be complex. The enterprise edition of Neo4j introduces features like fabric (for sharding) and causal clustering, but effectively partitioning

a graph without breaking important relationships is a non-trivial task that often requires domain-specific strategies. In addition, Neo4j (like most graph databases) lacks the kind of ad-hoc analytical querying that SQL data warehouses excel at. If one needs to perform heavy number-crunching on large tabular data extracted from the graph (e.g., summing total trade volumes across all routes), it might be more efficient to export or mirror some data to a relational or analytical store. There is also the consideration of learning curve and tooling: while Neo4j's community is strong and growing, many organizations still have deep expertise in SQL and may find it challenging to adopt a new technology stack. However, Neo4j has made strides in ease-of-use, and the emergence of standards (OpenCypher, GQL) and interoperability (e.g., GraphQL integrations) is lowering the barrier.

In the context of retrieval-augmented generation systems and AI pipelines, Neo4j serves well as the graph backend. Microsoft's recent "Graph RAG" initiatives [14], for example, provide accelerators that connect LLMs (like Azure OpenAI's GPT models) with Neo4j graph data, enabling developers to build applications where an LLM answers questions by querying a Neo4j knowledge graph in real-time. This kind of architecture exemplifies the practical convergence of the topics discussed in this chapter: supply chain data (stored in a graph database like Neo4j), large language models (to interpret queries and generate answers), and retrieval-augmentation (using the graph as the source of truth for factual information).

To conclude, Neo4j is a mature and feature-rich graph database that embodies the benefits of graph data management. Its use of the property graph model and Cypher query language has set a foundation that many other systems follow. In supply chain analytics, Neo4j provides a way to bring together diverse data (suppliers, locations, shipments, risks) into a connected structure that can be efficiently queried and analyzed. When combined with large language models in a retrieval-augmented framework, Neo4j helps ensure that the AI's answers are grounded in the reality of the data, leveraging the power of graphs to inform and support advanced decision-making processes.

Chapter 3

Proposed Solution

This chapter presents the solution proposed in this thesis, a system designed to enhance decision-making processes in supply chain management through Graph Retrieval-Augmented Generation. The primary functionality of this system involves receiving a natural language query from the user, subsequently performing targeted queries against a specialized graph database representing the supply chain, and finally generating coherent, contextually relevant natural language responses.

In the following sections, we will describe how supply chain data have been structured and modeled into graph form to efficiently capture relationships and dependencies among entities. We will then outline the overall architecture of the proposed system, highlighting how various components interact to achieve accurate graph retrieval and natural language generation. Lastly, we provide comprehensive technical details regarding the implementation, covering the technologies utilized to build this solution.

3.1 Data Modeling for Supply Chains

3.1.1 Data Structure Design

In this project, the proposed solution has been developed within Accenture, which already maintains a standardized model for supply chain data representation. This data model primarily utilizes a graph structure consisting of nodes and lanes. Specifically, nodes represent pairs of materials and locations, while lanes depict the relationships or pathways connecting these nodes, classified into two main categories: manufacturing lanes and transportation lanes.

Figure 3.1 provides a concrete example of the structure used in Accenture's supply chain network. The figure illustrates a simplified supply chain scenario, showing both transportation and manufacturing lanes, as well as several material-location nodes. This visual representation serves to clarify how the Accenture standardized graph model represents different types of operational flows and material transfers within the network.

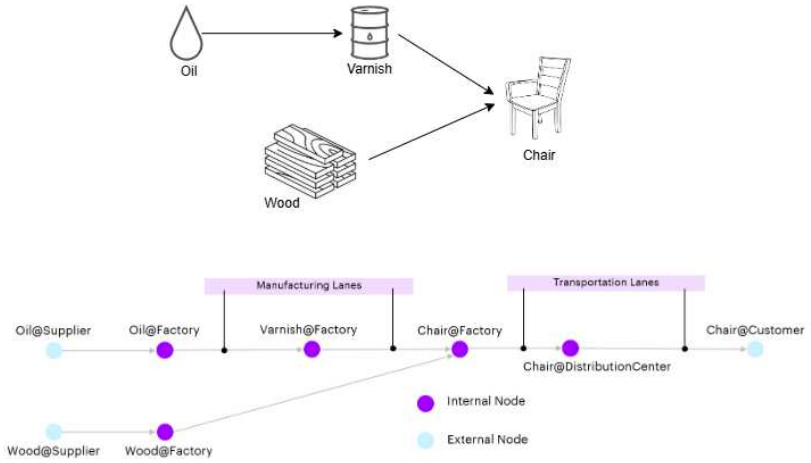

Figure 3.1: Standardized graph model representing a simple supply chain of the finished good chair. Each node is named using the format `<material>@<location>`.

Each node in the graph carries a unique identifier (ID) along with various descriptive fields. Material-related information includes details such as material descriptions, names, brands, inventory levels, and prices. Additionally, location-specific information comprises geographical coordinates and detailed descriptions. All the fields used are described in Table 3.1.

Similarly, lanes are characterized by distinct identifiers and information fields. Table 3.2 describes all the fields contained in this object.

Field	Description
material_id	Unique identifier of the material
location_id	Unique identifier of the location
node_id	Unique identifier for the node itself, composed of material and location ids concatenated
material_type	Type of material (e.g., raw material, semifinished good, finished good)
location_type	Type of location (e.g., warehouse, plant)
location_desc	Description of the location
location_name	Name of the location
address	Physical address of the location
city	City of the location
region	Region of the location
country	Country of the location
material_desc	Description of the material
base_uom	Unit of measure (e.g., kg, m^2)
material_unit_price	Unit price of the material
material_group	Material grouping for categorization
material_hierarchy	Hierarchy classification of the material
business	Business unit or division
category	Category of the material
brand	Brand of the material
inventory_qty	Average inventory quantity
avg_monthly_purchased_qty	Average monthly quantity purchased
avg_monthly_purchased_price	Average monthly purchase price
avg_monthly_selling_qty	Average monthly selling quantity
avg_monthly_selling_value	Average monthly selling value
cogs	Cost of goods sold
days_of_inventory	Days of inventory available
imported	Booelan indicator if material is imported

Table 3.1: Description of MatLoc Node Fields

Field	Description
lane_id	Unique identifier of the lane, composed by the two nodes IDs connected by the lane
value	Average monetary value associated with the lane
quantity	Average quantity of materials moved through the lane
lead_time	Time required for transportation or manufacturing through the lane
unit_price	Unit price for the transported or manufactured material
material_from_qty	Quantity of material required on the source node to produce one unit of the material of the destination node, field only used in manufacturing lanes

Table 3.2: Description of Lane Fields

Accenture employs a proprietary Python library to aggregate these nodes and lanes into a unified Network object, thus facilitating the application of specialized analytical algorithms and methods. Due to Accenture's established use of this nodes and lanes structure as an industry standard approach for supply chain related projects, this thesis adopts the same model. Altering this established format would necessitate additional data processing and introduce unnecessary complexity, thereby potentially hindering seamless integration with existing project workflows and tools.

3.1.2 Data Ingestion into Neo4j

The data ingestion process into the Neo4j graph database leverages the Neo4j Data Importer tool [15], which provides a highly visual and intuitive interface designed to facilitate the rapid and efficient transfer of data into Neo4j. This tool supports importing structured data sets, enabling users to visually map and align source data files (typically CSV or JSON formats) directly to graph entities such as nodes and relationships. In this case, nodes and lanes data is stored in CSV files.

Through Data Importer, it is possible to clearly specify how data from external sources maps to the node and lane schemas previously described. This includes setting IDs, labels, defining property mappings, and specifying the relationships connecting nodes, thus simplifying data preparation tasks significantly. Figure 3.2 represents the scheme of the nodes and lanes.

Additionally, Neo4j's Data Importer has the capability of generating corresponding Cypher scripts of the ingestion process. These scripts precisely replicate the import operation, providing an effective backup solution and significantly simplifying re-ingestion tasks, should the data need to be updated or reloaded at any

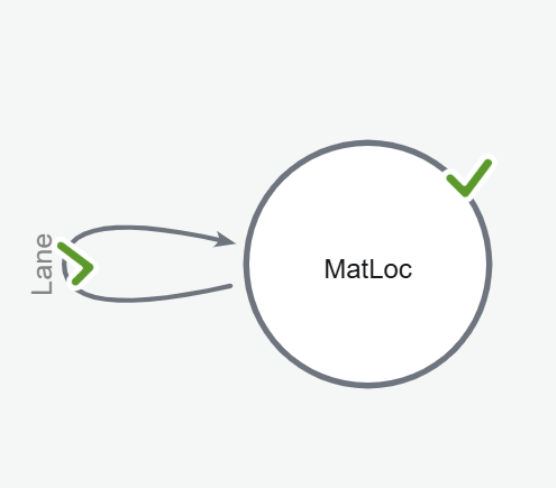

Figure 3.2: Neo4j data importer scheme representing the data model of nodes and lanes.

later point. The availability of these scripts ensures reproducibility capabilities through a simple command execution that triggers the stored script.

3.1.3 Dataset Description

The dataset used for this project has been provided by Accenture and consists of anonymized supply chain data. To preserve confidentiality, no real products, materials, or locations are directly identifiable. Material identifiers follow a generic format such as `material_23453`, while location identifiers take a similar anonymized form like `location_353345`. Although the locations are fictional, each is randomly assigned to a real U.S. city and region to maintain a degree of geographic realism useful for modeling purposes.

In total, the dataset is composed of 252 nodes and 197 lanes, representing a moderately complex supply chain network structure. While this scale is sufficient to test and demonstrate the system's functionality, it is important to note that realistic supply chain networks of large international clients may include up to one million nodes, each representing a unique material-location pair. However, for the development and debugging of the prototype system presented in this thesis, a smaller and more manageable dataset is preferable. It allows for faster iteration cycles, easier visualization, and more efficient identification and resolution of issues.

This network can be visually explored using Neo4j's desktop application, specifically the Neo4j Browser tool, which provides a graphical interface for inspecting the graph. A full view of the graph is showed in Figure 3.3, and a closer view of a subset of the graph is also included in Figure 3.4.

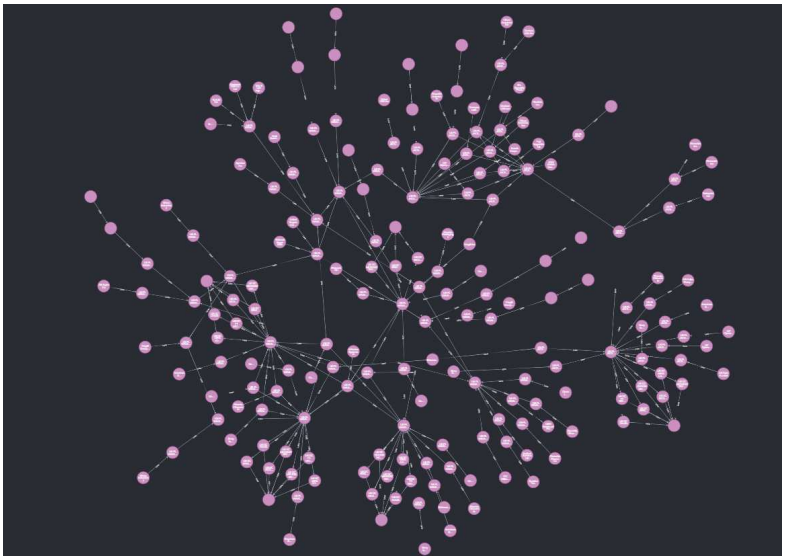

Figure 3.3: Complete visualization of the supply chain graph in Neo4j Browser

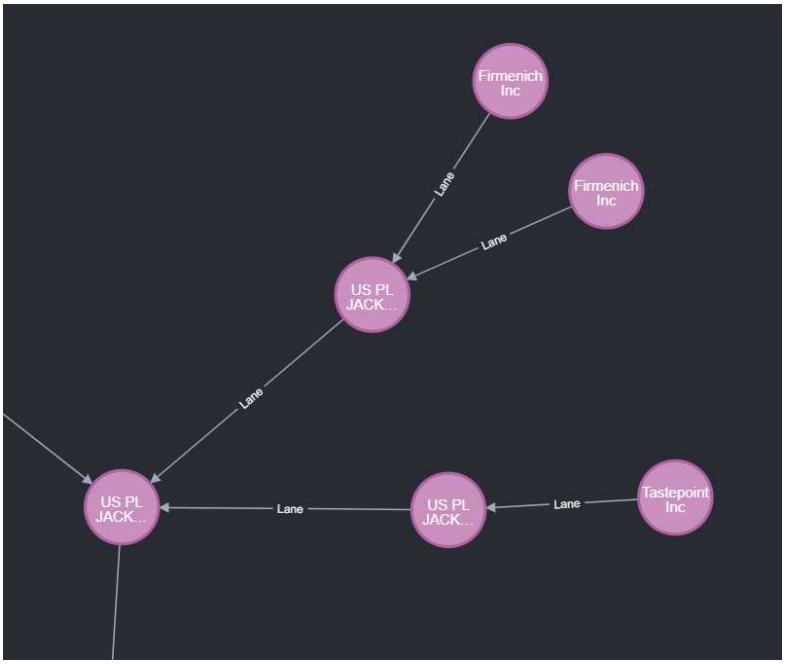

Figure 3.4: Zoom in section of the graph. It illustrates the suppliers Firmenich and Tastepoint, who deliver a material to a manufacturing plant identified as US PL Jacksonville. At this location, a new product (leftmost node) is manufactured from these two materials. Note that each node represents a unique material-location pair, multiple nodes may share the same location while differing in the material component, allowing for a more granular representation of production flows.

3.2 System Architecture & Implementation

In this section, first we look into the general idea behind the project and then we delve into the more technical details and the specific mechanisms that enhance the system's performance.

3.2.1 Architecture overview

The main technologies and their interactions are described in Figure 3.5. The journey from user query to answer can be summarised in the following decisive steps:

1. **Question capture.** A user submits a natural-language question about the supply chain through the Streamlit interface.
2. **Question enrichment.** The question is embedded in a prompt where it is combined with the supply chain context, the neo4j database schema and the user's chat history.
3. **Query creation.** The enriched question is sent to an OpenAI model, which will generate a Cypher query.
4. **Graph retrieval.** The Cypher query is executed against Neo4j, returning an array that contains the requested facts.
5. **Answer generation.** The query response is sent to a second OpenAI call which converts the structured result into a clear, contextually appropriate natural language answer that is returned to the Streamlit interface.
6. **Display response**. Streamlit renders the conversation and displays the final answer returned for the user.

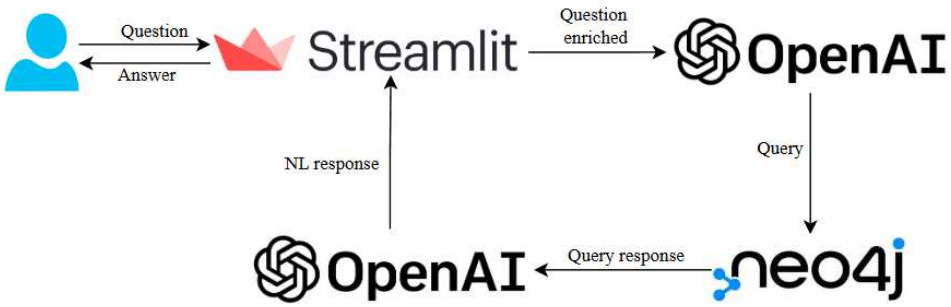

Figure 3.5: High-level flow of the Graph-RAG prototype.

3.2.2 Detailed Workflow and Internal Logic

Figure 3.6 provides an expanded view of the backend of the pipeline, extending the naive view of Figure 3.5 now it has been understood. Streamlit is simply the frontend prototype, it is not included in this diagram as it has no importance to the comprehension of the algorithm.

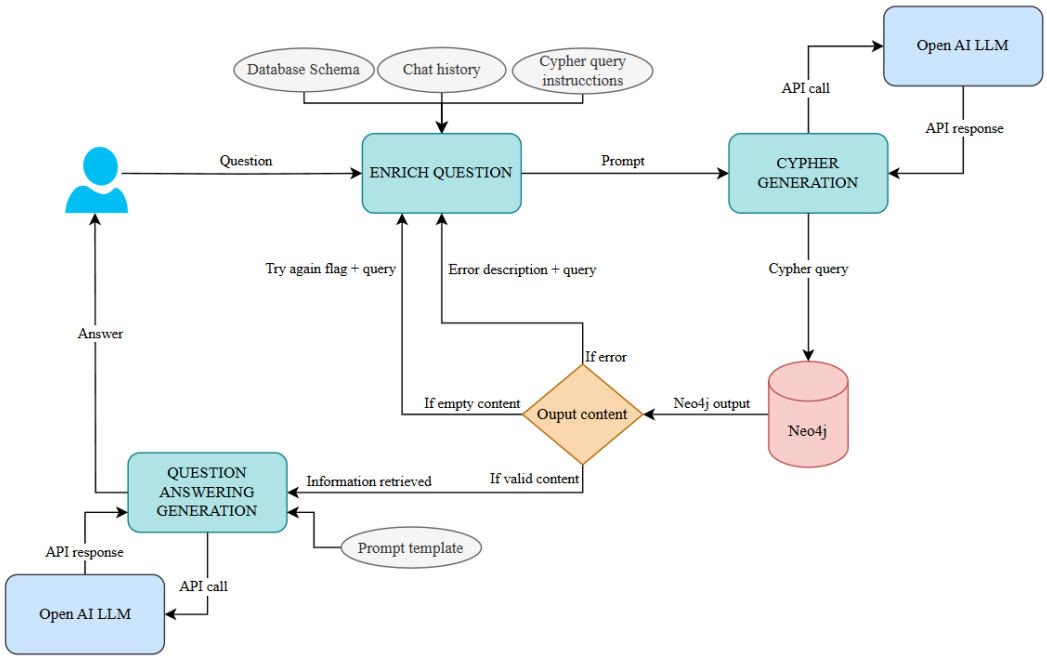

Figure 3.6: System backend detailed logic diagram.

This diagram expose the detailed logic of the algorithm and the mechanism of query generation retries if the output of Neo4j is not valid or empty.

Cypher Generation

The prompt fed to the Cypher generation model is assembled from several distinct components, a process represented as the "Enrich Question" step in the diagram. Initially, this prompt consists of: (i) the user's original question, (ii) the database schema, and (iii) a set of specific instructions that contextualize the supply chain environment and clarify to the LLM the goal of producing a valid Cypher query using the provided schema. Over time, this composition has been enhanced to address common failure scenarios, such as generating invalid queries or receiving empty results from Neo4j. In such cases, the prompt is enriched with additional information, including (iv) the failed query, (v) a flag indicating that an alternative

attempt should be made, and (vi) a description of the error encountered. This allows the model to adjust its output, increasing the likelihood of producing a syntactically correct and semantically relevant Cypher query.

Retry of Cypher Query Generation

The conditional node in the workflow diagram details the possible outcomes following a query to Neo4j. Three distinct branches are represented. First, if an error occurs (such as a syntax error or other execution failure), the workflow loops back, enriching the prompt with the error details and the query that caused it, enabling the model to refine or correct its subsequent query. Second, if Neo4j returns empty content, indicating that the query ran successfully but did not retrieve any relevant information, the workflow also loops back. In this case, a specific flag is set in the prompt instructing the model to attempt generating an alternative Cypher query, with the aim of retrieving information through a different logical path. Third, if Neo4j produces valid content, the workflow proceeds to the question answering stage, where the retrieved information is integrated into the natural language response for the user. However, to not get stuck in an infinite loop of generating empty or wrong answers, the maximum times allowed to generate a new Cypher query is three times.

Question Answering Generation

The prompt for the natural language answer generation model is constructed more simply. It comprises the user's original question, the output returned by Neo4j, and the final Cypher query used for retrieval. These elements provide sufficient context for the language model to generate a coherent and contextually appropriate answer to the user's query, reflecting both the specific data retrieved and the reasoning behind its selection.

Chat Memory Management

To provide conversational continuity and enable the system to reference previous exchanges, a memory mechanism has been incorporated through a `chat_history` variable in the prompt template. This variable stores all questions and responses exchanged between the user and the system within the same session.

The accumulation of excessive conversation history could lead to inefficiencies and increased prompt size. To prevent this, a cap has been imposed on the number of stored messages. Specifically, the chat memory retains up to 30 messages (comprising both user queries and system responses). Once this limit is reached, the oldest entries are discarded as new interactions occur. This sliding window strat-

egy ensures that the model benefits from relevant recent context while avoiding issues related to unbounded memory growth.

3.2.3 Development and Supporting Technologies

The development of this system has been facilitated by several complementary technologies, each contributing to a specific aspect of the project workflow.

For traceability and observability, **LangSmith** has been an essential tool. By integrating LangSmith tracing into the pipeline, it was possible to visualize the end-to-end execution flow of each request, which significantly aided in identifying and resolving bugs. Additionally, LangSmith provides detailed breakdowns of token usage for each model call and precise timing metrics, enabling both performance monitoring and cost estimation.

For rapid prototyping and user interaction, the **Streamlit** framework was employed to build the frontend interface. It was developed based on the official Streamlit guide for building chat applications [16], which provided a solid foundation for implementing conversational features and best practices. A screenshot can be seen in Figure 4.1. Streamlit enabled the creation of an interactive and visually accessible prototype with minimal overhead, allowing for efficient testing and demonstration of the system's capabilities in a real-world usage scenario.

The entire project has been implemented in **Python**, leveraging its rich ecosystem of libraries, such as `langgraph` for the agent framework, `evaluate` to measure the natural language metrics and `pandas` for the initial stage of data exploration, as well as Neo4j and OpenAI integrations. Python's flexibility and readability have been critical in enabling iterative development and quick adaptation to evolving requirements.

Lastly, the use of the **Visual Studio Code** integrated development environment (IDE) has streamlined the development process. Its powerful debugging tools, integrated terminal, and extension ecosystem have been instrumental in identifying issues and ensuring code quality throughout the implementation phase.

Chapter 4

Evaluation and Results

This chapter presents the comprehensive assessment of the proposed Graph RAG system in the supply chain domain. We begin by describing the evaluation methodology and quantitative metrics used to measure system performance. Next, we introduce representative use cases that contextualize our experiments and illustrate realistic scenarios. Finally, we report detailed results, including example queries with generated answers, and analyze system strengths and limitations.

4.1 Evaluation Methodology and Metrics

4.1.1 Methodology

To construct a robust foundation for performance assessment, a domain-specific evaluation dataset is manually assembled through the following steps:

1. **Question Formulation:** Crafting a diverse set of queries that reflect common and complex supply chain information needs.
2. **System Execution:** Submitting each query to the Graph RAG system under consistent runtime conditions, capturing both the generated textual answer and the underlying graph retrieval evidence.
3. **Response Pairing:** Pairing each system-generated response with its corresponding expert-validated answer to prepare for metric-based comparison.

This protocol guarantees that the evaluation dataset accurately represents real-world scenarios and supports rigorous, repeatable analysis.

4.1.2 Metrics

A combination of automated similarity measures, a manual verification and a novel *LLM-as-Judge* approach are employed to capture both lexical overlap and semantic correctness:

- **ROUGE [17]:** Recall-Oriented Understudy for Gisting Evaluation (ROUGE) quantifies n-gram overlap between generated and reference texts. We calculate ROUGE-L (longest common subsequence) precision, recall, and F1 scores. These measures reveal the extent to which the system reproduces the reference wording.
- **BERTScore [18]:** Using contextual embeddings from a pre-trained BERT model, BERTScore computes a token-level cosine similarity matrix between generated and reference sentences. We extract precision (average maximum similarity from generated to reference tokens), recall (reference to generated), and F1. This metric captures semantic equivalence beyond exact string matches, effectively recognizing paraphrases and concept preservation.
- **LLM-as-Judge Correctness:** A language model is prompted with each generated-answer/reference pair to produce a binary correctness label (correct" or incorrect"). The model evaluates factual accuracy and completeness with respect to the gold-standard response.

- **LLM-as-Judge Coherence:** A separate LLM judge assesses each generated answer's coherence relative to the original query, independent of any reference. This evaluation ensures that the response logically addresses the question, maintains contextual relevance, and follows a clear structure.
- **Manual Evaluation:** The responses are assesed by a human who determine correctness and contextual appropriateness, capturing nuanced errors that automated metrics may overlook.

These complementary metrics provide a holistic view of system performance, balancing quantitative rigor with qualitative insight into the Graph RAG system's strengths and areas for improvement.

4.2 Use Cases

To assess the practical value of the proposed system, we defined a representative set of use cases based on common information needs in supply chain management. These use cases were designed to evaluate the model's ability to reason over heterogeneous graph-structured data and generate accurate responses. The following categories summarize the types of questions the system is expected to handle effectively:

- **Bill of Materials (BoM) Exploration:** The system is able to identify and list the raw materials required to manufacture a given product, supporting engineers and planners in understanding product composition and upstream dependencies.

- **Supplier Risk Identification:** Focus on determining whether materials are single or multi-sourced, which is critical for assessing supply vulnerability. The system can also quantify sourcing distribution percentages across suppliers for multi-sourced materials.

- **Sourcing Strategy Insights:** The system can identify whether a material is only purchased, only manufactured, or eligible for both, enabling better decisions on procurement and production flexibility.

- **Geographical Manufacturing Footprint:** It can determine in which regions a product is partially manufactured, allowing users to understand the regional distribution of production and its potential implications on lead times or regional risks.

- **Financial Impact Analysis:** The system supports the extraction of revenue figures associated with specific materials, both individually and in relation to their sourcing strategy. This provides supply chain managers with insights into financial exposure due to supplier concentration.

- **Supplier Spend Assessment:** Another set of use cases involves identifying the supplier or supplier location with the highest financial impact in the supply network of a given product, which can inform negotiation strategies or risk mitigation planning.

- **Logistics and Distribution Routing:** The system is capable of retrieving the recommended shipping routes from manufacturing plants to distribution centers, accounting for regional constraints and target destinations.

These use cases reflect a broad spectrum of decision-making scenarios in supply chain operations, including procurement, risk management, production planning, and logistics. The ability of the system to address them accurately is key to demonstrating its value in real-world industrial contexts.

4.3 Results and Analysis

Experiments were performed using multiple OpenAI models to determine the most efficient configuration in terms of inference latency, answer precision, and cost per token. For each model, the metrics explained in subsection 4.1.2 were recorded, as well as the response time.

Results are displayed in Table 4.1.

Model	Manual	Correct.	Coher.	BERT	ROUGE	Time	Cost
gpt-4.1	**0.90**	0.90	0.89	0.92	0.59	4.49	$2.00
gpt-4.1-mini	0.50	0.40	0.77	0.90	0.47	7.43	$0.40
gpt-4.1-nano	0.10	0.10	0.63	0.88	0.40	3.73	$0.10
gpt-4o	0.00	0.10	0.60	0.87	0.35	4.95	$2.50
o3	**0.90**	0.90	0.87	0.95	0.69	37.26	$2.00
o4-mini	0.60	0.60	0.90	0.93	0.59	13.86	$1.10

Table 4.1: Performance of different OpenAI models on the question answering dataset. All columns except Time and Cost are normalized between 0 and 1. Cost is measured for 1M input tokens and Time in seconds.
Manual: Manual validation score. Correct.: Correctness. Coher.: Coherence. BERT: BERTScore. ROUGE: ROUGE-L. Time: Response time in seconds. Cost: input cost in USD per million tokens.

4.3.1 Interpretation of Results

- Manual validation emerges as the most critical metric for evaluating system performance. While the LLM-based correctness score is useful, it does not always align with the manual assessment. In some cases, the correctness metric marked answers as incorrect when the model provided information using the description or name fields rather than exact material or location ids, or due to the addition of extra information.

- The best-performing models are o3 and gpt-4.1, both achieving the highest accuracy scores. However, there is a substantial difference in response time: o3 has an average latency of 37 seconds, while gpt-4.1 responds in 4.5 seconds on average. This latency difference is an important factor when selecting a model for deployment in practical, interactive systems.

- Although the price per token for the most advanced models is comparatively high (2$ per 1M tokens), overall cost efficiency depends on the total number

of tokens processed per query. Notably, models that rely on explicit chain-of-thought reasoning, such as o3, tend to consume more tokens per query than models like gpt-4.1, as they generate longer intermediate reasoning steps and verbose explanations before arriving at the final Cypher query. In the evaluation, it has been observed that o3 can consume approximately 2 times more tokens per query compared to gpt-4.1 in the same tasks. Considering an average of 1,500 tokens per query for gpt-4.1 and 3,000 tokens per query for o3. Even under these conditions, and according to official OpenAI pricing, the cost per query remain below one cent (USD 0.01) for both models. Therefore, despite higher token consumption, the overall cost per query remains manageable in practical settings, with the primary trade-offs remaining in latency and accuracy rather than in financial feasibility.

- The gpt-4.1 model failed in only one out of ten test cases, specifically with the query regarding the number of echelons or layers in the supply chain. In this case, the model only considered the echelons where the specific material was present, rather than the complete supply chain, including upstream raw materials. However, with an additional clarifying user prompt, the system was able to produce the correct response (Figure 4.1). This highlights both the limitations of automated assessment and the potential for improvement through interactive clarification.

- Regarding performance, gpt-4.1-mini exhibited significantly longer response times. This is attributable to the model entering repeated execution loops after encountering Neo4j query errors, stemming from less accurate Cypher query generation. As a result, overall latency increased compared to other models.

- It is also evident that the lighter models, such as gpt-4.1-nano, lack the capability to generate the complex Cypher queries necessary for advanced supply chain reasoning. This underlines the need for large, high-capacity models in tasks requiring structured graph query generation and domain reasoning.

- The limitations of BERTScore and ROUGE are evident in this context. For example, given the question "What is the revenue associated with material 47412?", an incorrect answer such as "I'm unable to provide the revenue associated with material 47412" can still yield a BERTScore of 0.91 and ROUGE of 0.6, simply due to overlapping words. Therefore, these metrics are not suitable for tasks requiring precise, fact-based evaluation, and are more appropriate in scenarios where exactness of key information is less critical.

- Overall, the evaluation shows that while several automated metrics provide partial insights, only manual or advanced LLM-based assessments reliably reflect the actual utility of the generated responses for supply chain question answering.

4.3.2 Example of System Performance

To exemplify the system performance differences given different LLM models, Table 4.2 shows the practical impact and how the final chosen model is capable of answering a question of the dataset, while another lighter model is not able to find the correct Cypher query.

Question	What raw materials are required to manufacture material_123136?
Expected Answer	The raw materials required are: material_9087, material_22070, material_61709, material_121994, material_51074, material_89103, material_86307, material_131685, material_133058, material_91050.
gpt-4.1 Answer	The raw materials required to manufacture material_123136 include the following: material_22070, material_9087, material_51074, material_61709, material_89103, material_86307, material_91050, material_131685, material_133058, material_121994.
gpt-4.1-nano Answer	I'm unable to determine the raw materials required for material_123136 based on the available information.

Table 4.2: Comparison between a high-performing and a low-performing model for a representative supply chain query of the evaluation dataset.

4.3.3 Cypher Query Examples for Dataset Questions

Some models have not been able to execute the task of generating the Cypher queries successfully, as Table 4.1 and Table 4.2 have shown. To illustrate the complexity inherent in the translation from natural language to Cypher queries in this dataset, some representative examples are provided below. Each query demonstrates the non-trivial reasoning, schema understanding, and logical chaining that must be performed by a generative model in order to correctly retrieve or calculate the required information.

- **Bill of Materials (BoM) Exploration**

What raw materials are required to manufacture material_123136?

To answer that example question, the following cypher statement is generated.

```
MATCH (product:MatLoc {material_id: "material_123136"})
MATCH path = (raw:MatLoc)-[:Lane*1..]->(product)
WHERE raw.material_type = "raw_material"
RETURN DISTINCT
raw.material_id AS raw_material_id
```

Complexity: Requires expanding the supply chain nodes of the product to trasverse the graph until it reaches suppliers, through multiple hops of the network. It also requires filtering by a specific field that the model must understand looking at the schema of the database.

- **Supplier Risk Identification**

  Next question of the dataset.

  Which of these materials are supplied by a single source?

```
MATCH (rm:MatLoc)
WHERE rm.material_id IN ['material_9087', 'material_22070',
    'material_61709', 'material_121994', 'material_51074',
    'material_89103','material_86307', 'material_131685',
    'material_133058', 'material_91050']
WITH rm
MATCH (supplier:MatLoc)-[:Lane]->(rm)
WHERE supplier.location_type IN ['supplier', 'vendor']
WITH rm.material_id AS material_id,
COLLECT(DISTINCT supplier.location_id) AS supplier_ids
WHERE SIZE(supplier_ids) = 1
RETURN material_id, supplier_ids[0] AS supplier_location_id
```

Complexity: It requires consulting its memory from the previous answer. It must take into account the types of locations, in this case supplier and vendor are the relevant ones. Then it must collect all the suppliers traversing the graph for the list of materials and count the number of different suppliers. Then it filters where there is just one supplier.

It is a long query that even a human could fail, but the system manages to respond correctly.

- **Logistics and Distribution Routing**

  What shipping route do I have to follow to deliver product material_47412 from its plant to a distribution center in Georgia?

```
MATCH (plant:MatLoc {material_id: 'material_47412',
    location_type: 'production plant'})
MATCH (dc:MatLoc {material_id: 'material_47412',
    location_type: 'distribution center', region: 'Georgia'})
MATCH path = shortestPath((plant)-[:Lane*]->(dc))
RETURN
    [n IN nodes(path) | {material_id: n.material_id,
    location_id: n.location_id, location_type: n.location_type,
    region: n.region}] AS shipping_route
```

  Complexity: Requires finding the origin, the possible destinies given the filter of the question, and then, calculating the shortest path between the nodes, which is a function available in the Cypher language.

These queries exemplify the advanced schema reasoning and multi-step chaining necessary for correct graph retrieval, demonstrating the difficulty and importance of robust query generation from natural language, which constrain the choice of the LLM to the most advanced ones: gpt-4.1 and o3.

4.3.4 Interactive prompting

Beyond standard evaluation, the system was also tested using an interactive, few-shot prompting approach, leveraging a custom graphical interface developed with Streamlit. In this setting, queries were issued one by one, and additional guiding questions or clarifications were provided in cases where the initial model response did not fully resolve the original query.

This interactive approach was particularly effective for complex or ambiguous questions. For example, in the single failure case encountered by gpt-4.1—concerning the calculation of echelons in the supply chain—an extra clarification prompt allowed the model to adjust its interpretation and generate the correct Cypher query. As a result, 100% accuracy was achieved after brief follow-up interactions.

This finding highlights two important points: first, that advanced language models can benefit significantly from incremental guidance in domains with intricate schemas or ambiguous requirements; and second, that real-world deployment scenarios can leverage interactive systems to achieve robust accuracy, even when initial responses are incomplete or imprecise.

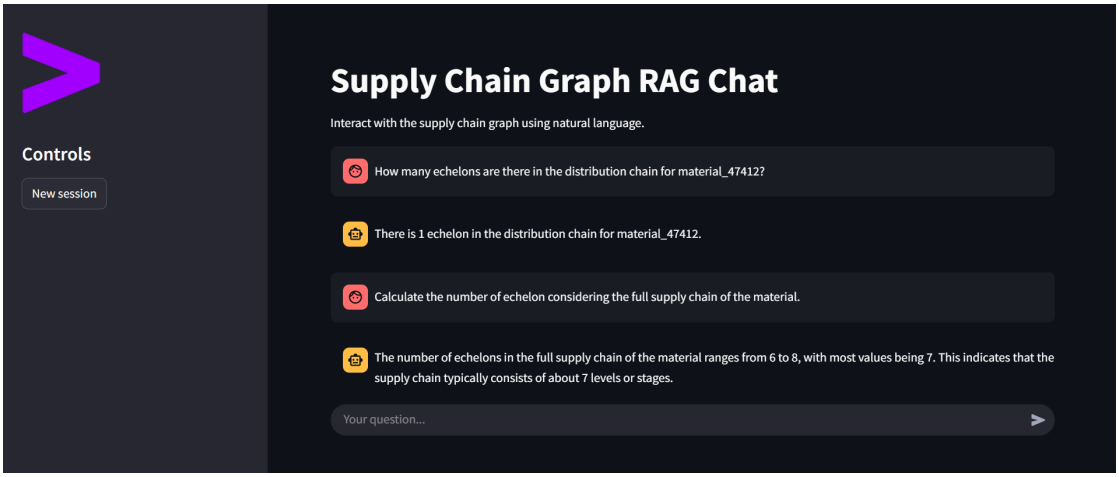

Figure 4.1: Graphical interface built with Streamlit. When initial responses are ambiguous or incorrect, targeted follow-up questions help the system converge to the correct answer. The example illustrates the scenario in which gpt-4.1 corrected its output after additional clarification.

Chapter 5

Conclusions and Future Work

This final chapter summarizes the main findings and contributions of the project, reflecting on the results obtained and the implications for both research and practical applications. In addition, it outlines promising directions for future work aimed at addressing current limitations and expanding the capabilities of the proposed system. The chapter is structured into two sections: the first presents the main conclusions derived from the development and evaluation of the system, and the second proposes a set of future research and implementation lines.

5.1 Conclusions

The results obtained in this work demonstrate the value and viability of applying a Graph Retrieval-Augmented Generation approach to the supply chain domain. The proposed system, built upon a modular and extensible architecture, leverages structured knowledge graphs to provide context-aware, accurate, and timely responses to complex supply chain queries. Several key findings and contributions can be highlighted:

- **Feasibility and Impact:** Integrating retrieval-augmented generation with structured graph data has proven not only feasible but also beneficial, given the inherently relational and interconnected nature of supply chain information.
- **Improved Efficiency and Usability:** The developed system significantly accelerates the process of information retrieval and synthesis compared to manual analysis by human experts. Its user-friendly interface allows both technical and non-technical users to access complex supply chain data, reducing the reliance on specialized knowledge for querying and interpretation.
- **Model Performance:** Comparative evaluations of different large language models indicate that state-of-the-art models (such as GPT-4.1 and O3) achieve higher accuracy in Cypher query generation. These improvements are particularly notable in handling complex, multi-hop queries that require a deep understanding of both the schema and the business context.
- **System Modularity and Extensibility:** The architecture's modular design facilitates the integration of custom knowledge graphs, retrieval strategies, and prompt templates, making it adaptable to a wide range of supply chain scenarios and supporting future expansions or adjustments with minimal disruption.
- **Economic Impact and Cost Efficiency**: From an economic perspective, the development of this solution represents a strategic investment for Accenture. The total cost of the project, estimated at approximately 5000 € over a five-month internship period (including the intern's involvement in other non related tasks) has enabled the creation of a functional prototype with immediate and long-term value. In the short term, the system can be leveraged by analysts to efficiently interact with clients' supply chain data, extracting actionable insights in less time and with less manual effort. This not only increases analyst productivity but also enhances the quality and speed of client deliverables. In the longer term, the solution has the potential to be

packaged and offered as an added-value product in future projects, enabling clients themselves to query their supply chain data in natural language and obtain reliable, interpretable answers. Moreover, the operational cost of the system is extremely low, with each query incurring an expense of less than one cent. This dual benefit positions the investment as highly cost-effective and likely to generate substantial returns for the company.

While these advances represent a significant step forward, some limitations remain, primarily related to the evolving nature of large-scale knowledge graphs and the current capabilities of generative models. For instance, the system's effectiveness depends on the quality, completeness, and stability of the underlying schema; unforeseen changes or ambiguities in the data model may affect query generation or interpretation. Although the system dynamically retrieves schema information from the Neo4j database, certain types of errors, such as misinterpretation of schema elements or rare query types, can still occur. Moreover, scaling the solution to larger datasets or more demanding operational environments may require further optimizations in resource management and error handling.

Overall, the thesis demonstrates that Graph-RAG approaches can deliver practical value in supply chain management, providing a flexible and scalable foundation as a support tool for advanced decision. The insights gained highlight both the strengths of this approach and the areas where future research and development can further enhance its robustness and impact.

5.2 Future Work

Building on the foundation established in this work, several promising directions can be pursued to enhance the performance, robustness, and applicability of Graph-RAG systems in supply chain and related domains:

- **Hybrid Retrieval Strategies:** Combining graph-based retrieval with vector-based approaches could further improve both usability and retrieval performance. In particular, vectorizing product and location names or descriptions would enable users to refer to entities more intuitively, without needing to know specific IDs. This enhancement would lower the barrier for formulating queries about specific entities, improve semantic matching, and support more natural and flexible interactions, especially as data volumes and schema complexity grow.
- **Enhanced Error Detection and Handling:** Developing more robust mechanisms for detecting, explaining, and recovering from query errors, such as ambiguous prompts, would increase system reliability and user trust. Ideally, the system would identify ambiguities in user queries and would ask the user what they are specifically referring to.
- **Human-in-the-Loop Evaluation:** Establishing evaluation pipelines that include feedback from supply chain professionals or end users will provide deeper insights into system utility and guide further refinements tailored to practical requirements.
- **Scalability and Deployment Optimization:** Try a real large supply chain dataset to check if the processing time scales properly. If not, investigating strategies for optimizing computational efficiency and managing costs, such as model distillation, query batching, or edge deployment, will be essential for scaling the solution to production environments.
- **Improved Explainability and Transparency:** Designing user-facing features that enhance the interpretability of generated queries and responses will facilitate adoption and debugging, supporting greater trust in automated decision-support systems. For instance, displaying in real time the subgraph from which the answer has been generated could simplify the verification of the response's veracity and provide users with a more intuitive, visual understanding of how the system is reasoning over the data. Leveraging Neo4j's graph visualization capabilities to present these subgraphs would help users not only to audit results but also to grasp the relationships and data paths involved in each answer, ultimately increasing transparency and trust in the system.

- **Integration of Tool Functions:** Incorporating access to specialized tools within the agent workflow. For example, to trigger stress test algorithms or other domain-specific actions for which pre-existing code is available. It would expand the system's capabilities and facilitate the integration of proprietary functionalities already developed by Accenture or other stakeholders.
- **Exploration of Alternative Language Models:** Evaluating the performance of non-OpenAI LLMs or applying fine-tuning techniques specifically for Cypher query generation could yield further improvements in accuracy, speed, and cost-effectiveness.
- **Expansion of the Evaluation Dataset:** Broadening the test dataset to encompass a wider variety of supply chain scenarios and queries would enhance the reliability and robustness of system evaluation, providing a stronger empirical basis for assessing its real-world utility.
- **User Interface Enhancement:** Further development of the user interface beyond the current prototype stage would improve overall user experience, making the system more suitable for operational deployment and adoption by non-technical users.

Bibliography

[1] Ted Schrecker. "Globalization and economies". In: *The Intersection of Global Health and Sustainable Development*. Edward Elgar Publishing, 2025, pp. 265–280. URL: https://www.elgaronline.com/edcollchap/book/9781800885981/book-part-9781800885981-17.xml.

[2] Yushan Liu et al. "A knowledge graph perspective on supply chain resilience". In: *arXiv preprint arXiv:2305.08506* (2023).

[3] Ryan Chandler. *NLP at Scale for Maintenance and Supply Chain Management*. Accessed May 27, 2025. 2018. URL: https://neo4j.com/blog/supply-chain-and-logistics/nlp-at-scale-maintenance-supply-chain-management/.

[4] Alexander R Pelletier et al. "Explainable biomedical hypothesis generation via retrieval augmented generation enabled large language models". In: *arXiv preprint arXiv:2407.12888* (2024).

[5] Karin Küblböck. *The EU raw materials initiative: scope and critical assessment*. Tech. rep. ÖFSE Briefing Paper, 2013.

[6] Ge Zheng and Alexandra Brintrup. "Enhancing Supply Chain Visibility with Generative AI: An Exploratory Case Study on Relationship Prediction in Knowledge Graphs". In: *arXiv preprint arXiv:2412.03390* (2024).

[7] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[8] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*. 2019, pp. 4171–4186.

[9] Tom Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[10] Patrick Lewis et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks". In: *Advances in neural information processing systems* 33 (2020), pp. 9459–9474.

[11] Tyler Thomas Procko and Omar Ochoa. "Graph Retrieval-Augmented Generation for Large Language Models: A Survey". In: *2024 Conference on AI, Science, Engineering, and Technology (AIxSET)*. 2024, pp. 166–169. DOI: 10.1109/AIxSET62544.2024.00030.

[12] Ian Robinson, Jim Webber, and Emil Eifrem. *Graph databases: new opportunities for connected data.* " O'Reilly Media, Inc.", 2015.

[13] Young-Chae Hong and Jing Chen. "Graph database to enhance supply chain resilience for industry 4.0". In: *International Journal of Information Systems and Supply Chain Management (IJISSCM)* 15.1 (2022), pp. 1–19.

[14] Darren Edge et al. "From local to global: A graph rag approach to query-focused summarization". In: *arXiv preprint arXiv:2404.16130* (2024).

[15] Neo4j, Inc. *Neo4j Data Importer*. Online; accessed June 14, 2025. Neo4j, 2025. URL: https://data-importer.neo4j.io/.

[16] Streamlit, Inc. *Build a basic LLM chat app*. Online; accessed June 17, 2025. Streamlit, Inc., 2025. URL: https://docs.streamlit.io/develop/tutorials/chat-and-llm-apps/build-conversational-apps.

[17] Chin-Yew Lin. "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: https://aclanthology.org/W04-1013/.

[18] Tianyi Zhang et al. "BERTScore: Evaluating Text Generation with BERT". In: *International Conference on Learning Representations (ICLR)*. 2020. URL: https://openreview.net/forum?id=SkeHuCVFDr.