A Comparison of Relational and Graph Databases for Supply Chain Data: a Qualitative and Quantitative Evaluation

SUBMITTED IN PARTIAL FULFILLMENT FOR THE DEGREE OF MASTER OF SCIENCE

Teun van Schagen
12405574

Master Information Studies
Information Systems

Faculty of Science
University of Amsterdam

June 28, 2019

1st supervisor
prof. dr. P. T. Groth
Faculty of Science
Informatics Institute
Universiteit van Amsterdam

2nd supervisor
MSc. Caspar Honée
CTO SIM B.V.
1812RL 18B, Alkmaar
The Netherlands

A comparison of relational and graph databases for supply chain data: a qualitative and quantitative evaluation

T. van Schagen
teun.vanschagen@student.uva.nl
University of Amsterdam
Amsterdam, the Netherlands

ABSTRACT

Data in supply chains is growing due to the trends of real-time product traceability and a higher need for data analysis due to competition. This research wants to evaluate whether graph databases are an effective method to deal with this expanding data in real-time systems. An evaluation was done on the use of graph database systems in supply chains through experimental performance- and usability analysis. 7 different datasets were benchmarked between two databases, and a tool was developed to generate and benchmark realistic structures and data in supply chains. The graph database was shown to perform 3-5 times faster in queries where a large number of traversals were done, where the relational database was up to 2 times faster in low-traversal filter queries.

CCS CONCEPTS

• **Information systems → Relational database model**; **Graph-based database models**.

KEYWORDS

relational and graph comparison, supply chain management, performance analysis, database benchmark, query complexity measurement

1 INTRODUCTION

Supply Chain Management (SCM) has become increasingly more important as businesses started competing on the performance of their supply chains [13]. Supply chain managers are using data analysis and rely on data to optimize the performance of their supply chains to more effectively compete amongst other supply chains [13]. According to Waller et al. [34] data science, big data, and predictive analytics have the possibility to change the approach to SCM in terms of design, relationship development, customer service, and day-to-day operations.

Another closely related aspect and trend in supply chains is that of transparency and traceability. Consumers demand verifiable evidence of how their products are made in food supply chains (transparency) [25] and new methods are proposed to actively track products and batches of products in supply chains through RFID tags (traceability) [16].

Increased interest in data analysis and real-time product tracking combined make for a case in which the amount of data and the complexity of questions is increasing. We need to take care of how these larger growing volumes of data in supply chains can be efficiently stored and queried in real-time systems.

This study compares relational databases to graph databases. Graph theory has been applied to the field of supply chain management before [12, 31, 32], but a comparison of different ways to store supply chain data has not been done. In this paper, we propose to compare storing and querying the data in a relational table structure versus a graph structure through a practical experiment. For this experiment, we compare these two databases and also develop an expandable data generator and benchmark that captures the unique properties of supply chain data, and reflects real-world supply chain data and queries. To the best of our knowledge, such a data generator and benchmark currently do not exist. The contributions of this paper are the open source supply chain data generator and benchmark, the list of queries we use to benchmark performance and usability, and the results from this benchmark in a number of different scenarios. Specifically, we tackle the following research question.

> To what extent is a graph database more effective for storing, manipulating and retrieving data than a relational database for Online Transaction Processing workloads in Supply Chain Information Systems?

To answer this research question, it is split up into four sub-questions.

RQ1 What data is typically stored, and what are typical database workloads in Supply Chain Information Systems?
RQ2 What are the most important the software quality attributes to judge databases on in the context of supply chain management?
RQ3 How can data stored in Supply Chain Information Systems be captured in a relational- and graph model to form the basis for a fair comparison?
RQ4 How does a graph database compare to a relational database?

This paper is structured as follows. Section 2 describes the preceding and related work relevant to this research. Section 3 lays out how the research question is answered and details the case study. Section 4 characterizes the data and workloads for information systems in supply chains, what quality attributes should be used to compare databases, how supply chain data can be modelled, how graph- and relational databases compare for supply chain data and discusses limitations and possible criticism. Section 5 concludes with answering the main research question, summarizing what this paper has contributed, and details ways in which the research could be followed up.

2 RELATED WORK

This section lays out what work has preceded this paper through describing practices and theory in supply chain data management, the types of database that are compared, the practices of database benchmarking, and the measuring of query complexity for declarative languages such as SQL.

2.1 Supply Chain Data Management

While data management is crucial in supply chains, there is no concise definition for supply chain information systems or certainty about what data they should store.

2.1.1 Interoperability. As mentioned in the introduction, data management and analysis is crucial for competition among supply chains. Denolf et al. [7] have defined effective communication between supply chain partners and information standards for exchange between partners as critical success factors for implementing SCIS. Choosing and accepting those standards can be difficult because of compatibility with many different information systems within organizations. Research has been done into the interfirm operability in Supply Chain Information Systems (SCIS) [17]. A number of ontologies have been developed to aid interoperability but have not succeeded so far [10]. They state that we are a long way off of making use of these ontologies to solve interoperability problems that currently exist. Current ontologies only operate on the strategic level, are too distant from real-world supply chains, and have a limited view on the scope of supply chains by focusing on the inter-business network. Other gaps include the missing of the time dimension, no account of material traceability, no account of information and secondary material flows, and the fact that different organizations may look at their supply chain in a conceptually different way. However, this research is a number of years old, and thus we can not definitevely say that there are no successful implementations of ontologies in SCM.

2.2 Types of databases

A number of different databases exist, and they are fundamentally different. Where relational databases (sometimes called SQL databases) are mature and widespread in use, graph databases (sometimes considered as part of the NoSQL movement) could be considered less mature as they lack features common in relational databases.

2.2.1 CAP. On a high level, databases can be categorized into 2 distinct categories: *Online Transaction Processing* (OLTP)[1], and *Online Analytical Processing* (OLAP)[2]. Where OLTP databases focus on day-to-day systems and transactional processing, OLAP databases main focus is to process large amounts of data for analytical purposes.

A central theory in databases is the CAP-theorem. This theorem says only two out of three characteristics of CAP-theorem can be met for networked shared-data systems. It consists of:

- having a consistent set of data (consistency);
- the ability for a high level of availability (availability);
- allowance to have copies of the data in more than one place, over a network (partitioned).

According to Brewer [3] the general consensus is that partitioning is of necessity in distributed systems, and thus a choice must be made between availability and consistency. Where the NoSQL movement can be seen as favoring availability over consistency (favoring the BASE philosophy), where earlier databases favored the ACID philosophy. This paper explores two types of databases: the relational database, and the graph database.

2.2.2 Relational databases. Relational databases are based on the relational model [5]. Relational databases are mature systems that were some of the earliest systems deployed over networks [14]. Data in relational databases is stored in tables. Generally, the table represents an entity, columns in the table represent the properties of the entity, and rows in the table represent the instances of the entity. Relational databases often designed through the use of an ER (*Entity Relation*)-schema, which is then converted through a process called *normalization* to arrive at the tables that are to be used in the database system. Relations between entities are achieved through primary and foreign keys, where a column is added to a table that connects it to another table. Most relational databases work with *Structured Query Language* (SQL) alter the database (*Data Definition Language*) or alter the data in the database (*Data Manipulation Language*). SQL is a declarative language, which means the only the intended output is asked, and the task of how to achieve this output is left to the database [14].

2.2.3 Graph databases. Graph databases are based on concepts from graph theory. According to Angles & Gutierrez [1] graph database models differ from relational databases through their basic data structure, their abstraction level, and their information focus. Graph databases focus on storing relations and data, where relational databases focus on storing data and attributes. A strength of graph databases is that they can explore the paths, neighborhoods, and patterns where relational databases through SQL, lack this ability natively. Another difference of the graph database model is that the schema is not defined beforehand, but can be altered on the fly. The need for graph databases comes from a number of origins: for classical applications such as transport networks, for the need of more expressive query languages, and using complex networks such as social networks, information flow (provenance), and technological networks such as telecommunication networks [1]. As there are different types of graphs in graph theory, different graph database models exist. RDF and labeled property graphs are two well-known examples. Graph databases are currently not as mature as relational databases, as they lack in terms of for example integrity constraints [27]. For a comprehensive overview of graph databases, and specifically Neo4j, we refer to 'Graph Database Management Systems' [9] and 'Graph Database Applications and Concepts with Neo4j' [24].

2.3 Database quality attributes

Software quality attributes are common in software development, as there are a number of standards defined by different organizations. Attributes for databases specifically are not as common.

Chen et al. [4] have conducted research through grounded theory to determine what they think are the most important quality attributes for software systems. There are several standards or models defined by the International Standards for Organizations (ISO), namely ISO/IEC 9126:1999 (replaced by ISO/IEC 25010:2011), and ISO/IEC 25040:2011. IEEE also has its own set of standards, namely IEEE 730-2014. Miguel et al. [22] have gathered the major standards

[1]OLTP https://en.wikipedia.org/wiki/Online_transaction_processing
[2]OLAP https://en.wikipedia.org/wiki/Online_analytical_processing

and models for a quality attribute based evaluation of software products. Lourenco et al. [18] define a number of quality attributes for databases but provide no methodology as to how they were gathered or why exactly they are important. They state that the chosen QA's are based on 'use of these attributes in every software project' and 'are intimately tied with the topic of databases' or 'are extensively covered in literature'.

2.4 Database benchmarking

There currently, to the best of our knowledge, does not exist a benchmarking tool for testing performance of supply chain data in different kinds and implementations of databases. Supply chain data is, however, an interesting area to develop a benchmark for, because of its unique nature, as shown later in this paper. Comparisons between graph- and relational databases have been done before, but it is not clear whether supply chain favors one over the other.

2.4.1 Previous comparisons and existing benchmarks. Many benchmarks for databases and more specifically graph benchmarks on databases exist. Examples are the LDBC Graphalytics, and the LDBC Social Network Benchmarks[3]. The Graphalytics benchmark focuses on graph-specific algorithms and serves for comparing graph databases on these algorithms for different data sizes and configurations. The Social Network Benchmark has different types of workloads, including the interactive (OLTP) workload, which could be considered similar to this paper. Social Network data differs fundamentally from supply chain data, which will be explored in this paper.

Vickair et al. [33] have done a comparison between relational and graph databases for provenance. They generated 3 types of datasets: integer, 8K char, and 32K char data in sizes of 1000, 5000, 10000, and 100000 nodes and or rows. In graph traversal, Neo4j was significantly faster for retrieving nodes with a depth of 128, while for a depth of 4, performance was generally comparable. For data queries (queries that filter integer data) MySQL was significantly faster. The authors attribute this last phenomenon to the fact that Neo4j treats all its data as text through its Lucene indexing engine.

In another comparison between a number of open-source graph databases and SQLite (a relational database), the latter performs among the best scorers in some benchmarks (PageRank and shortest path) but among the lowest scorers in connected components and the updates benchmarks [19]. The authors, however, do not give an explanation of the relative performance of the relational database compared to the graph databases. In the context of supply chain data. Favoring one implementation over another could depend on the level of graph-like queries, the number of traversals, data types and other factors.

2.5 Measuring query complexity

There are currently, to the best of our knowledge, no formal metrics of complexity for declarative programming languages such as SQL and Cypher. There is however a metric of software complexity that can be applied to these languages: Halstead complexity.

2.5.1 Halstead Complexity. Halstead complexity is a number of measures that include the difficulty of a software program, but also the effort to program, and estimated number of bugs delivered. In order to calculate the Halstead complexity, 4 metrics are necessary: the number of distinct operators (η_1), the number of distinct operands (η_2), the total number of operators (N_1), and total the number of operands (N_2). We can then calculate the *difficulty* measure as follows.

$$D = \frac{\eta_1}{2} \times \frac{N_2}{\eta_2}$$

3 RESEARCH METHODOLOGY

This research comes in the form of a comparative study. The comparison is between the relational- and graph database models, for supply chain data. To this end, a comparison is done through a set of experiments on two exemplary databases; MySQL and Neo4j. To form a fair comparison, the database models are constructed from scratch, and the instruments used to perform the experiments are validated in the case study. We describe our methodology to tackle each sub-research question below.

3.1 Characterizing data and workloads

It is necessary to characterize data stored and typical workloads in Supply Chain Management (SCM) information systems in order to i) create representative data models, ii) perform an experiment with realistic dummy data, and iii) perform the experiment with realistic workloads.. The space of SCM information systems is shortly explored to view how information systems are used in the field of SCM. Frameworks and ontologies for supply chains are examined to find out what entities and properties are most often used to describe data in supply chains. Typical workloads are characterized using a case study of an SCM information system called *SIM Solutions*[4]. To this end, a set of real-world queries are compiled based on literature and the case study and are then verified with domain experts from the organization.

3.2 Setting up quality attributes

To compare graph databases with relational databases, it is necessary to set up some sort of framework to be able to compare them to reason about their qualities. This is done through software *Quality Attributes* (QA's). A literature study is carried out to decide what kinds of quality attribute models for comparing software and more specifically databases are available. QA's appearing in many models will be evaluated for use in this paper. It is argued why they are inherent to the underlying data model used in a database, and why they are important for supply chain data in particular.

3.3 Modelling supply chain data

A short literature review is carried out to capture the most common best practice modelling approaches for graph and relational database models. These are then applied to the data that is to be captured in a graph, and the resulting data models are described. This is necessary to form a fair comparison between the two databases, to make the best use of their specific implementations. We chose not to take the existing data model from the case study because

[3]LDBC benchmarks http://ldbcouncil.org/benchmarks

[4]SIM B.V. https://www.simsupplychain.com/

this would introduce an unfair bias to the graph database as the development of the existing data model was not under our control.

3.4 Comparing graph and relational databases

Finally, the actual comparison needs to be done. Qualitative reasoning is accomplished with the quality attributes defined in *RQ2* accompanied by literature research. An experiment is carried out in which the models are realized and compared through performance benchmarking and usability testing in Neo4j[5] and MySQL[6]. With the list of queries compiled in *RQ1* query complexity is measured and through the data generator and benchmarking tool the resource- and time efficiency is evaluated. Neo4j and MySQL are the databases of choice because they are both `ACID`[7] compliant transaction-focused databases, they are the most popular[8] databases of their kind and because Neo4j and MySQL represent the respective underlying native models of the graph and relational table structures.

4 RESULTS

4.1 Characterizing data and workloads

To judge performance of the graph- and relational database in comparison, a realistic data model, structure, and workloads are necessary to say something about the applicability in real-world applications. To this end, we describe the field of *Supply Chain Information Systems* (SCIS), identify the most important activities, and compile a list of queries that form the workloads for SCIS.

As perhaps the most broadly recognized definition for Supply Chain Management (SCM) Mentzer et al. [21] states that for SCM, there is a multitude of different definitions in literature. They define that SCM, in general, can be characterized into three categories: (i) a management philosophy, (ii) implementation of a management philosophy, and (iii) a set of management processes. Stock & Boyer [30] use their definition as one of 173 to develop an encompassing definition of SCM as follows.

> “The management of a network of relationships within a firm and between interdependent organizations and business units consisting of material suppliers, purchasing, production facilities, logistics, marketing, and related systems that facilitate the forward and reverse flow of materials, services, finances and information from the original producer to final customer with the benefits of adding value, maximizing profitability through efficiencies, and achieving customer satisfaction”

This definition also relates to information systems, as systems and information are explicit parts of this definition. The term *Supply Chain Information Systems* (SCIS) is used in literature [8, 11, 17, 20], but a definition is not apparent. SCM and SCIS have close relations to *Product Lifecycle Management* (PLM), *Enterprise Resource Planning* (ERP), *Supplier Relationship Management* (SRM), and *Customer Relationship Management* (CRM) [26, 28]. These together are sometimes called *Enterprise Systems*.

A good candidate to find out what information is crucial for and in SCM and SCIS is the *Supply Chain Operations Reference* (SCOR) framework. The SCOR model consists of six processes: plan, source, make, deliver, return, and enable. Important is thus: product and supply chain development (plan); searching and selecting suppliers (source); manufacturing activities (make); the flow of materials, forwards as well as backwards, (deliver and return); and everything necessary to make these processes run effectively and efficiently, such as network management, contract management, and compliance and risk management (enable).

4.1.1 Characterizing workloads. A List of queries has been compiled based on the analysis in the previous section and then verified with domain experts from SIM B.V. They could be considered to mainly operate on the enable process in relation to SCOR, as they map the supply chain for their customers and check their compliance to the standards that their client has set for their supply chain. The list is displayed in table 1. The queries were set up in a way that follows a number of real-world scenarios that can occur in SCIS.

4.2 Quality attributes for databases

In order to judge the use of graph- and relational databases for supply chain data, a number of aspects should be selected to compare them on. As no frameworks or standards exist for data storage in supply chains in particular, we base our aspects on software quality attribute- and SCIS literature. As the goal in this paper is to compare the databases as much as possible independent from the vendor and specific implementation, aspects chosen should not rely fully on specific implementations of the graph- and relational models. In the following sections, the chosen aspects are detailed and argued.

4.2.1 Efficiency. Efficiency of software in combination with the hardware determines the perceived performance of an information system. As databases are often tasked with persisting data in a consistent manner, the efficiency of the database system is one of its most important aspects [29]. We can split efficiency up into two dimensions; time efficiency and resource efficiency.

- Time efficiency: indicates how quick a response is received from the system after a query
- Resource efficiency: indicates how efficient the system deals with its available resources

4.2.2 Usability. Usability is the “degree to which the software product makes it easy for users to operate and control it” [23]. Usability can aid the efficiency of system development and maintenance when the system meets the needs and goals of its users. When the system is flexible and maintainable, it can also help with system interoperability, which is considered as a critical success factor by Denolf et al. [7]. As usability can mean a number of things, we split it up into three parts.

- Ease of use: indicates how easy the system is able to fulfill the desired goal of a user
- Maintainability: indicates in what manner the system aids changes made over time
- Flexibility: indicates the ability of the system to deal with changes

[5]Neo4j https://neo4j.com/
[6]MySQL https://www.mysql.com/
[7]ACID https://en.wikipedia.org/wiki/ACID_(computer_science)
[8]According to DB Engines Ranking https://db-engines.com/en/ranking

#	Query
Q1	For all the products in one supply chain, retrieve all underlying products
Q2	Q1 with a maximum number of relations of 4
Q3	For all the products in one supply chain, retrieve the products, organizations, and locations that use product *x* anywhere in the chain
Q4	Retrieve all the underlying organizations and how far they are removed in number of relations
Q5	For all products in one supply chain, check whether their number of employees are under 1000, over 5000, or in between, and label them by business type as small, regular, or large changes
Q6	For every product in the whole supply chain, check whether they match a number of criteria
Q7	For every product and organization in the whole supply chain, check whether a keyword appears in their description
Q8	For all organizations in the whole supply chain, give me those that match an area of the world (long/lat), their products, activities, and location
Q9	Delete all the underlying relations for an organization with a specific activity
Q10	Modify the supply chain for a certain product, by adding a new product and supplier in between them

Table 1: List of queries capturing typical workloads in supply chain information systems.

4.2.3 Security. Security is the ability of the system to prevent malicious use through access, modification, destruction, or disclosure [23]. Security is another critical success factor as determined by Denolf et al. [8], as actors in supply chains wish not to share information with their competitors. Reliability is also explicitly mentioned by Denolf et al. as outages can prevent successful implementations of SCIS. We split up the security attribute into three aspects, borrowed from the CIA-triad.

- Confidentiality: indicates the ability of the system to prevent unauthorized access
- Integrity: indicates the ability to maintain the accuracy and completeness of the information in the system
- Availability: indicates the ability to provide access to the information when requested

4.3 Modelling supply chain data

If we want to make a fair comparison for the databases we are comparing, we need to make sure to model them in such a way that suits their strengths. We also need to make sure that the model is flexible enough to answer realistic queries. To this end, everything that is supplied by some actor in the supply chain is modelled as a product. This means that every end-user (a supply chain actor) could ask the same questions about their supply chain(s). We consider ingredients or parts of products, as well as raw materials as products. The basis of the model is thus a product-hierarchy. This product hierarchy represents the material flow between actors in the supply chain as well. Activities, organizations, and the locations are then added to this product hierarchy. This also had the added benefit of preventing cyclical relationships from existing, which could be a problem for models without relationship directionality such as relational databases in general. We describe the implementation of these principles in the following sections.

4.3.1 Relational model. There are different approaches to modelling graph-like and hierarchical data in a relational model. The product hierarchy defined earlier is essentially a tree structure. Karwin [15] defines five options for storing hierarchical data in relational databases: the adjacency list, recursive query, path enumeration, nested sets, and closure table. Karwin rates the closure table and the adjacency list as the overall best approach. We choose to model the standard approach, as this will give the best estimation of real-world scenario's in which hierarchical data is stored. The `supplier` (or organization), `activity`, `product`, and `location` entities are defined, and `supplies` and `consists_of` tables are created in order to store multiple-to-multiple relationships. Some relational databases include graph capabilities, but these are outside of the scope of this paper. The logical relational model can be seen in figure 1.

Physical model. In the physical model, primary- and foreign keys are added. These keys are indexed and saved by the database engine in a b-tree data structure allowing for fast lookup; an essential part of efficiently querying the database [14].

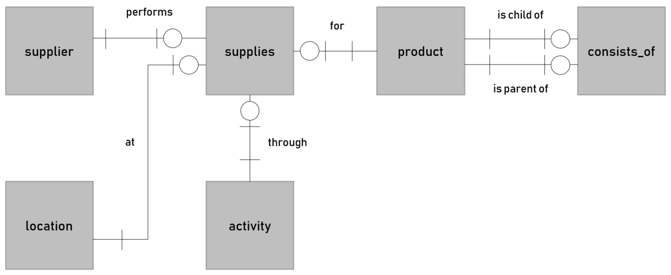

Figure 1: Relational database model of a supply chain, described using the crow's foot notation. Squares are entities, and lines are relationships. A circle means optional, a pipe symbol means mandatory, a straight line means a cardinality of one, a line with split ends means a cardinality of more than one.

4.3.2 Graph Model. Virgilio et al. [6] describe a methodology for modelling graph databases. They stress that graph databases should be designed so that data in queries are kept together in the graph as much as possible, which aids query performance. The final graph model is derived from the same principles as the relational model; the entities are `supplier` (or organization), `activity`, `location`, and `product`. Compared to the relational model, there is no need for multiple-to-multiple relationship tables as relationships are explicitly stored, and can have attributes. The logical graph model can be seen in figure 2.

Physical model. While constraints and indexes are also available to use in graph databases, most databases do not have as rigorous support for integrity constraints as relational databases [27]. As there are no foreign or primary keys, no indexes are created.

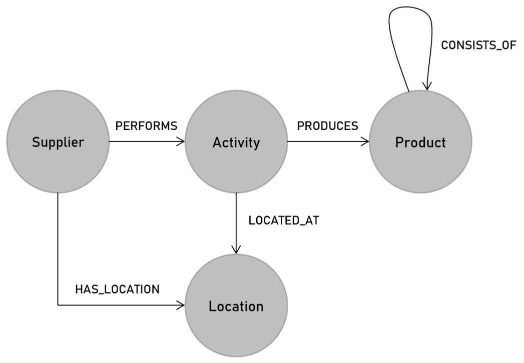

Figure 2: Graph database model of a supply chain. Circles are entities (nodes), arrows are directed relationships (edges) and the text is the label for the relationship.

4.4 Comparing graph- and relational databases

To measure the performance on the quality attributes defined before, a systematic way of evaluating this performance is deemed necessary. Therefore we implement our data model in Neo4j and MySQL and translate the queries defined into Cypher and SQL code. A small excerpt of queries can be found in the appendix, and all queries are made available on the `scm-ddg` repository.

4.4.1 Experimental set-up. A supply chain data generating- and benchmarking tool is created to generate realistic data for a number of different scenario's, and evaluate performance after. The tool is called `scm-ddg` and is made available on Github[9]. The data generation process is visually described in figure 6. The tool works as follows. First, the user is asked to set a number of parameters for the data generation process: the *number of top-level organizations*, the *number of top-level products* each, the *depth* of the chain, the *breadth* of the chain (i.e. in how many products is each product split up in the subsequent depth level), the *number of activities*, and the *total number of organizations* in the chain. All static entities (activities, organizations, and locations) are generated first. For each product of each top-level organization, a full product tree is then generated, which creates the products. Finally, the static entities are systematically assigned to the products. All entities have a number of realistic data fields which are generated randomly with a seed value, ensuring the same 'randomness' to make sure the there is no bias between different parameter configurations and different databases. To illustrate the output, a small part of the graph can be seen in figure 3. Experiments were run in identical virtual machines: hardware and software configurations can be found in the appendix.

[9]scm-ddg repository https://github.com/TvanSchagen/scm-ddg

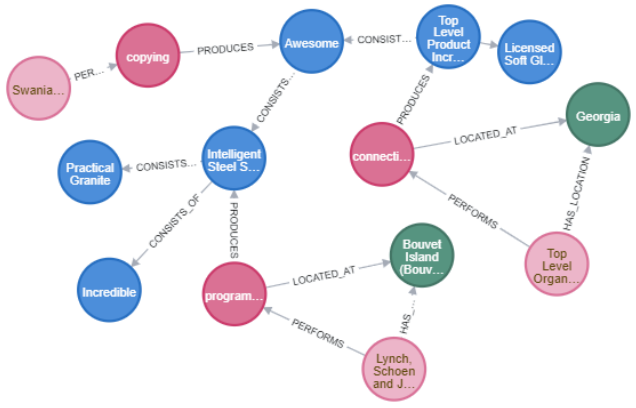

Figure 3: A small number of nodes and edges generated by the `scm-ddg` tool. Captured in Neo4j Browser.

We made sure to give both instances enough memory to fully cache the graph- and relational databases.

4.4.2 Efficiency. In the first part of the experiment, we measure the time efficiency and resource efficiency for each query in a number of different configurations. The configurations can be found in table 2. We chose for a configuration with varying depth, because this expands the most central influential entity (product), and because it gives a clear view of the performance of the queries for exponentially (because we chose *breadth* = 2) expanding data. The number of top-level organizations and the number of products per top-level organization was kept low in order to make the generation and benchmarking process manageable on lower-end hardware.

Measuring efficiency. Time efficiency is measured with the response time of a certain query in milliseconds. We use the default response times from each database vendor, reported in the *Performance Schema* in MySQL, and directly from the API in Neo4j. For resource efficiency, we measure the total database size (excluding transaction logs) which gives an estimation as to how much storage space and memory is needed. For response time, we execute each query 10 times, and remove the fastest and slowest times as outliers, and then average them out. The results for time- and resource efficiency can be seen in table 3 and 4. In figure 4, we can see that results are similar between the two databases for queries Q1 and Q2. These queries concern only traversing a single type of entity (product). When multiple entities are introduced (Q2-Q5), Neo4j is the best performer by a factor between 3 and 5. The difference between the two databases for query Q4 is visualized in 5. The biggest difference is in the full-text search of Q7, where Neo4j performs better with a factor of roughly 100. In the filtering on a single type of entity with floating point values (Q8), MySQL outperforms Neo4j by a factor of 2. The physical data size on disk is similar between the two databases, as can be seen in table 4.

4.4.3 Usability. Usability of graph databases against relational databases has not been extensively researched to our knowledge. Vickair et al. [33] does note that while highly relational data is easy to query in graph database systems against relational systems,

Depth	2, 4, 6, 8, 10, 12, 14
Breadth	2
Top Level Organizations	3
Products Per Top Level Organization	3
Number of Organizations	100
Number of Activities	100

Table 2: Configurations used for the scm-ddg tool data generation.

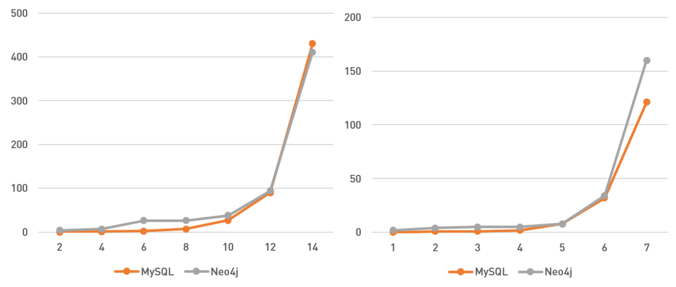

Figure 4: Response times for Query #1 (left) and 2 (right), with depth as a variable factor from 2-14.

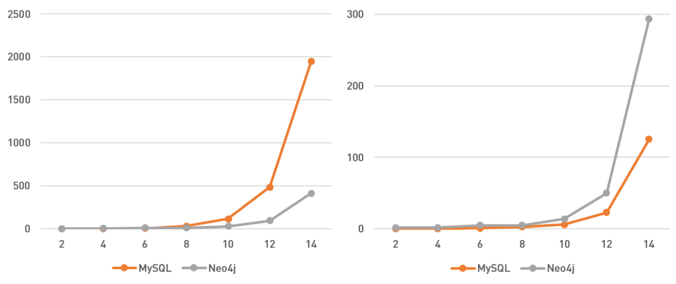

Figure 5: Response times for Query #4 (left) and 8 (right), with depth as a variable factor from 2-14.

it is realizable in relational databases. They say the opposite is not true however, and that classical relational structures are hard to query in graphs. This, however, isn't detailed or argued. We measure the usability through the measure of software complexity measures, more specifically the *Halstead* complexity measure. For each query we defined, the complexity is measured with the formula as described in §2.5.1. In table 5, the Halstead complexity measures are shown for each query. The Cypher queries (on the Neo4j graph database) are simpler by a factor between 2 and 5 compared to the SQL queries (on the MySQL database). The (absolute) difference is notably big on Q10, where a part of a graph is modified to include new products and a new supplier. Not all usability can be captured in just the query complexity measure of the queries we defined. The Neo4j database schema is more flexible in terms of possibilities because of the model used: a labeled property graph. Each node can have its own properties and labels, unlike a relational database.

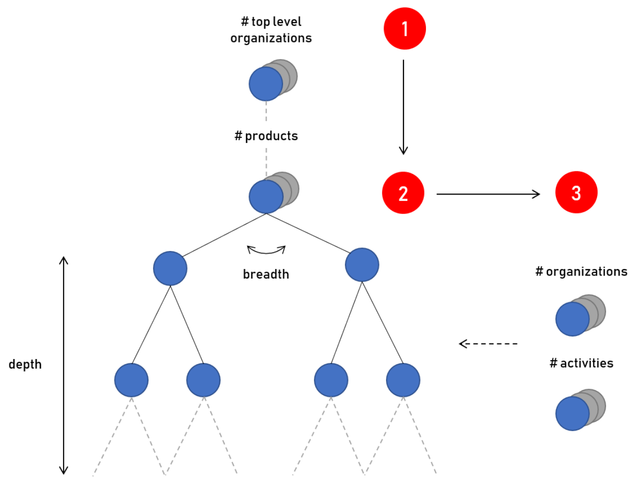

Figure 6: Data generating process used in the `scm-ddg` tool. Textual labels are parameters that can be determined at runtime. First, the entities are generated, second, the product tree is generated, and finally, the organizations and activities are assigned to the products.

The downside is that maintenance could become harder, as there are fewer options to enforce how and where information is stored.

4.4.4 Security. While security might not always be directly related to the underlying data structure used to store information, databases offer different ways of ensuring the confidentiality, integrity, and availability of the data they store. We do not perform any experiments to prove whether graph databases are generally more secure than relational databases, but as the literature review pointed out, security is considered a priority in SCIS. Graph databases can currently be considered less secure, because of the fact that they are an emerging technology, compared to the relational database [2]. There are also fewer options available for integrity constraints in databases, as demonstrated in the literature review in §2.2.3. Both Neo4j and MySQL offer clustering as an option to enable high availability, both allow users to set role-based security, and offer integrations with security providers to restrict access to unwanted users. These aspects are however dependent on the specific implementation of the database, and not the underlying storage model.

4.5 Discussion

From the results, we see that Neo4j has a distinct advantage when performing queries that traverse a high number of nodes that are close to each other. This is what we would expect from a database tailored highly interconnected data. For the raw performance of joins versus graph traversals (Q1 and Q2), MySQL seems to have a slight advantage, but when more entities are introduced to a query (Q3-5), Neo4j scales better when chain depth is increased. When traversal is limited and more filters are applied (in Q6) MySQL outperforms Neo4j. In Q8, a filter-only query without any product hierarchy traversal, MySQL also outperforms Neo4j (by roughly a factor of 2). The biggest difference, however, can be seen in full-text

Depth	MySQL							Neo4j						
	d = 2	*d* = 4	*d* = 6	*d* = 8	*d* = 10	*d* = 12	*d* = 14	*d* = 2	*d* = 4	*d* = 6	*d* = 8	*d* = 10	*d* = 12	*d* = 14
Q1 (ms)	0	1	2	7	27	90	430	4	7	26	26	38	94	411
Q2 (ms)	0	1	1	2	8	31	121	2	4	5	5	8	34	160
Q3 (ms)	1	2	5	22	86	375	1638	5	4	17	17	32	97	392
Q4 (ms)	1	2	6	31	116	483	1947	2	4	12	12	28	94	414
Q5 (ms)	1	2	7	35	124	496	2127	4	8	34	34	96	640	717
Q6 (ms)	1	3	3	15	28	67	358	7	12	38	38	127	512	209
Q7 (ms)	0	1	136	587	2172	9198	38946	1	1	7	7	24	86	371
Q8 (ms)	0	0	1	3	6	23	125	2	2	5	5	14	50	293

Table 3: Response times in milliseconds for each query and each generated dataset.

Depth	*d* = 2	*d* = 4	*d* = 6	*d* = 8	*d* = 10	*d* = 12	*d* = 14
Number of Products	63	279	1143	4599	18423	73719	294903
Size MySQL (MB)	0,3	0,5	1,9	3,6	14,5	54,8	191,9
Size Neo4j (MB)	0,3	0,4	1,2	4	15,1	59,7	238

Table 4: Number of products generated per depth level and the physical size of the database on disk.

	SQL	Cypher
Q1	6,7	1,7
Q2	6,5	1,7
Q3	8,7	1,7
Q4	9,4	3,4
Q5	12,7	4,7
Q6	9,5	2,3
Q7	4,6	2,2
Q8	3,3	1,6
Q9	6,0	1,8
Q10	16,3	3,6

Table 5: The Halstead complexity measures for each query in SQL (MySQL) and Cypher (Neo4j).

search (Q7). This is likely due to the Neo4j indexing engine, which treats all its data as text. That could also be the reason why MySQL performs better on floating point data, as shown in Q8. This last fact is in line with the results seen earlier in comparisons between Neo4j and MySQL.

As for the complexity results, we can see that Cypher queries have a lower complexity overall. This could mainly be because in SQL a *Recursive Common Table Expression* is used to query hierarchical data. This method has a lot of duplication of operators and operands to traverse the hierarchy. Differences are smaller when no traversals are done (Q7, Q8).

For our results in terms of security, we decided to mention it because it was mentioned multiple times in literature as an implementation barrier. Even though we did not do any tests, it is important to keep the security aspect in mind when selecting a database for use in SCIS.

We can not say with certainty that these results will actually translate into real-world SCIS, as their data models and structures will most likely be more complex than the ones we developed. However, the data models presented in this paper were made as generic as possible, fitting to different types and scenarios in supply chains, which aids the scalability. One could also argue that the data sizes in this paper are not very representative of real world datasets, and for application of large scale data analysis. But that was not the point of this paper. We evaluated graph databases for OLTP use, and tried to simulate larger datasets by increasing the depth up to 14, and focusing on querying the supply chain as a chain, through the product hierarchy.

5 CONCLUSION

The main contributions of this paper are the supply chain data generator- and benchmark, along with a data model and list of real-world queries on this model. These were developed to answer the main research question of this paper. We found out that data in supply chains mainly focus on the flow of materials. Therefore the model developed to test against was based on the fact that for every actor in the supply chain, they deliver a product of some sort. Most queries in the systems in which supply chain data is stored, concern the traversal of supply chain actors and their products. The main quality attributes for storing data in supply chains are efficiency, usability, and security. The graph database can be considered more effective for supply chain data. The graph database has an advantage in efficiency, mainly when a high number of traversals are done. The graph database has a larger advantage when it comes to usability, which is emphasized when querying hierarchical structures, like the ones we defined in this paper.

5.1 Future work

A scenario that was not explored yet in this paper, is that of the use of graph algorithms in supply chains. There could be applications for these algorithms that would be easier to perform on supply chain data stored in graphs., this could give another advantage to the use of graph databases in supply chains.

Another way in which this research could be followed up is in the form of adding different implementations and kinds of databases to the `scm-ddg` tool. This could help SCIS developers and consumers to decide which database best fits their use.

To increase the applicability of the tool, more queries could be added that touch the same or new parts of performance in databases, and the model could be expanded to accompany new types of queries. And instead of the current 'Top Level'-based approach to data generation, a more dynamic structure could be adopted to reflect real-world scenarios better.

As we mentioned before, no extensive research was found on the usability between graph- and relational databases. This could also be a future direction for research, especially when a common graph query language could be adopted.

ACKNOWLEDGMENTS

I would like to thank SIM B.V. for a very interesting and enjoyable internship, and my supervisor Caspar Honée in particular, for pushing me in the right direction and giving me plenty of feedback on my findings.

REFERENCES

[1] Renzo Angles and Claudio Gutierrez. 2008. Survey of Graph Database Models. *ACM Comput. Surv.* 40, 1, Article 1 (Feb. 2008), 39 pages. https://doi.org/10.1145/1322432.1322433

[2] Miguel Hernandez Boza and Alfonso Munoz Muñoz. [n. d.]. ([n. d.]).

[3] E. Brewer. 2012. CAP twelve years later: How the "rules" have changed. *Computer* 45, 2 (Feb 2012), 23–29. https://doi.org/10.1109/MC.2012.37

[4] Lianping Chen, Muhammad Ali Babar, and Bashar Nuseibeh. 2013. Characterizing Architecturally Significant Requirements. *IEEE Software - SOFTWARE* 30 (02 2013), 38 – 45. https://doi.org/10.1109/MS.2012.174

[5] E. F. Codd. 1970. A Relational Model of Data for Large Shared Data Banks. *Commun. ACM* 13, 6 (June 1970), 377–387. https://doi.org/10.1145/362384.362685

[6] Roberto De Virgilio, Antonio Maccioni, and Riccardo Torlone. 2014. Model-Driven Design of Graph Databases. In *Conceptual Modeling*, Eric Yu, Gillian Dobbie, Matthias Jarke, and Sandeep Purao (Eds.). Springer International Publishing, Cham, 172–185.

[7] Janne M. Denolf, Jacques H. Trienekens, P.M. (Nel) Wognum, Jack G.A.J. van der Vorst, and S.W.F. (Onno) Omta. 2015. Towards a framework of critical success factors for implementing supply chain information systems. *Computers in Industry* 68 (2015), 16 – 26. https://doi.org/10.1016/j.compind.2014.12.012

[8] Janne M. Denolf, Jacques H. Trienekens, P.M. (Nel) Wognum, Jack G.A.J. van der Vorst, and S.W.F. (Onno) Omta. 2015. Towards a framework of critical success factors for implementing supply chain information systems. *Computers in Industry* 68 (2015), 16 – 26. https://doi.org/10.1016/j.compind.2014.12.012

[9] Oshini Goonetilleke. 2017. *Graph Database Management Systems: Storage, Management and Query Processing.* Ph.D. Dissertation. RMIT University, https://researchbank.rmit.edu.au/view/rmit:162343.

[10] Tonci Grubic and Ip-Shing Fan. 2010. Supply chain ontology: Review, analysis and synthesis. *Computers in Industry* 61, 8 (2010), 776 – 786. https://doi.org/10.1016/j.compind.2010.05.006 Semantic Web Computing in Industry.

[11] A Gunasekaran and E.W.T Ngai. 2004. Information systems in supply chain integration and management. *European Journal of Operational Research* 159, 2 (2004), 269 – 295. https://doi.org/10.1016/j.ejor.2003.08.016 Supply Chain Management: Theory and Applications.

[12] Weihong Guo, Qi Tian, Zhengqian Jiang, and Hui Wang. 2018. A graph-based cost model for supply chain reconfiguration. *Journal of Manufacturing Systems* 48 (2018), 55 – 63. https://doi.org/10.1016/j.jmsy.2018.04.015 SI: Advancing Manufacturing Systems Research at NAMRC 46.

[13] Benjamin T. Hazen, Christopher A. Boone, Jeremy D. Ezell, and L. Allison Jones-Farmer. 2014. Data quality for data science, predictive analytics, and big data in supply chain management: An introduction to the problem and suggestions for research and applications. *International Journal of Production Economics* 154 (2014), 72 – 80. https://doi.org/10.1016/j.ijpe.2014.04.018

[14] Joseph M. Hellerstein, Michael Stonebraker, and James Hamilton. 2007. Architecture of a Database System. *Found. Trends databases* 1, 2 (Feb. 2007), 141–259. https://doi.org/10.1561/1900000002

[15] Bill Karwin. 2010. *SQL Antipatterns: Avoiding the Pitfalls of Database Programming* (1st ed.). Pragmatic Bookshelf.

[16] Thomas Kelepouris, Katerina Pramatari, and Georgios Doukidis. 2007. RFID-enabled traceability in the food supply chain. *Industrial Management & Data Systems* 107, 2 (2007), 183–200. https://doi.org/10.1108/02635570710723804 arXiv:https://doi.org/10.1108/02635570710723804

[17] Mikko Kärkkäinen, Sanna Laukkanen, Sami Sarpola, and Katariina Kemppainen. 2007. Roles of interfirm information systems in supply chain management. *International Journal of Physical Distribution & Logistics Management* 37, 4 (2007), 264–286. https://doi.org/10.1108/09600030710752505 arXiv:https://doi.org/10.1108/09600030710752505

[18] João Ricardo Lourenço, Bruno Cabral, Paulo Carreiro, Marco Vieira, and Jorge Bernardino. 2015. Choosing the right NoSQL database for the job: a quality attribute evaluation. *Journal of Big Data* 2, 1 (14 Aug 2015), 18. https://doi.org/10.1186/s40537-015-0025-0

[19] Robert Campbell McColl, David Ediger, Jason Poovey, Dan Campbell, and David A. Bader. 2014. A Performance Evaluation of Open Source Graph Databases. In *Proceedings of the First Workshop on Parallel Programming for Analytics Applications (PPAA '14)*. ACM, New York, NY, USA, 11–18. https://doi.org/10.1145/2567634.2567638

[20] Tim McLaren, Milena Head, and Yufei Yuan. 2004. Supply chain management information systems capabilities. An exploratory study of electronics manufacturers. *Inf. Syst. E-Business Management* 2 (07 2004), 207–222. https://doi.org/10.1007/s10257-004-0035-5

[21] John T. Mentzer, William DeWitt, James S. Keebler, Soonhong Min, Nancy W. Nix, Carlo D. Smith, and Zach G. Zacharia. 2001. DEFINING SUPPLY CHAIN MANAGEMENT. *Journal of Business Logistics* 22, 2 (2001), 1–25. https://doi.org/10.1002/j.2158-1592.2001.tb00001.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/j.2158-1592.2001.tb00001.x

[22] Jose Miguel, David Mauricio, and Glen Rodriguez. 2014. A Review of Software Quality Models for the Evaluation of Software Products. *International journal of Software Engineering Applications* 5 (11 2014), 31–54. https://doi.org/10.5121/ijsea.2014.5603

[23] Jose Miguel, David Mauricio, and Glen Rodriguez. 2014. A Review of Software Quality Models for the Evaluation of Software Products. *International journal of Software Engineering Applications* 5 (11 2014), 31–54. https://doi.org/10.5121/ijsea.2014.5603

[24] Justin Jay Miller. 2013. Graph Database Applications and Concepts with Neo4j.

[25] Umezuruike Linus Opara. 2003. Traceability in agriculture and food supply chain: A review of basic concepts, technological implications, and future prospects.

[26] Jongkyung Park, Kitae Shin, Tai-Woo Chang, and Jinwoo Park. 2010. An integrative framework for supplier relationship management. *Industrial Management & Data Systems* 110, 4 (2010), 495–515. https://doi.org/10.1108/02635571011038990 arXiv:https://doi.org/10.1108/02635571011038990

[27] Jaroslav Pokorný, Michal Valenta, and Jiří Kovačič. 2017. Integrity constraints in graph databases. *Procedia Computer Science* 109 (2017), 975 – 981. https://doi.org/10.1016/j.procs.2017.05.456 8th International Conference on Ambient Systems, Networks and Technologies, ANT-2017 and the 7th International Conference on Sustainable Energy Information Technology, SEIT 2017, 16-19 May 2017, Madeira, Portugal.

[28] Andreas Riel, Serge Tichkiewitch, Damian Grajewski, Weiss Z, Draghici A, George Draghici, and Richard Messnarz. 2009. Formation and Certification of Integrated Design Engineering Skills, Vol. 10.

[29] Eugenia Stathopoulou and Panos Vassiliadis. 2009. Design Patterns for Relational Databases. (01 2009).

[30] James R. Stock and Stefanie L. Boyer. 2009. Developing a consensus definition of supply chain management: a qualitative study. *International Journal of Physical Distribution & Logistics Management* 39, 8 (2009), 690–711. https://doi.org/10.1108/09600030910996323 arXiv:https://doi.org/10.1108/09600030910996323

[31] Wen Jun Tan, Allan N. Zhang, and Wentong Cai. 2019. A graph-based model to measure structural redundancy for supply chain resilience. *International Journal of Production Research* 0, 0 (2019), 1–20. https://doi.org/10.1080/00207543.2019.1566666 arXiv:https://doi.org/10.1080/00207543.2019.1566666

[32] Ebrahim Teimoury, Issa Chambar, Mohammad Reza Gholamian, and Mohammad Fathian. 2014. Designing an ontology-based multi-agent system for supply chain performance measurement using graph traversal. *International Journal of Computer Integrated Manufacturing* 27, 12 (2014), 1160–1174. https://doi.org/10.1080/0951192X.2013.874584 arXiv:https://doi.org/10.1080/0951192X.2013.874584

[33] Chad Vicknair, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen, and Dawn Wilkins. 2010. A Comparison of a Graph Database and a Relational Database: A Data Provenance Perspective. In *Proceedings of the 48th Annual*

Southeast Regional Conference (ACM SE '10). ACM, New York, NY, USA, Article 42, 6 pages. https://doi.org/10.1145/1900008.1900067

[34] Matthew A. Waller and Stanley E. Fawcett. 2013. Data Science, Predictive Analytics, and Big Data: A Revolution That Will Transform Supply Chain Design and Management. *Journal of Business Logistics* 34, 2 (2013), 77–84. https://doi.org/10.1111/jbl.12010 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/jbl.12010

A EXPERIMENTAL SET-UP CONFIGURATION

Table 6: Virtual Machine Configuration

Processor	AMD Ryzen 5 1600 @ 3,2-3,6GHz 2 cores / 4 threads enabled
CPU Execution Limit	90%
Memory	8192 MB @ 2800 MHz
Storage	Crucial BX100 SSD
Operating System	Ubuntu 18.04.2 LTS 64-bit

Table 7: Neo4j instance settings

Application version	3.5.6 Enterprise Edition
Heap memory maximum size	4096 MB
Pagecache size	2048 MB

Table 8: MySQL instance settings

Application version	8.0.2 Community Edition
InnoDB buffer pool size	6144 MB

B SAMPLE QUERIES

C QUERY RATIONALE

```
WITH RECURSIVE cte (
        parent_product_name, parent_product_id, child_product_name, child_product_id, depth
)
AS
(
        SELECT parent.name, parent.id, child.name, child.id, 1
        FROM consists_of
        JOIN product AS parent ON consists_of.parent_product_id = parent.id
        JOIN product AS child ON consists_of.child_product_id = child.id
        WHERE parent.name = "Top␣Level␣Product␣Awesome␣Steel␣Car"

    UNION ALL

        SELECT parent.name, parent.id, child.name, child.id, cte.depth + 1
        FROM consists_of
        JOIN product AS parent ON consists_of.parent_product_id = parent.id
        JOIN product AS child ON consists_of.child_product_id = child.id
        JOIN cte ON cte.child_product_id = parent.id
        WHERE depth < 5
)

SELECT parent_product_name, parent_product_id, child_product_name, child_product_id
FROM cte
```

Listing 1: Example query for the relational database, in the language SQL (type: adjacency list). Gets all products from a product hierarchy, up to a depth of 4.

```
MATCH (p:Product)-[:CONSISTS_OF*1..4]->(c:Product)
WHERE p.name = 'Top␣Level␣Product␣Awesome␣Steel␣Car'
RETURN p.name, ID(p), c.name, ID(c)
```

Listing 2: Example query for the graph database, in the language Cypher. Gets all products from a product hierarchy, up to a depth of 4.

Table 9: The scenarios coupled to the queries, and the performance aspects touched. All queries are from the perspective of a single supplier.

#	Scenario	Aspects
Q1	I want to see how my product is built up	Traversal, Single Type
Q2	I want to see how my product is built up to a certain depth	Traversal, Single Type, Scalability
Q3	I want to see whether and where a certain product is used in my supply chain	Traversal
Q4	I want to see who my suppliers are and who they supply to	Traversal
Q5	I want to know whether my supply chain is compliant to something	Traversal, Filtering
Q6	I want to perform a search into products in my supply chain that matches certain criteria	Traversal, Filtering, Scalability
Q7	For every product, check whether a certain keyword appears in its name or description	Traversal, Full-Text Search
Q8	I want to see which products are produced in a certain region of the world	Filtering
Q9	I stopped performing a certain activity in all supply chains	Changing structure
Q10	I have added a new sub-supplier that performs an activity between two existing ones	Changing structure