Right fit Database Technology for Discounts

ALEXANDRA GALONJA & ANNELIE SINANDER
BACHELOR´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Right fit Database Technology for Discounts

Alexandra Galonja
Annelie Sinander

Department of Electrical and Information Technology
Lund University

Supervisor: Christin Lindholm, Suchit Gupta

Examiner: Christian Nyberg

June 16, 2022

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

This thesis has been conducted in order to compare how four different database management systems (DBMSs) perform in a limited number of areas, in order to draw conclusions about which one works best for storing discount data at IKEA. The DBMSs being evaluated are MongoDB, MariaDB, Neo4j and PostgreSQL. To enable a comparison, a form of testing was performed - a benchmark. The benchmarking consists of setting up a database for each DBMS to be compared. Discount data is then stored in each database and the execution time is measured for each test database. To accumulate information about how the database is set up but also about the different discounts used by IKEA, two interviews were conducted where this information was able to emerge.

The results from the testing and the interviews showed that PostgreSQL, which is the DBMS used today, performed the best overall. This was the case for the licensing metric, for instance, where Postgres was the only DBMS that does not come with a licensing cost. The results from one of the DBMSs, MongoDB, only consisted of estimates and not exact measurements and therefore the conclusion is not completely unambiguous. Furthermore, Neo4j's results when it came to the latency and scalability were deviating compared to the remaining DBMSs, which may have been due to an disadvantageously configured database.

Lastly, a number of development opportunities are presented which show that there is an immense potential to further explore the question of how the discount data should be stored. However, IKEA is proposed to continue to use the clear winner PostgreSQL as a DBMS.

Keywords

Database Technologies, Discounts, SQL, NoSQL, Latency, Signal-to-noise-ratio

Sammanfattning

Detta examensarbete har utförts i syfte att jämföra hur fyra olika databashanterare (DBMS) presterar inom ett begränsat antal områden för att därigenom dra slutsatser kring vilken som fungerar bäst för att lagra data om rabatter på IKEA. De databashanterare som testas är MongoDB, MariaDB, Neo4j och PostgreSQL. För att åstadkomma en jämförelse utfördes en form av testning - en benchmark. Benchmarken består av att en databas sätts upp för varje DBMS som ska jämföras. Därefter lagras rabattdata i respektive databas och exekveringstiden mäts för varje testdatabas. För att ackumulera information kring hur databasen sätts upp men också kunskap om rabatterna som finns hos IKEA, utfördes två intervjuer där denna information framkom.

Resultaten av testningen och intervjuerna visade att PostgreSQL, den databashanterare som används idag, presterade bäst överlag. Postgres uppvisade lägst latens och är den enda av de testade databashanterarna som inte medför några licenskostnader. Resultaten från en av databashanterarna, MongoDB, bestod endast av uppskattningar och inte exakta mätningar och därför är inte slutsatsen helt entydig. Fortsättningsvis visade det sig att Neo4j hade väldigt avvikande värden jämfört med resterande databashanterare, vilket kan ha berott på en ofördelaktigt konfigurerad databas.

Slutligen presenteras ett antal utvecklingsmöjligheter som visar på att det finns en stor potential att vidare utforska frågan kring hur rabattdatan ska lagras. IKEA föreslås emellertid att även fortsättningsvis använda den klara vinnaren PostgreSQL som databashanterare.

Nyckelord

Databasteknologier, Rabatter, SQL, NoSQL, Latens, Signal- brusförhållande

Table of Contents

List of Figures

List of Tables

Foreword

After many months of arduous work with this thesis and lots of moments of pure frustration, it is finally completed. We would like to thank our supervisor from IKEA, Suchit Gupta, who guided us and helped us every week with the thesis. In addition, we would like to thank Christin Lindholm for all her support and guidance, along with her feedback that helped us tremendously. Lastly, we would like to thank our examiner, Christian Nyberg, who was also helpful with his guidance and feedback.

Chapter 1
Introduction

1.1 Introduction

Database technologies are under a constant development which makes it pivotal for companies to utilize and manage the technology that is accessible to them. In the case of IKEA IT, due to historical reasons, there has been a continuous use of the same database technology for storing discounts (see chapter 1.2 for further information). The database management system (DBMS) that is currently in use belong to the row-oriented RDBMS group. Examples of these are Oracle and Postgres. RDBMS is an acronym for relational database management, the actual database is one that contains different data that are associated with each other [15]. Due to this database technology being row-oriented, the information is structured into rows.

There are, however, problems that exist within the current database technology which makes it crucial to find a technology that is more suitable. The current problems consist of the fact that it is inefficient from a time and performance point of view to store discounts in a hierarchical way.

There is a possibility that better and more suitable technologies have been developed through these years that the current one has been in use. As a result, it could be considered worthwhile to investigate other options in the pursuit of finding the most efficient one in terms of computational performance, time-efficiency, infrastructure cost and memory usage.

A solution to the aforementioned problems is to investigate other technologies that could be more suitable for the traversal of discount data. Conducting investigations and learning more about other technologies but also about how discount data is being stored and used by IKEA today – and conducting various tests to see if the results are in fact as anticipated – will enable the possibility of finding the best solution to the ones that are under investigation. Several technologies should be investigated and compared to the row-oriented database technology. These include a graph-based database technology storing data in graphs, a document-oriented technology where data is stored in documents and a column-based technology using columns for storing data.

1.2 IKEA IT

This thesis work was executed in cooperation with IKEA, a global furniture company established in 1943, with more than three hundred stores worldwide [16]. Ingka Group (IKEA Retail) is a franchise with IKEA being the commercial name for the company [17]. Ingka Group is one of 12 other franchises within the company [18]. The objective for each year when concerning the franchise is to enable more than two thousand products to be brought to the market [18].

In conjunction with always trying to launch new products, there is at times necessary to incorporate the usage of discounts in the old products as well as in the new ones. When concerning IKEA, these discounts mainly come into existence on specific occasions due to IKEA always striving to have some of the lowest prices on the market, which does not make it necessary at all times. These discounts occur when it comes to various deals on their membership plan *IKEA Family*, offers that are time restricted, personalised vouchers, coupons, rewards, best before dates, end date sales, just to name a few. Since there are several occasions where the discounts come into place, it is of foremost importance to manage these discounts in effective ways, starting with having a database that is able to manage this.

1.3 Purpose

The purpose of this thesis is to investigate whether any of a few chosen contemporary database technologies is better suited for storing discounts than the currently in use database technology. The technologies that were investigated consisted of graph based, document-oriented, columnar based and the technology currently in use - row based. The aim was for IKEA IT to use the most suitable database technology out of those that would be examined in this thesis, to save as much resources as possible. These resources consist of processing time and memory but also licensing costs and maintenance.

1.4 Goals

The thesis investigated the advantages and disadvantages associated with four contemporary database technologies in the context of managing discounts. An evaluation, based on measurements from testing, of how the database technologies perform in terms of efficiency, was performed. After the technologies had been evaluated, they were compared to one another and one of them was deemed most suitable and therefore suggested for IKEA IT to use.

1.5 Questions

This thesis will answer the following questions:

1. How is discount data being stored and used at IKEA today?
2. What are the trends regarding discounts, both at IKEA and in the retail industry in general?
3. What are the advantages and disadvantages associated with different database technologies on a general level?
4. What criteria are to be considered useful in a database technology handling discounts?
5. How does the four different database technologies perform in relation to the criteria answered in the previous question?

1.6 Scope

The scope was limited to four database technologies: graph based, document-oriented, columnar based and the technology currently in use - row based. Only one DBMS out of each of these database technologies was assessed; it should be emphasized that the goal is not to find the best DBMS but to conclude which technology is best suited for the management of discounts.

Chapter 2
Technical Background

2.1 Discounts

According to the Cambridge Dictionary [26], discounts are defined as: "a reduction in the usual price" with the example of receiving the discount in the case of buying numerous duplicates of the same item [26]. Discounts usually occur with the intention of being incorporated as a business strategy to encourage more people to buy a company's products.

The world of discounts has increased, especially during the recent years due to the uprise of new companies. Due to this, there has been a trend among many of these companies to incorporate and heavily invest in a variety of discount strategies with the goal of acquiring more customers. With more companies comes a lot of competition which is the reason why it is now more important than ever to, through the use of discounts, entice people into purchasing products.

2.1.1 General trends

Discounting trends generally tend to fluctuate. Companies such as Foodora [36], a food delivery company, has tried and is still trying to entice their customers through different types of discounts. The types of discounts usually offered by Foodora vary. Sometimes free delivery is offered when ordering from a restaurant from a particular range, if the order meets a certain price criteria or price threshold. Other times, if the customer invites a friend, both the customer and the friend get a certain percentage off. The offer can also consist of different kinds of vouchers. These kinds of discounts appeal to many people, hence Foodora encourages visitors of their website to place an order by displaying discounts.

In addition to IKEA, which is the subject of this thesis, Mio is another rather big company in Sweden specialized in selling furniture [35]. Mio could be considered a competition to IKEA and has similarly to Foodora incorporated the use of discounts in their selling scheme. The most often occurring type of discount on their website consists of reduced prices on certain furniture. Some furniture is discounted as much as 50 percent. Potential customers might feel eager to place an order while the item is still in stock and the discount is still valid. When these

types of discounts are happening, they are usually not applied on all the furniture (except for on special occasions). In general, the discounts are applied to one certain type of furniture. One week the discounts could be applied to e.g., beds, while the week after sofas are the furniture that are being discounted.

Moreover, discounts are more likely to occur during certain occasions. These occasions consist of Black Friday sales and after Christmas sales, e.g. The strategies that are incorporated during these special occasions usually begin with a teaser on the companies' websites and with the help of advertisement on the products that are discounted. These two strategies allow the company to reach out with their offers to a lot of different people, which might encourage these people to put in orders from that store opposed to other stores that do not incorporate this kind of strategy in their selling scheme.

2.1.2 Discounts at IKEA

The discount strategy that IKEA has incorporated for years is not that massive compared to some other companies. Their goal is not to heavily discount their items but to instead ensure that they have among the lowest prices on the market compared to other companies, making these heavy discounts redundant.

2.1.2.1 Discount strategies

IKEA, as well as other companies, have incorporated well thought out discount strategies. Depending on the situation and what the desired achievements are, a strategy may be more or less suitable to use. It is important to think about which strategy works best in which situation.

Family rewards: When a customer acquires a membership in IKEA Family, they receive a membership card through which they get access to various rewards, i.e., discounts, which are exclusive to them [27]. Whenever the card has been used, the card owner receives the possibility of winning a gift card. This possibility occurs in the form of a raffle, a raffle that occurs every month at IKEA [27]. This is one way of enticing the customers into becoming members.

Markdown: Markdowns are probably what first comes to mind when people are thinking about discounts, and they are part of a markdown strategy at IKEA. A markdown is a price reduction on an article. The reason behind the price reduction varies. One reason behind a markdown is when a collection gets obsolete and therefore is offered at a lower cost for it to get purchased quicker.

Personalized coupons: A personalized coupon is a personal coupon that is intended to be used by one person. When such a coupon has been used, it gets invalidated. These coupons can be created for several reasons. One of these reasons could be that the company wants to compensate the customer.

Vouchers: A gift card is an example of a voucher that is offered at IKEA. When

purchasing one, the gift card can either be digital or physical.

2.1.2.2 Discount types

A couple of discount types have already been introduced, but there are additional types incorporated in the general discount strategy at IKEA. The most important ones are listed in this section.

Amount based discounts: An amount-based discount arises when a customer spends above a certain price point and receives a reduction on the purchase, e.g. spending 400 SEK and getting a 40 SEK reduction.

Combined discounts: A combined discount means that a customer gets a reduction on the purchase when combining certain items. A customer could, for instance, be offered a certain percentage off when buying a bed and a mattress together.

One time use coupons: These kinds of coupons are limited to one use per coupon.

Public coupons: Public coupons are meant for public use, meaning they could be used by anyone.

Quantity discounts: Quantity discounts appear when a customer buys a certain number of items and gets one of them at no cost.

Tiered discounts: Tiered discounts are also dependent on the quantity purchased by the customer. In the case of tiered discounts case, however, the discount comes in the shape of a percentage and only when a certain amount of items are purchased. E.g., when buying four items or more, a discount of 15% is applied.

2.2 Database theory

In the following section, information that is relevant to the databases and the DBMSs is presented and explained.

2.2.1 ACID

ACID is an acronym and a highly relevant concept in databases. The letters represent the following words: Atomicity, Consistency, Isolation and Durability. Each of these four words represent an important quality for guaranteeing the validity of the data in a database. All databases should be able to attain the ACID properties which is why it is crucial that the people who are implementing the systems incorporate them [7].

Atomicity: Atomicity is a word that means that if changes are about to occur to the data, either the whole transaction or no part of the transaction is permitted to occur [24].

Consistency: Consistency entails that only consistent data can be inserted into the database [24]. All data that is introduced must follow the rules of valid data for the database in question.

Isolation: Isolation means that a transaction has to be kept isolated in regard to other transactions [24], meaning that the execution of the transactions has to occur separately [7]. The transactions might be executed concurrently but must logically appear as if they were executed sequentially.

Durability: Durability entails that the data should be irreversible after a change has been committed [24]. Even in the case of failure there should not occur changes of any kind.

2.2.2 CRUD

CRUD is an acronym that plays an essential role in the world of data. It is a term that describes the four basic operations that can be performed on the data in a database. Each letter represents one of the following words (operations) respectively: Create, Read, Update and Delete. When concerning a database and the data being stored in that database, the following definitions can be uncovered:

Create: The creation of new entries added to existing data [3].

Read: The retrieval of data that is currently stored in the database [3].

Update: The editing of data that is currently stored in the database [3].

Delete: The removal of data that is currently stored in the database [3].

2.2.3 Latency

According to the Cambridge Dictionary [40], latency, when it comes to computing, is defined as: "the delay between an instruction to transfer (= move) computer information and the information being transferred, for example over the internet". By combining latency with the aforementioned CRUD, it allows for a very fundamental metric that plays an essential role in this thesis. When comparing several different DBMSs, among the most crucial aspects to compare is how the execution times vary. That is where latency comes in, which allows for a comparison to occur by measuring the execution time. When latency is henceforth mentioned, what is meant is the time for a query to be executed.

2.2.4 Signal-to-Noise ratio

Signal-to-noise ratio is a measure that is used for expressing a desired signal or information in relation to the background noise needed to generate said information or signal. To calculate it, both the signal or information and the background noise need to be quantified. Signal-to-noise-ratio is calculated by dividing the quantified signal or information by the quantified background noise. The result is thus a quota or percentage.

A high signal to noise ratio is considered bad. The goal should be to keep the SNR at a minimum. The noise can be viewed as the cost of producing a certain amount of data, the cost being computational power.

When referring to signal-to-noise-ratio in this thesis, what is meant is the amount of data that must be processed for the execution of a query. The phenomenon will be discussed, and the signal-to-noise-ratio will be estimated for the DBMSs mentioned in chapter 2.3 in a broad sense but no exact calculations will be made.

2.2.4.1 JOIN statements & SNR

The biggest signal-to-noise ratio in a relational database is produced by complicated queries with one or multiple JOIN statements, performed on multiple big tables [5]. This is due to the big amount of data that must be processed for such a query.

2.2.4.2 Graph databases and SNR

Depending on what type of data is stored and how the data is being stored, the migration from a relational database to a graph database can decrease or increase the signal-to-noise ratio. Graph databases are particularly suitable for databases with a lot of connections between the data points [48]. Simply explained, it is sufficient to just follow the relationships, or the edges, to find data that is interconnected. There is thus no need to search through an entire dataset, i.e., table, until the sought out data has been found, as would have to be done in a relational database [48].

For example, imagine there are two different databases: a graph database and a relational database. These databases store information about people and information about friendships between these people. In a relational database it would be customary to use one table to store information about people and one table to store the friendships between people. Bob is one of those people. In a relational database, one would have to search through the friendship table to find all of Bob's friends and then search for those friends in the people table to be able to retrieve information about them. In a graph database, Bob's node already has the connections to Bob's friends stored. They do not have to be detected in a friendship table and then searched for in a people table, it is possible to go directly to where the data about the friends is stored and retrieve the information. This is a simple

example, but it displays the benefits of a graph database compared to a relational database when there is a need to traverse a lot of connections.

2.2.4.3 Indexing

If a table in a relational database is indexed on a particular attribute that is also the target for searching, it can be a game changer for the performance. This is due to the DBMS now knowing where to look for the information. It does not have to search top to bottom through entire tables in the database to find the information it is looking for.

Graph databases can also use the method of indexing and the performance depends on the type of indexing service being used by the DBMS. Neo4j works best for full-text indexing and does not perform quite as good on other data types [5].

2.3 Database Management Systems

This chapter is a short presentation of the four different database management systems that will be tested, evaluated and finally compared to one another, along with an introduction to the technology each DBMS is using.

A DBMS is needed to access the data in a database and to make use of the database. *Teorey, Lightstone and Nadeau* define a database management system as "a generalized software system for manipulating databases" [11]. IBM defines it as "essentially nothing more than a computerized data-keeping system", explaining it as a system where operations can be performed to change the structure of the database or manipulate the data inside [25].

In this thesis, four database management systems are compared to one another and evaluated in relation to the metrics later mentioned in chapter 2.4.1.

2.3.1 MongoDB

The intention of developing MongoDB was not to satisfy the requirements of the masses, but to develop a DBMS that had the ability of working efficiently with documents, making it document based [6]. For this to be possible, it was necessary for the developers to exclude certain functions as these could lead to a decline in efficiency within the system [6]. As a result, the recovery time after a system has failed has increased tremendously [6]. According to the developers, this is something that falls within their concept since their philosophy consists of the databases not carrying only one but several backups [6].

The way MongoDB is structured is similar to the way JSON files are structured, essentially meaning that the documents differ from each other due to them having different data structures, e.g. [23]. Additionally, their usage of storage is interchangeable since both consist of one individual document [6]. Despite their similarities, JSON is not being used by MongoDB, BSON is what is being used

instead (this stands for Binary-JSON). As a result of the usage of BSON, it has allowed MongoDB to substantially increase the speed in regard to processing and searching for various documents [6]. Furthermore, additional features have been included that do not exist in JSON, e.g., being able to manage other data such as binary data [6].

2.3.2 MariaDB

As opposed to MongoDB, MariaDB was developed with the intention of pleasing everyone [8]. MariaDB initially started its development after MySQL started getting controlled by Oracle which resulted in it getting developed in the form of a fork of MySQL [8]. Many went over to using MariaDB instead of MySQL due to its interconnection with quite big companies [8]. Despite this, the actual transition from MySQL to MariaDB is quite easy since no code must be changed [8].

Like MySQL, the model that is being used by MariaDB is a client and server model [2]. What this model does, with the help of a server program, is that it keeps managing the incoming requests from one or multiple clients [2]. Moreover, MariaDB is a DBMS that is completely free for private use, the same goes for its source code, making it open source [9]. Making it completely free is something that is of value to the creators of MariaDB and therefore they have created a foundation to help them keep it free [10]. The purpose of this foundation is to protect the actual code by allowing it to stay the same even in the future [10].

2.3.3 Neo4j

Neo4j is the world's leading graph DBMS [19]. Neo4j offers a cloud license, a community edition license and three different enterprise licenses. For a company with the size of IKEA, the enterprise edition comes at a cost.

Inside a graph database, data is stored as nodes and the connection between two nodes in a database is called a relationship [20]. Each node holds a list of all its relationships [21], and this way of storing data cleverly eliminates the need for costly JOIN operations and search-and-match computations [20]. This is due to there not being a need to search for connections that are already stated and accessible, as is the case in a graph database where each node holds a list of each and every one of its relationships. The way a graph DBMS is managing relationships is what many consider the biggest advantage of the graph DBMS technology compared to the RDBMS technology.

2.3.4 Postgres

Postgres, or PostgreSQL, is an object-relation database management system [22]. It is an open-source DBMS, meaning it is completely free to use and entails no costs in licenses for commercial use either.

In a relational database, data is stored as tuples in tables [1]. One tuple equals a row in a table and a tuple has one or more attributes. Each attribute makes up a

separate column in the table. The object part in the term object-relation suggests that there is an object-oriented aspect to the technology. It is possible to create complex objects with attributes in Postgres. These objects can in turn be related to each other in different ways.

2.4 Benchmark testing

It is not easy to put the method of performing a benchmark into one single definition. Broadly speaking, benchmarking is a tool for comparing performance levels between participants and it can be applied to many areas [13].

In the context of databases, in this thesis, benchmarking is the method of comparing how four database management systems representing four different database technologies perform in relation to one another. To measure the performance, the metrics defined and explained in the subsection below are used.

2.4.1 Metrics determination

A prioritization of the metrics have been established through a discussion with the supervisor at IKEA. The supervisor has in turn prioritized based on what is most important to investigate. The metrics are as follows, given in a prioritized order, along with points of suggestions for what to explore when performing a benchmark on the metric:

1. Licensing
    - Identify the various kinds of licensing available for each DBMS (see chapter 2.3).
    - Identify what functionality each kind of licensing offers and pick the most fitting license for each DBMS (see chapter 2.3).
    - Identify the costs for the licenses chose the best fit for IKEA.
2. Latency
    - Identify the most used, i.e., the most often executed, queries (create, read, update, delete) in the currently used Postgres database for storing discounts at IKEA.
    - Identify for each DBMS (see chapter 2.3) which of the queries being most used are the most performance demanding.
    - Establish the response time for each DBMS (see chapter 2.3) to perform the most used queries.
3. Signal-to-noise ratio
    - Determine for every DBMS (see chapter 2.3) respectively the amount of data that must be processed to perform the most used queries (create, read, update, delete).

- Identify which of the most commonly used queries produce the highest signal-to-noise ratio.

4. Scalability

   - Identify how well the DBMSs (see chapter 2.3) respond to the increment and decrement of the amount of data.
   - Establish the impact on the performance, i.e., the response time, for each DBMS (see chapter 2.3) when increasing or decreasing resources.

5. Memory

   - Identify the amount of memory each DBMS (see chapter 2.3) requires storing the database.
   - Establish how much cache memory is used by each DBMS (see chapter 2.3) in the CPU.
   - Identify how much RAM, i.e., working memory, is used by each DBMS (see chapter 2.3) when performing the most commonly used queries executed in the currently in use Postgres database for storing discounts at IKEA.
   - Establish for each DBMS (see chapter 2.3) which of the queries identified as being most used require the most RAM.

Chapter 3
Methodology

The following chapter describes all the necessary information regarding the methodology. It describes the five separate phases the thesis underwent, phases that were mainly following the waterfall model but with one exception. This exception involves the first phase, Literature Study, since it was necessary to revisit this phase due to the addition of new information that had not been thought of prior to proceeding to new phases. Writing this thesis was an integral part of the process that occurred every week during all of these phases. The phases can be seen in figure 3.1.

Figure 3.1: The five phases in chronological order.

3.1 Literature Study

The first phase consisted of a literature study. It was crucial for this to occur in the beginning since a lot of information was still unknown. Moreover, the aim of this phase was to collect the correct and relevant information and understanding how to put the facts into practice. The information that was collected consisted of facts about the technologies covered in this thesis as well as results and conclusions from previous research and testing. In addition, a lot of time was invested in learning the best and proper ways of conducting interviews, but even more time was invested in preparing the benchmark testing.

3.2 Preparing & Conducting Interviews

The interview phase consisted of interview methodology studies based on Annika Lantz's book *Intervjumetodik* [12]. In the book, the interview work is divided into

three phases: the preparatory phase, the interplay during the interview and lastly how the answers are processed. These three phases determine the outcome of the interviews. Furthermore, the initial questions must cover important background info, such as asking the person being interviewed their occupation and education. Questions must be asked in an order that seems logical to the person being interviewed. Therefore, this order may or may not be the order that the interviewer finds logical. During this second phase, the interviews are prepared and conducted, though the answers are only transcribed and not processed. The processing takes place in the fifth and last phase.

3.2.1 Preparatory Work

Drawing conclusions from figures and statistics is easier than doing the same from the result from an interview, and therefore it is of importance that the interviews are highly qualitative ones [12]. The quality can be increased by constantly improving the interview methodology. Lantz emphasizes that it is beneficial to determine what type of analysis should be done based on the data, prior to conducting the interviews.

Research has shown beginners prepare more and conduct better interviews than experienced interviewers, stemming from the fact that experienced interviewers overestimate their own capabilities. There is thus an attitude-based aspect to the quality of an interview.

During the preparations, the purpose of the interview or interviews should be determined. The problem should also be linked to the existing theory to enable the formulation of what problems are to be solved and thereby formulate what questions should be answered. Both knowledge and skill are, according to Lantz, needed to perform an interview [12].

3.2.2 Make use of the results

Lantz claims that a well-conducted interview must achieve a certain level of reliability and validity [12]. The conclusions must be able to be reviewed by outsiders for the data to meet certain requirements for usability. To maintain a high quality, it is important not to fill in information gaps in the interviews with assumptions based on the answers the person being interviewed has given to other questions. The gaps should only be noted and not filled with made up information. The first step in processing the data is to reduce the amount of raw data. It is important this step is performed in a well thought out way to ensure that the data most relevant to the question at issue is not lost and less important data consequently being kept.

3.2.3 Structured versus Open interview

There are many ways in which an interview can be executed. Depending on the level of structure, a variety of different results and conclusions can be derived from

the interview. A fully open interview and a fully structured interview should be viewed as structural opposites, each on one end of the spectrum.

A completely open interview allows the person being interviewed to give their own full picture. As a result, it is less of a discussion and more based around letting the person think aloud and reason with themselves. The answers in such an interview are about the qualities of a phenomena. A structured interview, on the other hand, is based on some phenomena in some form of delimited context established in advance, as opposed to an open interview where the context is determined by the person being interviewed. The aim of the structured interview is to discover something quantitatively by asking certain questions and hopefully receive a quantitative answer.

A directed open interview is a form of an open interview, but more structured than a fully open one. In a directed open interview, the questions are grounded in receiving qualitative answers regarding a phenomenon, but instead of letting the person being interviewed decide which direction the interview shall take, the interviewer decides which question areas are to be covered.

A semi-structured interview is a more open form of a structured interview. In such an interview, the question areas are to be decided beforehand and followed up in a manner determined in advance. Unlike a structured interview, a semi-structured interview allows follow-up questions. Instead of only receiving answers, the interviewer also gets an appreciation of how meaningful the questions are to the person being interviewed.

3.3 Benchmark Preparations

The third phase consisted of making the preparations that were necessary for the actual benchmarking to be able to happen. These preparations consisted of looking for and deciding on which database tools to be incorporated in the actual testing and receiving the laptop the testing would be conducted on from IKEA. Additionally, scripts had to be written and sorted out in all the DBMSs.

3.3.1 Logs

Incorporating the use of logs to track the execution time and determine the latency in all DBMSs, enables the possibility of not including a software developed specifically for benchmarking. By analysing the log entries and the time it takes for the queries to fully execute, the latency can be determined.

3.3.1.1 System mode & Kernel mode

A processor running on Windows can either use user or kernel mode. User mode is used for running applications and kernel mode is used for running core operating system components [37]. When executing a query, the database software is working

in user mode and the query operations are being performed in kernel mode. It is preferable that the database handler does not impact the result and therefore logs measuring the execution time in kernel mode are the ones that have been used.

3.3.1.2 Postgres Logs

A Postgres server can be configured through its postgresql.conf file to log certain data. By setting specific parameters, it is possible to define what kind of information should be logged and where the log output should end up. With the right parameters, both kernel and user execution time are logged [38].

An alternative way of finding out the execution time for a specific query performed on a Postgres database is to use a profiling tool such as EXPLAIN ANALYZE [44]. EXPLAIN ANALYZE is built on top of the EXPLAIN tool and can be used to show the time spent on the different steps in the execution of a query, along with the total execution time in kernel mode. The tool is used by putting the text "EXPLAIN ANALYZE" (without quotation marks) before the query that is to be analysed. Along with the query result, the result from EXPLAIN ANALYZE is also returned. Example: EXPLAIN ANALYZE SELECT X FROM Y.

3.3.1.3 MariaDB Logs

To be able to measure the execution time and to log other data required for benchmarking MariaDB, the easiest solution found was to use the SHOW PROFILE statement [41]. The first step in being able to use it is by setting the profiling session variable to one in the MariaDB console with the following command:

SET profiling = 1;

This enables the activation of the statement **SHOW PROFILES**, which shows a list of the most recently executed queries along with their execution times [41].

3.3.1.4 MongoDB Logs

When it came to the logging regarding MongoDB, this was found to be more difficult than the rest of the DBMSs. After a lot of research, the best way of conducting the logging was found to be through applying the following method on every query [42]:

explain("executionStats")

For this to work, similarly to MariaDB, the profiling level variable had to be changed by writing out the following statement either in the console of MongoDB or in the database tool of one's choice [43]:

db.setProfilingLevel(2)

By setting the profiling level variable to 2, the Database Profiler logs the data and can write out the execution time of the query when applying the aforementioned method on every query, **explain("executionStats")** [43]. Furthermore, when executing the query and the method, two different time measurements show up: "executionTimeMillis" and "executionTimeMillisEstimate". The first measurement states all the time that is required, from selecting the most suitable query plan to the actual execution [42]. Since only the kernel mode execution times of the other DBMSs were measured, it was decided this was not suitable for the purpose of this thesis. Therefore, the second measurement was found to be the more appropriate one since it gives an estimate of how fast the queries are executed. This is not ideal in comparison to the logging measurements the other DBMSs had to offer, but it was the best alternative the thesis workers were able to find.

3.3.1.5 Neo4j Logs

Instead of reading the logs of Neo4j and trying to decipher which execution time was the appropriate one to use, it was decided to download a plugin that visualized all the values. The name of the plugin is: "Query Log Analyzer" and it was developed and released by an Neo4j employee [45]. For it to show the kernel mode execution time, the following statements had to be added in the neo4j.conf file [4]:

dbms.track_query_cpu_time=true

dbms.logs.query.time_logging_enabled=true

When these two statements had been enabled, it was possible to verify the execution time by searching for the executed queries in the plugin and look at the number that showed under the "Avg CPU". This number shows the execution time of the CPU [4].

3.3.2 Computer specifications

To conduct a benchmark testing that is as reliable and consistent as possible, the testing was carried out on one and the same laptop. The specifications for the laptop that was used is specified below.

- **Processor:** Intel(R) Core(TM) i5-6300U CPU @ 2.40 GHz 2.50 GHz
- **System type:** 64-bit operating system, x64-based processor
- **Installed RAM:** 16,0 GB

3.3.3 Database tools

Initially, the plan was to use the same database tool for all the DBMSs to avoid simple yet possibly very impactful mistakes such as confusing different types of execution times given by the DBMSs' different logging systems and methods. The right database tool might have been able to streamline the process and it is also probable that a database tool, developed specifically for benchmarking, would

make the competition between the DBMSs fairer. This kind of database tool was, however, never used due to it not being preferred by the supervisor at IKEA.

DBeaver Version 22.0.4.202205011839 [39] was the tool that was used in conjunction with PostgreSQL and MariaDB since both DBMSs are supported by this tool. It was initially decided to use this tool for all the DBMSs, but as mentioned earlier, it was later discovered that this was not possible. It does, however, support MongoDB as well but only when upgrading the license that is used with it. Since the upgrade only enables one more DBMS to be used it was decided by the thesis workers to use two completely different tools with the ones that were not supported. Instead, Studio 3T was the database tool that was chosen to be used with MongoDB, due to it being very user-friendly but also it being one of the most popular database tools when it comes to MongoDB. Lastly, the database tool that was used with Neo4j was Neo4j's own tool, Neo4j Desktop, which can be downloaded on the Neo4j website. It was an obvious choice since it is their own tool, and it therefore enables the possibility of getting support directly from the company if problems would arise.

3.3.4 Queries evaluation

As explained in section 3.5.3, the most used queries were obtained through a request to one of the software engineers involved in working with the database. All the queries were SELECT statements, with one UPDATE statement as an exception. In conjunction with the basic operations of a database, CRUD, *read* is the most prevalent operation (with one exception of *update*). Thus, there will be no investigation into *create* and *delete* in relation to the metrics.

Fifteen queries were received from interviewee 2 (a complete list can be viewed in table B.1, appendix B). About a third of the queries (query 10-15) were alike and trivial in the sense that all of them were queries in the form of *SELECT* x *FROM* y *WHERE* $z = b$, where the information was retrieved from primary key indexed tables where z was the primary key. Moreover, three queries (query 9-11) operated on tables with less than two hundred rows.

3.4 Benchmarking

In the fourth phase, the benchmarking was conducted which enabled the data retrieval to fulfil the metric requirements. This phase was divided into two parts - the first one consisted of the execution of the most used queries in all of the DBMSs and the second one consisted of the retrieval of information on the internet that was not able to be retrieved through testing, e.g., the licensing metric. The queries that were the most used had been given by the software engineer employed by IKEA who had participated in the second interview. The benchmarking of each DBMS is explained in more detail in the corresponding section of this chapter.

3.4.1 Postgres Benchmarking

When benchmarking Postgres, a local database server was set up along with a database for testing. A data definition language (DDL) SQL script was easily created in the database tool DBeaver. The skeleton of the database (i.e., schemas, tables, dependencies, indexes and functions, etcetera) was created by executing a modified version of such a script. The tables were thereafter filled with data from the testing environment's discount database. Since it was not possible to get access to the actual production data, the data that was used came from the pre-production environment.

3.4.2 MariaDB Benchmarking

A local MariaDB database server was set up for benchmarking, and a database for testing was created. To create the schemas, tables, dependencies, indexes and functions needed to set up the database, the Postgres create script was altered to be executed in the MariaDB database. To enable such alterations, Postgres specific data types had to be replaced with MariaDB specific data types. Lastly, the database had to be filled with data. This data was downloaded from the original database as CSV files and uploaded to each table respectively, with the original mapping being kept.

3.4.3 MongoDB Benchmarking

A local MongoDB database server was set up for benchmarking, and a database was created. To create the schemas, or collections as they are called in MongoDB, the original Postgres schemas were downloaded as CSV files and later loaded into the MongoDB database for testing. Postgres' specific data types had to be replaced with MongoDB's specific data types. Since keys do not exist in MongoDB but provide an important indexing function which had to be considered when evaluating the latency, indexes were created for the tables based on the primary keys from the original database.

3.4.4 Neo4j Benchmarking

A local Neo4j database server was set up in Neo4j's own database tool, "Neo4j Desktop". The setup occurred with the help of a tutorial that was described in depth on Neo4j's website [46]. After the setup was completed, the importing of CSV files begun which thereafter continued with the conversion of data types. When CSV files are imported in Neo4j, all the different data types are set as strings by default. Therefore, it is necessary to convert the ones that are not in fact strings. These could be easily changed by applying the following conversion function in the case of integers: **toInteger()**.

The Postgres discount database was migrated to the Neo4j database used for testing by transforming every row to a node, without creating any connections between the nodes. Every node coming from the same original Postgres table was created to be of the same type. Thus, it was possible to search within a specific

type of node, as an equivalent for searching an entire table. The original columns were turned into attributes of the nodes. Lastly, indexes had to be added since indexes have been incorporated in all the DBMSs that have been used throughout the thesis.

3.5 Results

In the fifth and final phase, a finalization of the end results that had been established in the previous phase occurred. The results were compiled in various tables and the percentage differences regarding the Scalability metric were calculated. This was the main purpose of the following phase, in addition to enabling the possibility of drawing the necessary conclusions based on the results. These conclusions consisted of the ones that had to be drawn due to the metrics that had been established in 2.4.1 and the questions from 1.5 that had to be answered in the conclusion chapter. Additionally, the interview results were also compiled in this phase. During this process, the thesis workers had to review the interviewees' answers that had been written down in phase two. This was completed to be able to write the summaries of the two interviews, enabling additional conclusions to be drawn.

3.6 Source criticism

The various sources have been able to be found by using the internet. These sources range from different articles and research papers to books written by the actual developers of some of these DBMSs, with MariaDB being an example of this. Most of these sources have, however, been found on various websites. Due to this, it has been crucial to incorporate a high level of source criticism when looking for sources in order to be able to find the most reliable ones.

3.6.1 Articles

A couple of articles were used, covering some of the DBMSs with MariaDB being one of them [1][2][3][4][5]. When it came to these sources and which information was included in the thesis, the information was only factual and no comparisons to other DBMSs had been made. One way of making this source more dependable could have been to include other sources that were making the same claims. The same goes for the other articles that were referred to, the information could have also been strengthened with other sources. With the use of articles, however, the sources in the articles are known. Due to this, an option could have been to investigate these sources as well, to confirm whether the sources are truly reliable.

3.6.2 Books

Books have been used as much as possible, due to it being harder in some ways to spread false information this way. The reasoning behind this statement is that

using books, the authors' names are printed out while on the internet, e.g., it is easier for people to be anonymous. Additionally, before printing the final version of books they have already undergone multiple stages and been proofread by several people. Despite these statements, it is still crucial to be careful when it comes to the purpose of the books, who the people who have authored the books are and how credible they are. Information about the authors is, for the most part, included in books. This makes it easier to investigate and see how dependable the books actually are. Based on this information, one can realize whether someone is biased. One sign of this could be that the author of the book is the same as the person who has created something. This holds true when it comes to the book written by R. J.T Dyer, who was one of the creators of MariaDB and MySQL. In these situations, it is of importance to not include certain opinions when deciding to include these kinds of sources or to at least include other sources besides the one that has the potential of being biased. This was not the case, however, and it could have been done to ensure a higher reliability.

3.6.3 IKEA IT Employees

One of the phases consisted of conducting interviews with two IKEA employees. During these interviews, information about the various discounts that exist at IKEA were uncovered and how they are currently being used at said company. Due to the information coming directly from the source, it made it highly dependable. Besides these two interviews, two other meetings with two other employees occurred where information about discounts were presented as well. Due to the information coming from diverse sources at the same company, information with no inconsistency, has made it even more reliable. In addition to these sources, the IKEA website has been used as a source as well, which has further strengthened the reliability of these sources.

3.6.4 Websites

Due to the easy access of the internet, this kind of source has been the most used one throughout this thesis. This has made it even more important to be careful when it comes to deciding which sources are more dependable than others. There are several reasons for why this is important, especially since the internet consists of plenty of people, one of these reasons is that the information that is presented can be biased. This can even include raw data due to how it is being presented [14]. Presenting certain data, or other information, without including other useful information that can have influence on what is presented can change the entire opinion of a person when it comes to a certain subject. Rumours are another factor to be conscious of, since these can be easily spread through the internet rather than through e.g., printed books. Usually, these rumours start with some kind of truth to then be turned and twisted into something that does not reflect the truth anymore [14]. Due to this, it is crucial to look at information that come from various sources and based on that information decide whether what has been stated is to be viewed as truthful or not [14].

The sources regarding the various DBMSs have been deduced dependable since most of the information came from the official websites of the DBMSs. The information that has been collected from these websites are factual and the various features can be controlled using the DBMSs mentioned. Due to this, the information cannot be considered biased since the information was not presented regarding comparing them to other technologies on the websites or to highlight them in certain ways that reflect the companies in good ways. These comparisons were made in the thesis by comparing them to each other, based on the information given on each website.

Chapter 4
Analysis

The following chapter consists of an analysis of the results and the reasoning behind certain decisions that were made throughout the entire process.

4.1 Decision making

During the thesis process, through all the phases, multiple decisions had been made in order for the process to move forward in the desired direction. Everything from deciding on which tools to choose all the way to deciding on what kind of techniques to incorporate in the conducted interviews, are all decisions that have been made and are presented in the following section.

4.1.1 Database setups

To set up the Postgres database for benchmarking, a DDL script was created using a built-in feature in DBeaver. A script had to be created since there was no existing script offered by IKEA. A script could have been created manually but it was decided to use DBeaver to speed up the process.

4.1.2 Database Management Systems

The reasoning behind which database technologies to include had its basis in the thesis description, and the choice of DBMSs was based on the individual DBMS's popularity among users as well as companies. It was suggested in the thesis description that in addition to the currently in use relational technology graph based, document based and columnar based technologies were the ones of interest. It was not specified, however, which DBMS to use from each technology, which led to the investigation of which ones to choose. After some research of several types of DBMSs, it was deduced which ones to choose. The main reason for deciding on the most used ones in each technology was due to the DBMSs' big communities which enabled the possibility of getting provided with a lot of support in case something would go wrong. Therefore, if an error would have occurred during the setups, it would have been a lot easier to get the support that was needed compared to other, smaller communities. When it comes to bigger companies, such as IKEA, it is also of importance that the communities are big.

4.1.3 Database tools

The choice of using DBeaver as a database tool for manipulating the MariaDB and Postgres databases came naturally since this tool was introduced by a software engineer at IKEA in the beginning of the thesis. It was therefore deemed user friendly while no other alternatives had any obvious advantages, and therefore DBeaver continued being used. The reason for why not much work went into the choice of database tool is that the database tool itself should not impact the result, just ease the thesis work. Also, the database handler would for the most part only be used for executing SQL scripts and importing and exporting data, which can be considered a rather basic feature of a database tool, and thus there was no need to search high and low for a database tool with rare features.

4.1.4 Licenses

When it came to the licensing, it was quite an easy decision to make. Since IKEA is a big company, the Enterprise Licensing was deemed to be the license of interest in this thesis, regarding all the DBMSs. The reasons behind this were due to the advantages that came along with it but also, in order for everything to go as smoothly as possible, the Enterprise Licensing seemed to be the obvious choice. The advantages that came with it differed among all of the DBMSs mentioned, the result of these is presented in chapter 5.5. The other types of licenses were not able to support that big of a company and were therefore only recommended for uses by smaller companies and individuals, which is another reason for making this decision.

4.2 Interview decisions

During phase two of the thesis, two interviews were prepared for and thereafter conducted. The two interviewees come from different professional backgrounds. One of them has a technical background and is working as a software engineer while the other one is working as a pricing leader. Along with the two diverse backgrounds comes two different inputs, which is the reason interviewing two people with diverse backgrounds felt more convenient than interviewing two people from the same background. Moreover, due to the differences, the interview questions were adjusted accordingly. The reason for why the interview with the software engineer was important was due to it being crucial to find out how the discount databases were being used at IKEA, to be able to test the kinds of queries that were most commonly used, for instance. Additionally, help was needed with setting up the database that is used for discounts to be able to conduct the necessary testing, which was also given by the same person. This ended up being espacially useful since all of the questions were answered which made the practical side of the thesis work easier. Furthermore, the other interview was of significant importance as well, due to questions about discounts being answered and the selling scheme that exists within IKEA being uncovered. It was necessary to obtain this knowledge since a lot had not been known prior to this.

4.3 Interview Processing

It was decided to transcribe the interviews right after they had been executed in the second phase. The transcription was done to get an overview of what had been established in the interviews, and to create a good foundation where no information was left out for the processing to be performed correctly. Since the answers from the interviews were considered by the thesis workers to be intelligible, there was no rush to process the results. The raw data was thought to be easy to understand and it was thus possible to move forward in the process. The raw data was processed during the last phase, when finalizing the thesis. The thesis workers sifted out what they considered to be most relevant for the question at issue, and left out what was irrelevant, in accordance with the methodology described by Lantz [12]. The thesis workers were adamant in making sure that no information gaps were filled so that good interview methodology practices could be followed.

4.4 Problems & solutions

In conjunction with making various decisions, both advantages and disadvantages arise. With disadvantages, however, problems usually come about and can have big effects on the process as well as the end results. Due to this, it is of importance to take these into account and reflect on, for development to occur. The problems that occurred during the thesis are therefore presented in the following section along with their solutions.

4.4.1 Database setups

Problems arose when the Postgres DDL script would be adjusted for the setup of the MariaDB database. One of the issues was that certain data types in the script were Postgres specific data types meaning they did not exist in MariaDB. The solution to the issue was to replace the affected data types with MariaDB approved data types. In some cases, this was easy. In others, the number of bits within a variable changed with the data type substitution.

Another problem that occurred when setting up the MariaDB database was that there were tables that did not have a primary key in the Postgres database. When these tables would to be set up in MariaDB, MariaDB did not accept a table without a primary key. Because of this, a primary key had to be created on what seemed to be the most fitting column (i.e., the id column in one case).

4.4.2 Database tools

One problem occurred when the MongoDB database would be set up. MongoDB drivers were not included in the DBeaver Community Edition, which was the software that had been used for the previously tested DBMSs. No functional MongoDB drivers that could be imported and used were found for free online. Instead, another database tool was used: Studio 3T. The DBeaver license could be upgraded for a certain amount of money but since the database tool itself should

not impact the result, it was decided to simply use another free tool. The same can be said about Neo4j where it was later decided to choose Neo4j's own tool, for the reasons mentioned in 4.1.2.

4.4.3 Conduction of interviews

Two main problems occurred during the interviews: language confusion and bad audio. Language confusion arose from the fact that English was not the mother tongue of the people who conducted the interviews, and only the native language of one of the interviewees. On one hand, this language confusion had a negative impact on the interplay and made the interview go more in the direction of being a questioning than being an easy conversation. On the other hand, it eventually led to some of the right questions not being asked. There were two reasons for the right questions not being asked. One was that information did not get picked up in real time and therefore the questions arose afterwards during a listening through the recorded audio. The other was that the interviewers produced questions to ask but did not know how to phrase it. Both two main problems mentioned was of course worsened by the fact that not much was known about the subject in question yet and by the fact that the interviews leaned towards being of the open type.

The interviews were conducted online over Microsoft Teams instead of in person due to the pandemic restrictions still being in place at IKEA. The audio quality had a substantial impact on the interviews. Bad audio quality, on top of the language confusion problem already explained, added to the arising confusion. There was, however, one great benefit to conduct the interviews over the internet; it made the recording of the interviews easy through the built-in recording features in Teams. Even the video from the interviewees screen sharing could be recorded, which was of much use and would not have been done just as easily in real life.

4.4.4 Data retrieval

There were some problems regarding getting access to the proper data that was tested. It was necessary to get access to the production data since the testing would have to occur on the data that is being used by the employees at IKEA, otherwise it could be considered meaningless to conduct the testing. Additionally, it would not be possible to find the most used queries without it. It was not possible, however, to get access to the production data in the end. As a result, the testing had to be conducted on the data in pre-production which contained less data points than the production one. Due to only getting access to this data, it was not possible to retrieve the most used queries. Therefore, help was received from the IKEA employee with whom one of the interviews had occurred with, and the most commonly used queries were sent, and the testing could thereafter begin.

4.4.5 Execution time, MongoDB

Everything went well when it came to measuring the execution times in all the DBMSs, except for when it came to MongoDB. A lot of research was conducted

to be able to retrieve the most accurate execution time when it came to this DBMS. The problem was, however, that MongoDB does not provide any values for the execution time in certain situations. The best and only way to find out the execution time is to do as explained in section 3.3.1.4. The problem that arose with doing it this way was that the execution time is always measured as 0 if the execution time is below 1 millisecond, which was the case for all of the queries except for one. The main reason behind this problem was that pretty much all of the tables, or documents as MongoDB is a document-based DBMS, were indexed. Therefore, the correct document could be found straight away with no problem. Unfortunately, a solution could not be found, and the exact execution times are still unknown. However, it is safe to say that the queries last for less than 1 millisecond.

4.4.6 Benchmarking, Neo4j

There were several issues in relation to the benchmarking of Neo4j. The first issue was the fact that Neo4j was blocked on the laptop that was received from IKEA and it took quite some time to get it unblocked. When it finally was unblocked and everything was set up within Neo4j, additional problems showed up, with one being finding the execution time. After multiple hours of searching the web, it was finally found that the execution time could be read using an analysing tool that was developed by an Neo4j employee. Lastly, when the execution times were verified, it could be clearly seen that something was wrong. All the execution times showed extremely high values in comparison to the other DBMSs and there was therefore a high possibility of the Neo4j database setup being wrongly configured. There were attempts in trying to sort this out, but nothing ended up being successful in the brief time that was left within the time frame for this thesis. One reason to this could have been that the strongest advantage with the use of Neo4j had not been incorporated - the usage of nodes. Viable solutions to this could have also been to contact Neo4j directly and ask for their help regarding this.

Chapter 5
Result

In the following chapter, the results are presented; the end results that are derived from the benchmarking and the results that came from the conducted interviews.

5.1 Interview result

In this thesis, the interview technique that was deemed most suitable was a semi-structured interview technique, meaning that the questions were determined prior to the interviews. Additionally, follow-up questions were asked as well, which were dependent on the answers that were given by the interviewees. Two interviews were conducted in the second phase and all the interview questions that were asked are compiled in Appendix A. Due to the interviewees having their expertise in two different areas, the questions that were asked differed.

5.1.1 Interviewee 1

The first interview involved an individual with a role in pricing. Due to this, they possessed a lot of knowledge when it comes to the discounting at IKEA and how these were incorporated in their selling scheme. The interview contributed to a lot of information getting obtained when it came to the discounts and pricing but also information about IKEA. The information that was considered the most useful for the purposes of this thesis was the information about the discounts, what kind of discounts that exist, which are described in chapter 2.1. The information that was shared consisted of in-depth descriptions of discounts such as the IKEA Family Discounts, but also about a less common discount that can be applied in the case of buying a dining table, e.g. In the case of buying a dining table, the average customer tends to look for four chairs that match the table. Instead of offering a discount in the form of a percentage, an offer in the form of "buy four chairs, pay for three" can be implemented instead. Additionally, another useful piece of information that was shared involved the current trends regarding discounts. One trend that is on the uprise, especially in the US, is the various types of bundles that are being offered at IKEA. One example of this kind of bundle is the bed-in-a-bag bundle. This bundle includes a bed together with a couple of items that are considered customary to have in a bed, e.g., duvet and pillows. This comes to

show that the discounts that exist within IKEA, but also within other companies, can be a lot more complex than a simple percentage off.

5.1.2 Interviewee 2

The interviewee in the second interview explained that their job title was software engineer and that they are working within a subdomain under the customer domain. The team they belong to are responsible for developing, maintaining and supporting a discount managing tool named SDM (short for Sales Discount Management) according to the base and business requirements that come from the product team. The interviewees team delivers the product to the co-workers who in turn use the tool for creating discounts. They continued explaining that there are several databases in use but the discount database relevant for this thesis is a Postgres database hosted on Google Cloud Platform.

The co-workers, typically responsible for campaigns and doing promotions, use the SDM tool to create discounts for certain markets and channels. A market could, for example, be a country, but the discounts can also be store specific. The discount can be created for the website and mobile app or be created as a store level discount. About two or three people in each market can create a discount for their own market.

According to the interviewee, there are not any apparent problems when it comes to the currently used database management system. When collecting data for reporting purposes, the searches might take a couple of seconds, but otherwise the room for improvement is about optimization. Elksearch is the tool being used for searches. It was explained that Elksearch is particularly good for searching the items and details from their database, but it was not explained nor questioned in which way it was good. Any kind of delay cannot be seen, but it can be said that the tables containing the prices could be slow when querying the BigQuery for thousands and thousands of records. The system that is sitting on top of the database can manage millions and millions of records. For example, when working with BigQuery, all that data can easily be handled with the data being very fast as well.

When asked about if they have explored other alternatives to Postgres the following was explained. Regarding the relational technology, both PostgreSQL and SQL are being used. Different types of databases are used by different teams, and this is possible since work is executed in Google Cloud Platform where several database technologies are supported. Many teams are using Postgres though, and Postgres can be used at an enterprise level and is also open source. There are people at IKEA who are well experienced with Postgres. The interviewee describes it as being a very mature database in a sense.

The team responsible for developing the SDM tool has tried to optimize the total flow, since the flow is quite big. The information is coming from different environments and there is an internal latency when the data is flowing through different

channels and different systems. For example, whenever a coupon is used there is a desire for information about all the details, but that would take some time to provide. First, the basic information, order number and coupon number, is provided so the coupon can be marked as resolved. Then, later on, the whole data is provided so that all information can be appended which is one way of optimization.

Another kind of optimization is the way the data will be stored before the processing has been done. There is not a lot of things done to the data when it is received. The most important task is to update the database. Later on, the other processes can take over the work of different jobs that needs to be done. This is another type of optimization that is possible to do. Whenever one is working with a transaction type of data, time is essential. It is important in an optimization point of view to get the information as soon as possible, according to the interviewee.

The most common extraction of data is discount related data. Discount information is stored in discount tables, and the coupons are stored in their corresponding coupon tables. Data is extracted whenever somebody is modifying the data. For example, DATE is one important field that indicates if the discount is valid or not. Searches are made to the database to answer these questions (among others): what is the status of the discount, what is the qualification type, on what items should the discount be applied? This is more like a business perspective, they said. The team responsible for developing the SDM tool are providing the discount information to another team who then uses this information when calculating the pricing. If there are any changes made to the discount, the other team is updated about the changes allowing them to do the right adjustments for the price calculations.

Whenever a discount is set up or changed, inserts and updates are made. Coupon data is often updated since updates are made when a coupon is used. For the database, transactions per seconds (TPS) is somewhere between 2000 to 5000. Since the system is hosted on Google Cloud Platform, it can scale to that number. The scaling is based on the requirements Google Cloud platform has. The interviewee can't point at an exact number, but the database can cater to thousands of requests per seconds, especially for the coupon related information because that is more of a transactional or live thing since the coupon has to be killed once it is used.

When asked about the most frequent searches, the interviewee emphasized that SDM is for the co-workers. They use it to get discount information or coupon information, those are the most common objects of business items that would be fetched. There are searches within the tool itself, but those searches mainly concern item information; when a discount is created, a search for which items the coupon or discount is valid for has to be done. If a specific discount is being searched for, the search is done based on the discount name or discount ID, the status of the discount or some other information. The search will not be very much like NLP or language-based text - it is mainly directed. With respect to discounts or coupons, the users of the SDM system know what they are looking for so they directly set for the right information. The users know the system and the business domain, making their searches quite different from if it was customers doing the

searches.

The challenges that are faced when developing the SDM tool concern the integration with the other systems. For example, when a software engineer is working with SDM, they cannot sit by themselves. Instead, they need to talk with the people who are developing other tools within the IKEA ecosystem. This is because it is all intricately connected. Integration is an issue because the integration must be in place with the right kind of authentication and authorization, the interviewee explained. With a big company like IKEA, where thousands of tools are being used, there are always these kinds of challenges.

Finally, questions were raised whether they would benefit from using other database technologies, like graph databases. About Graph databases, the interviewee claimed there are tools for capturing the BPM and not for creating. It is for the coupons and how data is coming to BigQuery which can be exported from there. They can be exported to different BPM utilized by finding out the trends and how the discounts are performing.. It can show how the discounts are performing, the total amount of items, and the number of orders that are using a specific discount. Business users who are looking at the long-term trends take this information into other tools. The data can be supported by other tools, graph databases or other BPM tools, where it can be used for analysing and studying the trends.

5.2 Latency

For determining the latency of each one of the DBMSs mentioned throughout the thesis, benchmarking was conducted and the execution time of all those DBMSs was hence determined. The most used queries were initially identified with the help of the software engineer who was interviewed for this thesis, and these are listed in Appendix B. Thereafter, the testing was able to be conducted and the execution time for every DBMS was established and listed down below. Due to the establishment of the execution time of each DBMS, it was possible to determine which of the queries were the most performance demanding.

None of the DBMS's were able to execute query #1, #2, #7 or #8. Query #1 and query #2 could not be executed because those queries require encryption functions the thesis workers did not have access to. Query #7 and query #8 could not be executed since the thesis workers did not have access to the tables on which the queries were to be performed. Query #4 could not be performed on the MongoDB database nor the MariaDB database since no corresponding functions could be found for the Postgres specific array_agg() function. Query #10 could not be performed on the MariaDB database due to problems migrating the data. The queries that could not be executed are marked with "-". An overview showing the execution times of each DBMS can be seen below, in table 5.1.

Table 5.1: An overview of the query execution time for each DBMS.

#	*MongoDB*	*MariaDB*	*Neo4j*	*Postgres*
1	-	-	-	-
2	-	-	-	-
3	$<$ 1 ms	0.7684 ms	254 ms	0.497 ms
4	-	-	-	0.792 ms
5	11 ms	0.8266 ms	3 ms	0.360 ms
6	$<$ 1 ms	0.6975 ms	46 ms	0.7862 ms
7	-	-	-	-
8	-	-	-	-
9	$<$ 1 ms	0.7125 ms	140 ms	0.175 ms
10	$<$ 1 ms	-	328 ms	0.208 ms
11	$<$ 1 ms	0.7862 ms	217 ms	0.195 ms
12	$<$ 1 ms	8.7665 ms	-	0.163 ms
13	$<$ 1 ms	0.6892 ms	145 ms	0.171 ms
14	$<$ 1 ms	0.7183 ms	117 ms	0.320 ms
15	$<$ 1 ms	0.8266 ms	8 ms	0.7862 ms

5.2.1 MongoDB Latency

A list of the execution times from the MongoDB benchmarking for the most executed queries. In table 5.2, the executed queries are lined up from fastest to slowest. There were a couple of queries that were not able to be run for assorted reasons, these are therefore marked with "-".

Table 5.2: A table of MongoDB's execution times.

#	*Execution time*
3	$<$ 1 ms
6	$<$ 1 ms
9	$<$ 1 ms
10	$<$ 1 ms
11	$<$ 1 ms
12	$<$ 1 ms
13	$<$ 1 ms
14	$<$ 1 ms
15	$<$ 1 ms
5	11 ms
1	-
2	-
4	-
7	-
8	-

Additionally, for some of the queries being evaluated in the MongoDB benchmarking, no exact value for the execution time was provided by the DBMS. This is due to MongoDB not being able to estimate the exact duration when the execution time is less than 1 millisecond. Therefore, it is known that the queries were executed faster than 1 millisecond, but the exact duration is unknown.

5.2.2 MariaDB Latency

A list of the execution times from the MariaDB benchmarking for the most executed queries. In table 5.3, the executed queries are lined up from fastest to slowest. A couple of queries were not able to be run for assorted reasons, these are therefore marked with "-".

Table 5.3: A table of MariaDB's execution times.

#	*Execution time*
13	0.6892 ms
6	0.6975 ms
9	0.7125 ms
14	0.7183 ms
3	0.7684 ms
11	0.7862 ms
15	0.8266 ms
12	8.7665 ms
5	9.0706 ms
1	-
2	-
4	-
7	-
8	-
10	-

5.2.3 Neo4j Latency

A list of the execution times from the Neo4j benchmarking for all the most executed queries. In table 5.4, the executed queries are lined up from fastest to slowest. There were a couple of queries that were not able to be run for assorted reasons, these are therefore marked with "-".

Table 5.4: A table of Neo4j's execution times.

#	*Execution time*
5	3 ms
15	8 ms
6	46 ms
14	117 ms
9	140 ms
13	145 ms
11	217 ms
3	254 ms
10	328 ms
1	-
2	-
4	-
7	-
8	-
12	-

As can be seen in the table, the final execution times coming from benchmarking Neo4j showed to be extremely high in comparison to the other DBMSs.

5.2.4 Postgres Latency

A list of the execution times from Postgres' benchmarking for the most executed queries. In table 5.5, the executed queries are lined up from fastest to slowest. There were a couple of queries that were not able to be run for assorted reasons, these are therefore marked with "-".

Table 5.5: A table of PostgreSQL's execution times.

#	*Execution time*
15	0.051 ms
6	0.103 ms
12	0.163 ms
13	0.171 ms
9	0.175 ms
11	0.195 ms
10	0.208 ms
14	0.320 ms
5	0.360 ms
3	0.497 ms
4	0.792 ms
1	-
2	-
7	-
8	-

5.2.5 The Most Performance Demanding Queries

When it came to MongoDB, MariaDB and Postgres, none of the results from executing the queries showed that any of the DBMSs performed exceptionally bad. When it came to Postgres, all the execution times were distributed within a small range. MariaDB, by contrast, had two queries that were a lot slower than the rest: #5 and #12. The fastest execution times ranged from 0.6892 milliseconds to 0.7862 milliseconds while the two slowest execution times were more than 8 milliseconds for MariaDB. Ninety percent of MongoDB's execution times were below 1 milliseconds while one of the queries, which was query #5, had an execution time of 11 milliseconds. When comparing the execution time of this query to how the same query performed with MariaDB and Postgres, it was also MariaDB's most performance demanding query while it was the third slowest one when it came to Postgres. Since this query acted similarly in three of the DBMSs, it can be deduced that query #5 is most likely the most performance demanding query, despite it not performing exceptionally bad.

Lastly, Neo4j was the one DBMS that was performing the worst regarding every one of the queries that it could be evaluated for. The query that was the most performance demanding in the other DBMSs, query #5, was the one that performed the best when it came to Neo4j, with an execution time consisting of 3

milliseconds. Query #10 was the one that was the most performance demanding in Neo4j and when comparing it to the other DBMSs it had one of the better execution times.

5.3 Signal-to-noise ratio results

In this section, the signal-to-noise ratio for the most used queries will be discussed in relation to the DBMSs on which the benchmarking has been carried out in this thesis.

5.3.1 SNR-ratio in relational databases results

The biggest signal-to-noise ratio in a relational database is produced by complicated queries with one or multiple JOIN statements, performed on big datasets. This is due to the big amount of data that must be processed for such a query. Luckily, no such queries were found in the list of most executed queries. Instead, almost every one of the queries were performed using indexing. Those that were not performed using indexing were performed on small tables and, most importantly, these tables could be optimized by applying indexes. Due to the possibility to apply indexing on an attribute being the subject to *SELECT* x *WHERE* $y = z$ statements, the SNR can be kept low when using a relational DBMS for storing discount data at IKEA.

5.3.2 SNR-ratio in Graph DBMSs

Since the Postgres discount database was migrated to the Neo4j database used for testing by transforming every row to a node, without creating any connections between the nodes, the graph technology method of significantly reducing SNR by traversing related nodes was not used. This, in combination with the fact that Neo4j is bad at indexing other types of data than full-text, makes it probable that Neo4j has to travel through all nodes of a specific type when looking for a node, or multiple nodes, with a certain attribute. In a dataset with thousands of nodes, as is the case for a couple of the queries, this, in theory, makes the SNR quite high and a lot higher than the SNR for the relational database.

5.3.3 Amount of Processed Data

MariaDB, MongoDB and Postgres all use the same way of keeping the amount of data that must be processed to a minimum, by indexing the data. Since all the tables are indexed, except for the table in query #5, it might not come as a surprise that query #5 is exceptionally slow using all of the three DBMSs. In relation to the examined database in this thesis, this exceptionally slow query could easily be optimized to perform at a fraction of the time measured just by indexing the table.

There are no join operations occurring among the most used queries. Considering all the most used queries are operations performed on data that can easily be sifted

out from a big set of data points by using indexed tables, the signal-to-noise ratio can be kept at a minimum.

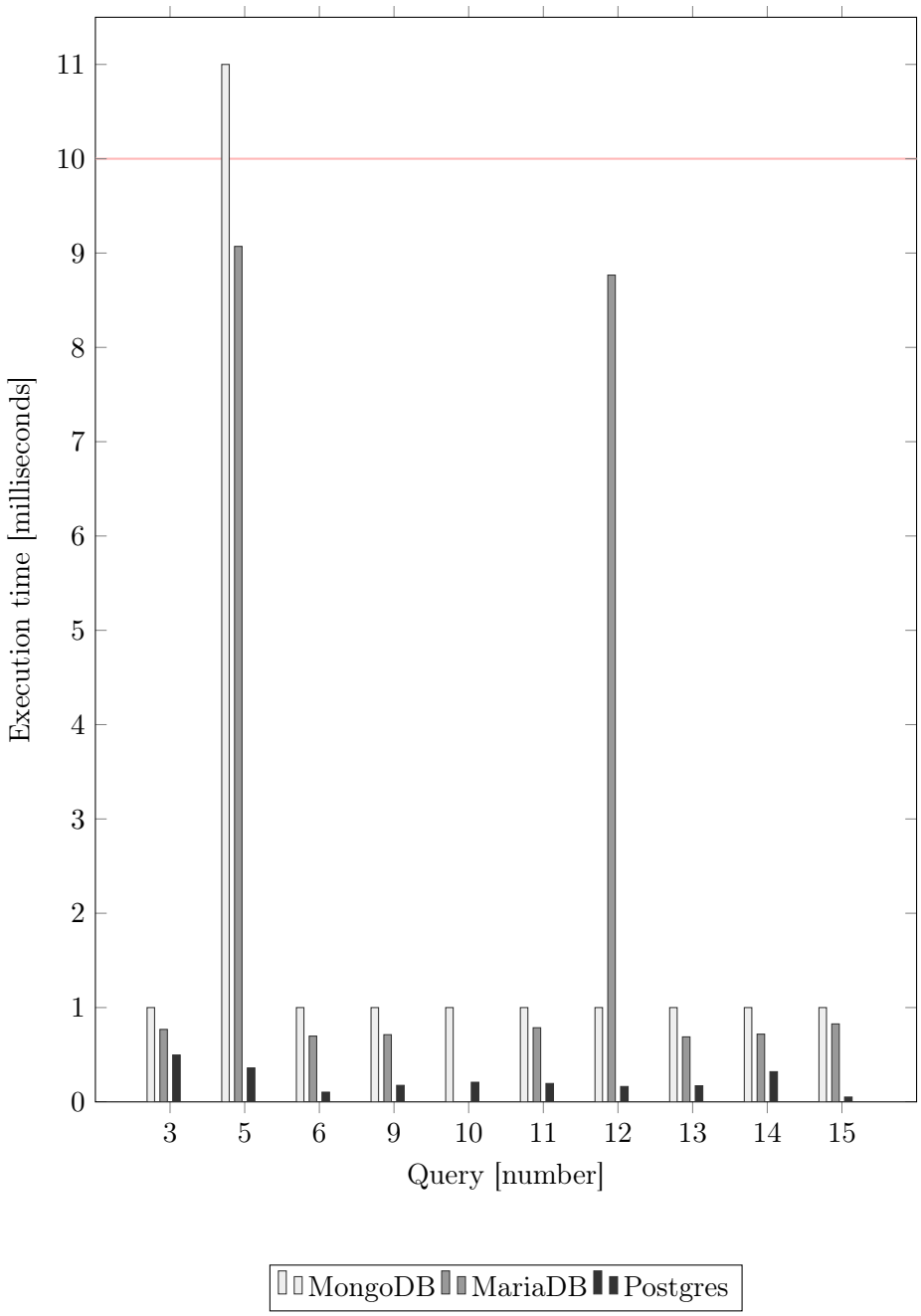

Figure 5.1: An overview of the execution times of the queries that were executable in all DBMSs except for Neo4j.

5.4 Scalability

In this section, the results of how well the various DBMSs responded to the increment and decrement of the amount of data are presented. The data was increased by a factor three and are presented with the final execution times of each DBMS and a percentage difference.

5.4.1 MongoDB Scalability

A table displaying the normal execution times from the MongoDB benchmarking for the most executed queries, along with the execution times when increasing the data by a factor three. Additionally, the difference between these two execution times have been calculated and divided by the old execution time and are presented as percentages. The executed queries are lined up from fastest to slowest. A couple of the queries were not able to be run for assorted reasons, these are therefore marked with "-". An overview of the execution times for each query is shown in table 5.6.

Table 5.6: An overview of MongoDB's execution times.

#	*Execution time*	*Execution time (3x data)*	*Difference*
3	$<$ 1 ms	3 ms	$>$ 200 %
11	$<$ 1 ms	3 ms	$>$ 200 %
6	$<$ 1 ms	$<$ 1 ms	?
9	$<$ 1 ms	$<$ 1 ms	?
10	$<$ 1 ms	$<$ 1 ms	?
12	$<$ 1 ms	$<$ 1 ms	?
13	$<$ 1 ms	$<$ 1 ms	?
14	$<$ 1 ms	$<$ 1 ms	?
15	$<$ 1 ms	$<$ 1 ms	?
5	11 ms	- ms	-
1	-	-	-
2	-	-	-
4	-	-	-
7	-	-	-
8	-	-	-

5.4.2 MariaDB Scalability

A table displaying the normal execution times from the MariaDB benchmarking for the most executed queries, along with the execution time when increasing the data by a factor three. Additionally, the difference between these two execution times have been calculated and divided by the old execution time and are presented as percentages. The executed queries are lined up from fastest to slowest. A couple of queries were not able to be run for assorted reasons, these are therefore marked with "-". An overview of the execution times for each query is shown in

table 5.7 and a visual overview is presented in figure 5.2.

Table 5.7: An overview of MariaDB's execution times.

#	*Execution time*	*Execution time (3x data)*	*Difference*
11	0.7862 ms	5.6061 ms	613.1 %
14	0.7183 ms	2.6865 ms	274.0 %
9	0.7125 ms	2.5853 ms	262.8 %
15	0.8266 ms	2.8767 ms	248.0 %
3	0.7684 ms	2.594 ms	237.6 %
6	0.6975 ms	1.5889 ms	128.0 %
13	0.6892 ms	0.9284 ms	25.8 %
12	8.7665 ms	10.3208 ms	17.7 %
5	9.0706 ms	4.574 ms	-49.6 %
1	-	-	-
2	-	-	-
4	-	-	-
7	-	-	-
8	-	-	-
10	-	-	-

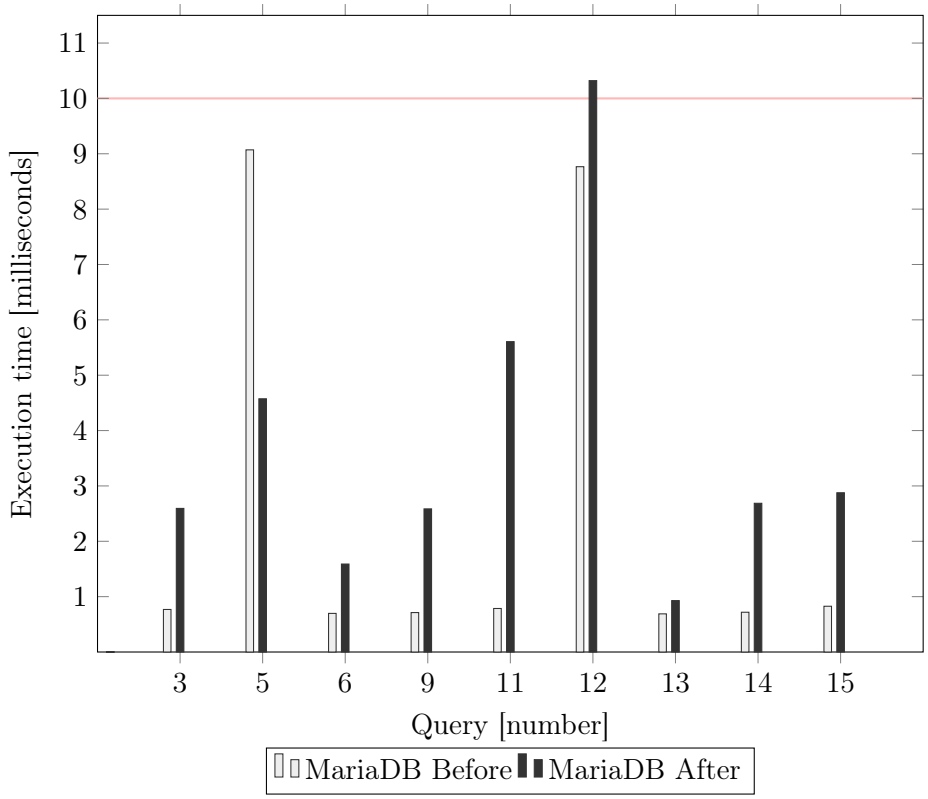

Figure 5.2: The before and after results of MariaDB.

5.4.3 Neo4j Scalability

A table displaying the normal execution times from the Neo4j benchmarking for the most executed queries, along with the execution time when increasing the data by a factor three. Additionally, the difference between these two execution times have been calculated and divided by the old execution time and are presented as percentages. The executed queries are lined up from fastest to slowest. A couple of the queries were not able to be run for assorted reasons, these are therefore marked with "-". An overview of the execution times for each query is shown in table 5.8 and a visual overview is presented in figure 5.3.

Table 5.8: An overview of Neo4j's execution times.

#	*Execution time*	*Execution time (3x data)*	*Difference*
5	3 ms	51 ms	1600 %
15	8 ms	126 ms	1475.0 %
6	46 ms	136 ms	195.7 %
11	217 ms	257 ms	18.4 %
3	254 ms	298 ms	17.3 %
14	117 ms	129 ms	10.3 %
13	145 ms	150 ms	3.4 %
9	140 ms	132 ms	-6.1 %
10	328 ms	294 ms	-10.4 %
1	-	-	-
2	-	-	-
4	-	-	-
7	-	-	-
8	-	-	-
12	-	-	-

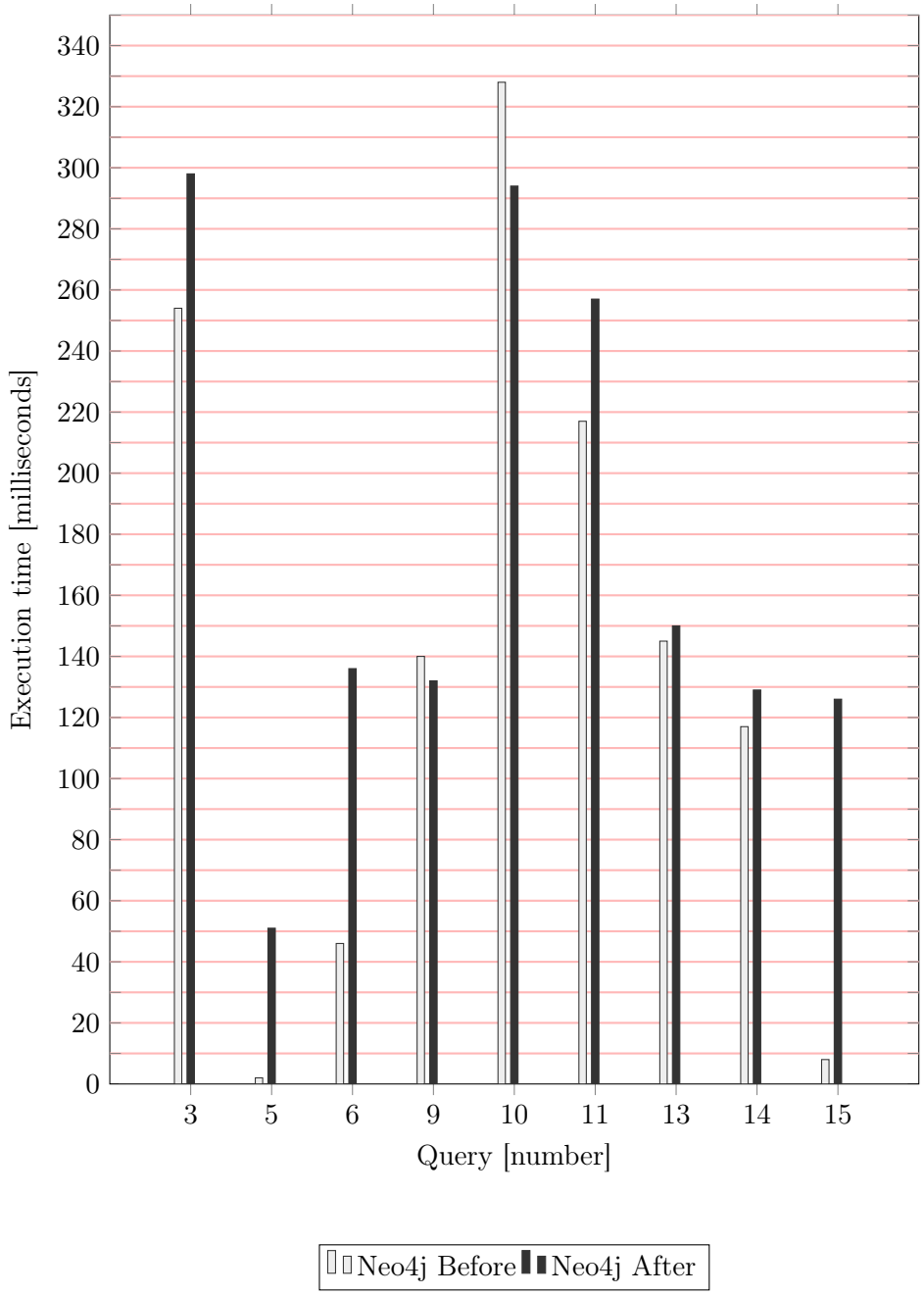

Figure 5.3: The before and after results of Neo4j.

5.4.4 Postgres Scalability

A table displaying the normal execution times from Postgres' benchmarking for the most executed queries, along with the execution time when increasing the data by a factor three. Additionally, the differences between these two execution times have been calculated and divided by the old execution time and are presented as percentages. The executed queries are lined up from fastest to slowest. A couple of the queries were not able to be run for assorted reasons, these are therefore marked with "-". An overview of the execution times for each query is shown in table 5.9 and a visual overview is presented in figure 5.4.

Table 5.9: An overview of Postgres' execution times.

#	*Execution time*	*Execution time (3x data)*	*Difference*
3	0.497 ms	14.947 ms	2907.4 %
5	0.360 ms	7.923 ms	2100.8 %
13	0.171 ms	1.280 ms	648.5 %
4	0.792 ms	2.811 ms	254.9 %
11	0.195 ms	0.625 ms	220.5 %
14	0.320 ms	0.529 ms	65.3 %
10	0.208 ms	0.315 ms	51.4 %
6	0.103 ms	0.117 ms	13.6 %
15	0.051 ms	0.055 ms	7.8 %
9	0.175 ms	0.188 ms	7.4 %
12	0.163 ms	0.042 ms	-74.2 %
1	-	-	-
2	-	-	-
7	-	-	-
8	-	-	-

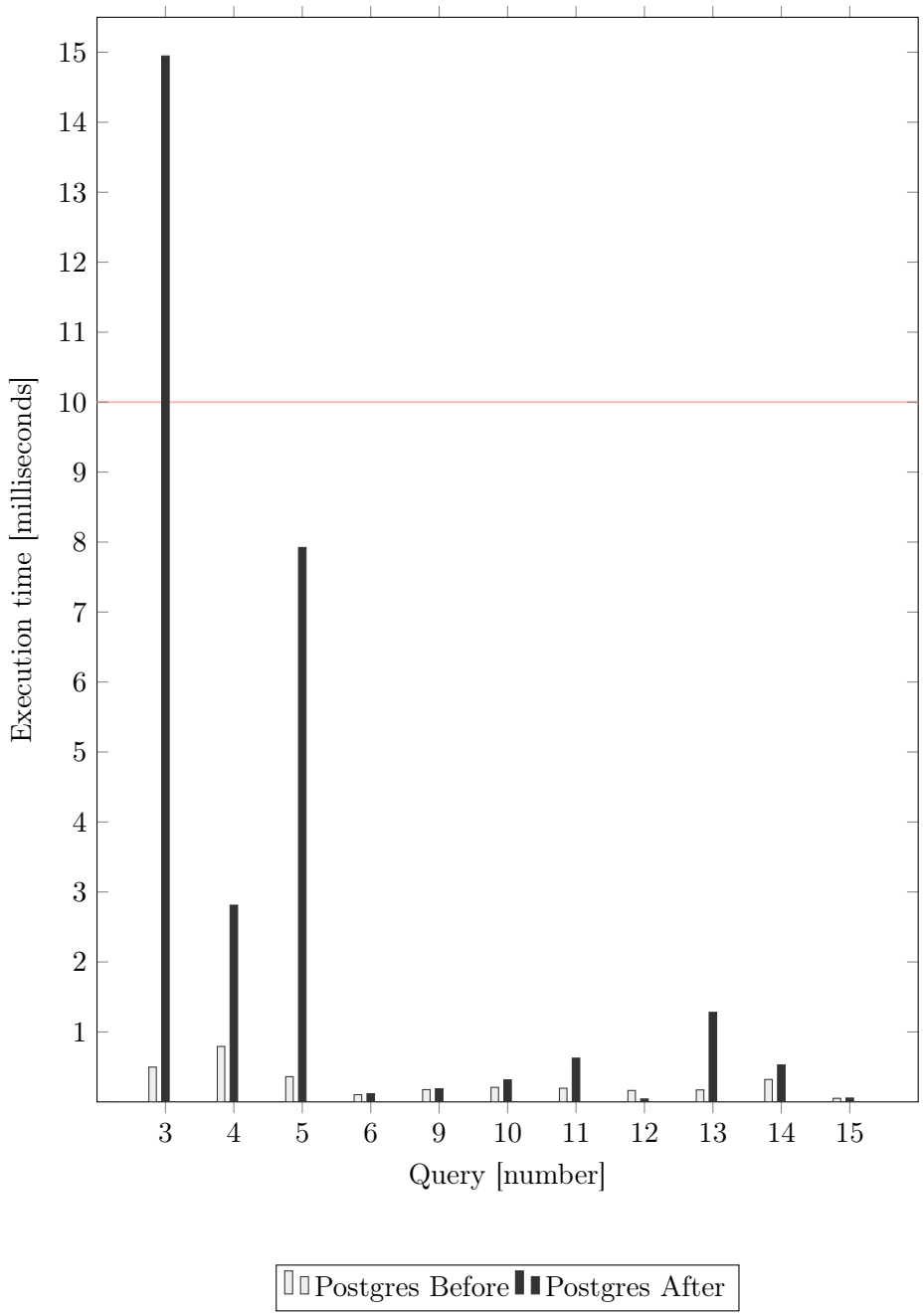

Figure 5.4: The before and after results of Postgres.

5.5 Licensing

In this section, the licensing costs are compiled. For each DBMS, the most fitting licensing has been chosen, i.e., the Enterprise Licensing. The reasoning behind the choice of this kind of license is due to the fact that the thesis has been made in cooperation with IKEA IT. Considering IKEA is a big company, the Enterprise Licensing was decided to be the best fit; that is if an enterprise license actually exists in each one of the DBMS. For more information about the basis for choosing the Enterprise license, see chapter 4.1.4. A concise summary of the chosen license and the licensing costs can be viewed in table 5.5, and an explanation of what is included in the license is given below.

Table 5.10: An overview of the most fitting licenses.

DBMS	*License name*	*Link*
MariaDB	MariaDB Enterprise	mariadb.com/pricing/
MongoDB	MongoDB Enterprise Advanced	mongodb.com/pricing
Neo4j	Neo4j Enterprise Edition: 1. Neo4j Commercial License 2. Neo4j Developer License 3. Neo4j Evaluation License	neo4j.com/licensing/
Postgres	-	postgresql.org/about/licence/

5.5.1 MongoDB

MongoDB offers only one type of enterprise licensing, called Enterprise Advanced. The service is described as "a collection of products and services that drive security, efficiency, and put you in control of your MongoDB databases" [28]. This licensing consists of three segments:

1. *Database management*
2. *Data and Business Protection*
3. *Support and services*

Database management includes automation, monitoring, optimization and backups. *Data and Business Protection* include auditing, encryption, authentication and commercial license. Finally, *Support and services* include technical support and additional MongoDB services.

An across-the-board cost for the MongoDB enterprise licensing is not printed on the website. Instead, they encourage those who are interested in buying the enterprise license to request a quote.

5.5.2 MariaDB

There are several features included when taking the decision of incorporating MariaDB with the database of one's choice, especially when it comes to the MariaDB Enterprise License. The differences that come with choosing this license over the Community License include getting better support, e.g. The support that is included in this one and not in the Community version is technical support, alerts when it comes to security, bug fixes, among others [31]. Besides these features, one other big component that is included in this license is the actual Enterprise server. These include both row- and columnar storage, object storage, backups, clusters, audits, etcetera [31]. The pricing for this license varies and can only be found out when requesting a quote through the MariaDB website.

5.5.3 Neo4j

The Enterprise Licensing scheme Neo4j offers consists of the following three licenses:

- Neo4j Commercial License.
- Neo4j Developer License.
- Neo4j Evaluation License.

With the use of the Enterprise License, the size restriction when it comes to the graphs is that it has no upper boundary. Besides the absence of size limitations, support is also included in this type of licensing [32]. The Commercial License is essentially the full version of the Neo4j Enterprise License. Due to this, all of the features are included in the Commercial License, including the backup being updated on an hourly basis, role-based access control and an one hour customer support response [33].

Neo4j Developer License is free when creating an account through the registration on their website [32]. Additionally, receiving a development license makes it possible to use Neo4j Enterprise on one's own desktop [32].

Neo4j Evaluation License consists of the full Enterprise License, but the license is only valid during a trial period [32]. Besides this, one can also receive help from Neo4j when needed with the use of this license [32].

The price of the Enterprise Licensing can differ depending on if the applications are of high or medium scale, as well as if they are cloud based or hosted on a local machine [33]. When it comes to medium scale on cloud-based databases, the price starts at $65 per month [33], while there is not a specified price when it is concerning high scaled. Self-hosting does not have a specified price on the website either. Meaning, in order for one to find out the price they would have to contact Neo4j directly for a price assessment.

5.5.4 Postgres

As stated on the Postgres website, Postgres is released under the PostgreSQL License. The PostgreSQL license is described as "a liberal Open Source license, similar to the BSD or MIT licenses" [29]. This means that the Postgres software, along with its documentation, is completely free to use, copy, modify and distribute [30]. It must be emphasized that the use of Postgres as a DBMS does not under any circumstances entail any licensing costs for IKEA.

Chapter 6
Conclusion

In the following chapter the conclusion of the thesis is presented. The end results have been summarized with answers to the questions stated in chapter one along with ethical aspects and development opportunities for the work in this thesis to be continued in the future.

6.1 Discounting trends

What are the trends regarding discounts, both at IKEA and in the retail industry in general?

As explained in chapter 2.1.1, discount trends tend to fluctuate, and they can change overnight. The companies that have been exemplified in this thesis are Foodora and Mio. Both different kinds of vouchers companies, along with IKEA, can be viewed as companies that are always trying to incorporate current trends into their selling scheme. With Foodora, various kinds of vouchers are usually offered to their customers. Vouchers including those where a percentage is deducted from the price of an order after the invitation of friends. When it comes to Mio, which is another big furniture company in Sweden, the discounts that exist within this company are usually in the shape of heavily discounted furniture. Some of these discounts can be as high as 50 percent which usually occurs during specific time-limited periods a number of times a year.

Lastly, the trends regarding discounts at IKEA consist of many different ones. These trends include different discount strategies such as Family rewards that can be obtained after becoming a member and receiving a IKEA Family Card, through vouchers that can be either digital or physical. Furthermore, personalized coupons and markdowns are also included. Markdowns are typically characterized by implementing reductions on certain articles. In addition, diverse kinds of discount types are being used at IKEA as well. These consist of amount-based discounts, combined discounts, quantity discounts and tiered discounts, along with various bundles.

6.2 Data handling at IKEA

The conclusion after researching and conducting interviews to learn about how the discount data is being handled at IKEA, i.e., how it is being used and stored, are presented in the following section.

6.2.1 Usage of discount data

How is discount data being used at IKEA today?

The usage of the discount data was uncovered through the help of the software engineer who was interviewed in one of the interviews. This help enabled the possibility of receiving the most used queries, which are listed in Appendix B, being executed in the production database for storing discount data at IKEA. Several of the most used queries consisted of non-complex ones, such as finding out all the discounts or coupons belonging to a certain ID but also finding out the time zone for a specific country, among others. The conclusion that can be drawn from this is that the discount data that is used by IKEA today is not for the most part used in order to find out the various discounts, but to find out other aspects that have to do with these discounts.

6.2.2 Storage of discount data

How is discount data being stored at IKEA today?

As emerged in the interview with the software engineer (interviewee #2), discount data at IKEA is stored in a Postgres database hosted on Google Cloud Platform. Due to Postgres being a relational database, data is stored as records, or rows, inside a schema. The database in question is made up of several schemas, closely connected through the use of primary and foreign keys.

6.3 What makes a DBMS good for storing discounts

What are the advantages and disadvantages associated with different database technologies on a general level? What criteria are to be considered useful in a database technology handling discounts?

The advantages and disadvantages associated with the different DBMSs on a general level differ depending on the DBMS itself and the context in which it is used.

There are several criteria considered useful in a database technology handling discount. The most important ones are brought up in this section, along with reasons behind why they are to be considered useful.

6.3.1 Big Community

It is important that a DBMS for storing discounts at IKEA is backed up by a big community. A big community facilitates the work of storing and using the discount data as of today in several ways. It makes it easier to find competent people as in people who master the technology, and it also makes the work easier since it should be easier to find information and useful solutions when encountering problems if the technology itself is prevalent.

As established in the interview with the software engineer at IKEA, the IT systems at IKEA are made up of smaller systems that are integrated with one another into a big ecosystem. A bigger community should make this integration smoother, since a more prevalent DBMS should have more ready-made solutions than a less prevalent one. Also, it is of importance the DBMS does not get outdated and that the technology itself keeps getting used by a large number of people. It is important the DBMS keeps getting updated and patches keep getting addressed.

6.3.2 Fast responses (low latency)

As the interviewed software engineer at IKEA explained, time is essential when it comes to discount information. When a coupon has been used, for instance, it must die instantly. If not, the coupon can be used several times which is something that is not desired. Therefore, it is important that the system responds as fast as possible, allowing the information to get updated or fetched quickly.

6.3.3 Cost Effectiveness

There is an economical aspect to what makes a database technology useful. Since a big community enables the software engineers to work more efficiently, a DBMS being backed up by a big community should be more cost effective, at least regarding resources such as workforce.

Other than workforce, there are two major categories of resources that are interesting to look into from an economical perspective - licensing and hardware. For big companies like IKEA, an enterprise licensing is needed and as concluded in section 5.5.4, such a licensing is only without cost for Postgres and none of the other DBMS mentioned in this thesis.

6.4 Benchmarking results

Due to there not being a definite execution time to be derived from the MongoDB testing database, the benchmarking result is inconclusive. What can be established, however, is that Postgres is faster, i.e., has a lower extent of latency, than

MariaDB. It is not possible to draw any conclusions about how fast MongoDB is in relation to MariaDB and Postgres other than when observing the result for query #5 where both MongoDB and MariaDB prove to be a lot slower than Postgres. Moreover, the query that had the best performance in Neo4j was the query that was the slowest in the others, query #5, while query #10 was the slowest. Due to MongoDB not providing the exact duration for an execution, there is not enough basis for an overall picture.

There could be at least two reasons for why query #5 is exceptionally slow for MongoDB and MariaDB, that query #10 was the slowest in Neo4j and that query #12 is exceptionally slow for MariaDB. The easiest explanation is probably that MariaDB and MongoDB is much slower than Postgres. But if that is the case, does this only show when executing these queries? Another possibility is that something failed in the setup of the MongoDB and MariaDB databases, possibly some indexing that got lost.

6.5 Performance

How does the four different database technologies perform in relation to the criteria answered previously?

Based on what was mentioned in the previous subsection about what makes a DBMS good for storing discounts, the performance of the four different technologies in relation to these criteria vary depending on the DBMS in question. When it comes to having a big community, all these DBMSs fulfil this criterion. All of them are among the most popular ones in their own technology and the ones that are the most frequently used on a general DBMS level. As a result, a big community exists within all of them with loads of forum threads that are there to help if any kind of problem would occur. Furthermore, speaking about having a low latency and fast responses really depends on the expectation. All these DBMSs are pretty fast, generally speaking, but comparing all of them to each other reflects a completely different aspect as there are clear winners when it comes to this criterion. The same can be said about the economical aspect to it. Depending on what each DBMS costs with respect to what features are offered in the actual costs can have major impacts when it comes to deciding which ones are the most worthwhile ones. As mentioned in 5.5.4, Postgres is the only DBMS that does not have a cost at all, making it the clear winner regarding this.

6.5.1 Signal-to-Noise Ratio Reflections

As concluded in the result chapter, the Neo4j database should have a higher SNR than the relational databases. It was not part of the result, but the thesis workers noted that the query did not run faster after appropriately indexing the node type. A high signal-to-noise ratio is strongly connected to the performance. Therefore,

a high SNR may be the reason for the Neo4j database to be many times slower than all the relational databases that had been tested.

In Neo4j's defence, the test was not fair. A fairer experiment would have been to set up a database that utilizes the advantages of the graph technology. On the other hand, for the test to be completely fair, all the other databases should in that case undergo corresponding optimization, if there is room for it. Optimization was not within the scope of this thesis and was not included for that reason.

6.5.2 Scalability results

The end results when it comes to the scalability metric, which are shown in section 5.4, could arguably be inconclusive. As shown in 5.4.1, which is concerning the scalability of MongoDB, the percentage differences were only able to be calculated in two instances: for query #3 and #11. With only having two values, it is not possible to give a clear picture of how MongoDB performs in general, which is the reason it could be said that the results are inconclusive. In the case of the few queries that were able to be estimated and presented, the differences show remarkably high percentages: >200%. Meaning, based on only these two queries, MongoDB does not handle increments very well when it comes to increasing the amount of data points. In comparison to the other DBMSs, Postgres also has query #3 as the one with the biggest difference with a percentage of 29074.4% while MariaDB has query #11 as their top query that showed the worst results. Furthermore, Neo4j had query #5 as the one with the worst result while Postgres had it as their second worst one while MariaDB had the same query in last place, as a difference below zero. Meaning, it went faster with this query after the increment of data points compared to the execution of the fewer data points. The reason for why this happened can be due to several factors, one could be that it showed a result that was too high on the first execution with the fewer data points. There is a possibility for this to happen when there are other programs working on the computer simultaneously. The same happened to both Neo4j and Postgres on one and two queries respectively and the same can be said about them when it comes to this.

In conclusion, all the DBMSs have mixed results when it comes to their ability of being able to handle increments and decrements well. Some queries ended up getting better results than others with various patterns being able to be detected when it came to a couple of the queries, making it seem like some of these were executed with something happening in the background. In order to improve this, what could have been able to be incorporated would have been to execute all of the queries the same number of times and to thereafter calculate the mean values. Additionally, these queries could have been executed on several different days to minimize the possibility of something else running in the background during these executions. A software for reducing interference from other programs running on the computer could also have been used.

6.5.3 Performance Demanding Queries

As mentioned in section 5.2.5, query #5 was the most performance demanding query in two out of four DBMSs and on the less demanding side in Postgres. In Neo4j, by contrast, it was the least performance demanding query. The conclusion that can be drawn by this circumstance is that, considering this query is an UPDATE statement, it is highly likely that MariaDB, MongoDB and Postgres are slower when it comes to this kind of statement in comparison to the SELECT statements. The opposite can be said about Neo4j which seems to handle UPDATE statements better than SELECT statements. Since the databases in all the DBMSs were indexed, it is likely to assume that some of these could have been more performance demanding if they would have not had the indexes they had. Especially considering some of the queries were executed on tables that had many more data points than the rest. Query #15 is an example of this, where the table coupon_campaign_t contains approximately 30 000 data points, making it the largest table of them all. Despite this, it performed the best in Postgres and among the best in Neo4j, in comparison to MariaDB where it performed among the worst. Without the indexing, it would be highly likely that the performance would have been worse since in that case it would not know where to look for the value and therefore traverse through the entire table until it found the correct one.

Furthermore, Neo4j was by far, the worst performing DBMS. The difference in execution time when it came to 80% of the queries were abnormally high. The reason for this is most likely due to it not being configured properly, hence not allowing it to show the correct execution times. Additionally, it showed the average execution times and since the first execution time was always extremely high it was therefore necessary to execute the same query multiple times to be able to present the fairest execution times. It can therefore be stated that the answers are inconclusive even on this one, in comparison to the other DBMSs. It is, however, possible to compare the queries in Neo4j and draw conclusions from that. In this case, the most performance demanding queries consisted of the ones where the AND operator was present. Indicating that Neo4j responds badly when such statements are included, assuming the differences in execution times are somewhat correct despite the execution times as a whole being most likely wrong.

6.6 Thesis Purpose Fulfillment

The conclusions that are derived from the results of both the initial questions that were presented in the beginning of the thesis in section 1.5 and the metrics determination that were presented in 2.4.1, have had major contributions to fulfilling the purpose of this thesis. The main purpose was to investigate and find out if there is a database technology that is better suited for storing discounts than the one that is currently in use. With the help of the initial questions and the various metrics, the results have been helpful in determining this. Incorporating the determined metrics, enabled the execution of the benchmarking. Moreover, it contributed to the knowledge of what to explore in an effort to discover the correct answers to fulfill the purpose. In addition, the questions were helpful due to it

being possible to move forward with quantitatively measuring each metric thanks to the information that came from the answers to the questions. One instance that shows this is the handling of data at IKEA, how it is being used and stored. After finding out this information, it was possible to determine the metrics and execute the actual benchmarking. The results indicated that Postgres, which is the one that is currently being used when it comes to storing discounts, is the one that came out on top when it came to most of the metrics. In this case, the results can be used by not transferring to a different database technology, but to keep using the existing one. This was also supported by the software engineer who in the interview stated that Postgres worked well and did not see a reason for them to switch to a different database technology.

6.7 Ethical aspects

The ethical aspects regarding the subject of this thesis have been reflected on and are presented in the following section.

6.7.1 Interviewee anonymity

For ethical purposes, certain measures have been taken to keep the anonymity of the interviewees. Before acting, a definition of what is an acceptable minimum level of anonymity had to be decided. The thesis workers define it as acceptable if people outside the company should not be able to identify the persons being interviewed.

To uphold this level of anonymity, naturally, none of the interviewees are named in this report. Information about gender and other attributes that enable identification are left out. Another way of anonymizing the interviewees is to not attach a transcription of the interviews, but to summarize the most important parts of the discussions and leave out what is irrelevant or unnecessarily disclosing.

6.7.2 The effect of discounts

Discounts on products gives the customers an incentive and makes their desire for buying these products even greater. Thus, their incline for impulse purchases can become higher, leading them to buy products they in fact do not actually need or really want. This can lead to negative implications in their private lives, where buying discounted items becomes a vicious circle that keeps them purchasing products for no reason and possibly for money they do not have, leading them into debt. Even though the example just described is of the more severe type, it does not take away from the fact that discounts can be damaging to people who are more susceptible to that kind of selling scheme. Even though it is not something that happens to all people or affect them in negative ways, the effects of discounts are something of importance that should be kept in one's mind.

Moreover, one of the many visions that exists within IKEA is the incorporation of a so-called democratic design. What is considered a democratic design by IKEA

is when a product fulfils the following five criteria [47]:

- Function
- Form
- Quality
- Sustainability
- Low price

There are thus multiple criteria that every product must fulfil, and therefore there has to be some kind of balance where some criteria are more important than others, in order to achieve all of them. One product that is the prime example of this balance is the FLISAT desk that is designed for children. It is sold for a decent price and the height of it can be adjusted and it can therefore continue to be used by the child for many years ahead [47]. Additionally, it is made of wood that is renewable but also sustainable, hence making it the perfect example of a democratic design [47]. Even though the severe consequences of discounts that was mentioned in the previous paragraph has a possibility of happening to individual people, the vision IKEA has can still give some additional advantages.

6.8 Development opportunities

The developmental opportunities for investigating the best DBMS for discounts are many. One of these opportunities is to conduct testing for additional metrics. In the initial stages of this thesis, two metrics were formulated that were down-prioritized and unfortunately did not fit into the time scope. These metrics are:

3. Signal-to-noise ratio
    - Determine for every DBMS (see chapter 2.3) respectively the amount of data that must be processed to perform the most commonly used queries (create, read, update, delete).
5. Memory
    - Identify the amount of memory each DBMS (see chapter 2.3) requires to store the database.
    - Establish how much cache memory is used by each DBMS (see chapter 2.3) in the CPU.
    - Identify how much RAM, i.e., working memory, is used by each DBMS (see chapter 2.3) when performing the most commonly used queries executed in the currently in use Postgres database for storing discounts at IKEA.
    - Establish for each DBMS (see chapter 2.3) which of the queries identified as being most used in the current database require the most RAM.

Another way of investigating further is to conduct the same type of testing for different DBMSs belonging to the same technology, preferably belonging to the technology from which the DBMS that is deemed the best fit in this thesis belongs to, to find the best fit DBMS and not only the best fit technology.

For an even better evaluation and to better answer the question of the best fit database technology for discounts, the same testing that had been done in this thesis could be conducted but with several DBMS alternatives competing for the same technology. This would ensure a better representation of each technology.

Lastly, this thesis is restricted to determine the best fit database technology for discounts from only four database technologies. The best fit database technology might not be one of these four, but some other technology that was not evaluated in this thesis. Therefore, one development opportunity is to broaden the search for the best fit database technology.

Chapter 7
Terminology

DDL	DDL, which stands for *Data Definition Language*, is defined by IBM as "a language for describing data and its relationships in a database" [49].
NLP	NLP, which stands for *Natural Language Processing*, is defined as the following by IBM as "[...] the branch of artificial intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can" [50].
ORDBMS	Object-relational database management system
RDBMS	Relational database management system
TPS	TPS, which stands for *Transactions per second*, is the number of transactions that occur each second.

References

Articles

[1] Cl.cam.ac.uk. 2022. *A Gentle Introduction to Relational and Object Oriented Databases*. [online] Available at: <https://www.cl.cam.ac.uk/ fms27/db/tr-98-2.pdf> [Accessed 1 February 2022].

[2] Tecmint.com. 2020. *What is MariaDB? How Does MariaDB Work?*. [online] Available at: <https://www.tecmint.com/what-is-mariadb-how-does-mariadb-work/#:~:text=Just%20like% 20MySQL%2C%20MariaDB%20also ,can%20be%20on%20different%20hosts> [Accessed 2 February 2022].

[3] Ieeexplore-ieee-org.ludwig.lub.lu.se. 2021. *Designing User Interface with UML and CRUD Concept for IoT-Based Water Quality Analysis Tool*. [online] Available at: <https://ieeexplore-ieee-org.ludwig.lub.lu.se/stamp/stamp.jsp?tp=arnumber=9650608> [Accessed 3 February 2022].

[4] Medium.com. 2018 *Meet the Query Log Analyzer - Analyzing Neo4j Query Log files on your Neo4j Desktop*. Available at: <https://medium.com/neo4j/meet-the-query-log-analyzer-30b3eb4b1d6> [Accessed 10 May 2022].

[5] C. Vicknair, M. Macias, Z. Zhao, X. Nan, Y. Chen, and D. Wilkins, "A comparison of a graph database and a relational database," *Proceedings of the 48th Annual Southeast Regional Conference on - ACM SE '10*, 2010.

Books

[6] E. Plugge, P. Membrey, T. Hawkins *The Definitive Guide to MongoDB: The NoSQL Database for Cloud and Desktop Computing* New York: Springer-Verlag, 2010.

[7] J. Gray, A. Reuter *Transaction processing: Concepts and techniques*. San Mateo: Morgan Kaufmann, 1993.

[8] R. J.T Dyer *Learning MySQL and MariaDB: Heading in the Right Direction with MySQL and MariaDB*. Newton: O'Reilly Media, Inc, 2014.

[9] D. Bartholomew *Getting Started with MariaDB*. Birmingham: Packt Publishing, 2013.

[10] F. Razzoli *Mastering MariaDB*. Birmingham: Packt Publishing, 2014.

[11] Teorey, T., Lightstone, S., Nadeau, T. and Jagadish, H., n.d. *Database Modeling and Design*, 4th Edition.

[12] A. Lantz *Intervjumetodik*. Lund: Studentlitteratur, 2013.

[13] T. Stapenhurst *The benchmarking book*. Amsterdam: Elsevier, 2009.

[14] K. Alexanderson *Källkritik på Internet: .SE:s Internetguide, nr 25*. Ödeshög: DanagårdsLiTHO, 2012.

Websites

[15] Oracle.com. 2022. *What is a Relational Database (RDBMS)?*. [online] Available at: <https://www.oracle.com/database/what-is-a-relational-database/> [Accessed 8 February 2022].

[16] Britannica.com. 2022. *IKEA Swedish company*. [online] Available at: <https://www.britannica.com/topic/IKEA> [Accessed 8 February 2022].

[17] Ingka.com. 2022. *Ingka Group governance*. [online] Available at: <https://www.ingka.com/this-is-ingka-group/how-we-are-organised/> [Accessed 8 February 2022].

[18] Inter.ikea.com. 2022. *One brand, many companies – the IKEA franchise system*. [online] Available at: <https://www.inter.ikea.com/en/this-is-inter-ikea-group/the-ikea-franchise-system> [Accessed 8 February 2022].

[19] DB-Engines. 2022. *DB-Engines Ranking*. [online] Available at: <https://db-engines.com/en/ranking/graph+dbms> [Accessed 1 February 2022].

[20] Neo4j Graph Database Platform. 2022. *What is a Graph Database? - Developer Guides*. [online] Available at: <https://neo4j.com/developer/graph-database/> [Accessed 1 February 2022].

[21] Neo4j Graph Database Platform. 2022. Concepts: Relational to Graph - Developer Guides. [online] Available at: <https://neo4j.com/developer/graph-db-vs-rdbms/> [Accessed 1 February 2022].

[22] Postgresql.org. 2022. *PostgreSQL: About*. [online] Available at: <https://www.postgresql.org/about> [Accessed 1 February 2022].

[23] Mongodb.com. 2022. *What is MongoDB?*. [online] Available at: <https://www.mongodb.com/what-is-mongodb> [Accessed 1 February 2022].

[24] Ibm.com. 2021. *ACID properties of transactions*. [online] Available at: <https://www.ibm.com/docs/en/cics-ts/5.4?topic=processing-acid-properties-transactions> [Accessed 31 January 2022].

[25] Ibm.com. 2021. *What is a database management system?*. [online] Available at: <https://www.ibm.com/docs/en/zos-basic-skills?topic=zos-what-is-database-management-system> [Accessed 31 January 2022].

[26] Dictionary.cambridge.org. 2022. *Meaning of discount in English*. [online] Available at: <https://dictionary.cambridge.org/dictionary/english/discount> [Accessed 15 February 2022].

[27] Ikea.com. 2022. *IKEA Family*. [online] Available at: <https://www.ikea.com/us/en/ikea-family/> [Accessed 16 February 2022].

[28] MongoDB. 2022. *MongoDB Enterprise Advanced*. [online] Available at: <https://www.mongodb.com/products/mongodb-enterprise-advanced> [Accessed 15 April 2022].

[29] Postgresql.com. 2022. *License*. [online] Available at: <https://www.postgresql.org/about/licence> [Accessed 18 April 2022].

[30] Opensource.org. 2022. *The PostgreSQL License (PostgreSQL)*. [online] Available at: <https://opensource.org/licenses/postgresql> [Accessed 18 April 2022].

[31] Mariadb.com. 2022. *Pricing*. [online] Available at: <https://mariadb.com/pricing/> [Accessed 18 April 2022].

[32] Neo4j.com. 2022. *Neo4j Licensing*. [online] Available at: <https://neo4j.com/licensing/> [Accessed 18 April 2022].

[33] Neo4j.com. 2022. *Neo4j Licensing*. [online] Available at: <https://neo4j.com/pricing/> [Accessed 18 April 2022].

[34] Mio.se. 2022. *Mio*. [online] Available at: <https://www.mio.se/> [Accessed 27 April 2022].

[35] Mio.se. 2022. *Mio - Soffor, Möbler och Inredning | Mio*. [online] Available at: <https://www.mio.se/> [Accessed 27 April 2022].

[36] Foodora.com. 2022. *Home | Delivery Hero*. [online] Available at: <https://www.foodora.com/> [Accessed 27 April 2022].

[37] Docs.microsoft.com. 2022. *User mode and kernel mode - Windows drivers*. [online] Available at: <https://docs.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/user-mode-and-kernel-mode> [Accessed 27 April 2022].

[38] Postgres.org. 2022. *20.1. Setting Parameters*. [online] Available at: <https://www.postgresql.org/docs/current/config-setting.html> [Accessed 27 April 2022].

[39] Dbeaver.io. 2022. *DBeaver Community | Free Universal Database Tool*. [online] Available at: <https://dbeaver.io/> [Accessed 3 May 2022].

[40] Dictionary.cambridge.org. 2022. *Meaning of latency in English*. [online] Available at: <https://dictionary.cambridge.org/dictionary/english/latency> [Accessed 6 May 2022].

[41] Mariadb.com. 2022. *SHOW PROFILE*. [online] Available at: <https://mariadb.com/kb/en/show-profile/> [Accessed 6 May 2022].

[42] Mongodb.com. 2022. *Explain Results*. [online] Available at: <https://www.mongodb.com/docs/manual/reference/explain-results/> [Accessed 6 May 2022].

[43] Mongodb.com. 2022. *db.setProfilingLevel()*. [online] Available at: <https://www.mongodb.com/docs/manual/reference/method/db.setProfilingLevel/> [Accessed 6 May 2022].

[44] Dev.mysql.com. 2022 *MySQL :: MySQL EXPLAIN ANALYZE*. Available at: <https://dev.mysql.com/blog-archive/mysql-explain-analyze/> [Accessed 8 May 2022].

[45] Community.neo4j.com. 2020 *Query Log Analyzer 1.0.2 is released*. Available at: <https://community.neo4j.com/t/query-log-analyzer-1-0-2-is-released/30752> [Accessed 10 May 2022].

[46] Neo4j.com. 2022 *How-To: Import CSV Data with Neo4j Desktop*. Available at: <https://neo4j.com/developer/desktop-csv-import/> [Accessed 10 May 2022].

[47] About.ikea.com. 2022 *Democratic Design. Making great design available to everyone*. Available at: <https://about.ikea.com/en/life-at-home/how-we-work/democratic-design> [Accessed 13 May 2022].

[48] Neo4j.com. 2022 *Concepts: Relational to Graph - Getting Started*. Available at: <https://neo4j.com/docs/getting-started/current/graphdb-vs-rdbms/> [Accessed 16 May 2022].

[49] IBM.com. 2022 *Generating DDL scripts*. Available at: <https://www.ibm.com/docs/en/radfws/9.6.1?topic=scripts-generating-ddl> [Accessed 18 May 2022].

[50] IBM.com. 2022 *What is Natural Language Processing?*. Available at: <https://www.ibm.com/cloud/learn/natural-language-processing> [Accessed 18 May 2022].

Appendix A

Interview Questions

A.1 Respondent 1

1. What is your job title?
2. What does your job entail?
3. What do the most common discounts look like?
4. Are the discounts incorporated differently depending on the department?
5. What are the current trends regarding discounts?
6. Do the discounts differ online compared to the ones in the stores?
7. How does IKEA differ in comparison to other retail stores?
8. How often do the discounts get updated?
9. How are the environmental values incorporated in the discounts?
10. What are the challenges when it comes to discounts?
11. What kind of tools are used when searching for different discounts?
12. When searching for discounts, which is the most common one?
13. How can we get access to a snapshot of currently available discounts globally?

A.2 Respondent 2

1. What is your job title?
2. What does your job entail?
3. What tools are being used in regards to the databases?
4. How do you use the data?
5. How is the SDM typically used?
6. Could you tell us a bit about the algorithm that calculates discounts?

7. Are there any problems when it comes to the currently used database management system? Do the searches take a long time for instance?
8. Have you explored or considered any other DBMS's than the relational?
9. What is the most common extraction of data that is performed?
10. How often is data inserted or deleted in the database?
11. How have you tried to optimize?
12. What do the most commonly used searches or reads look like?
13. What are the challenges concerning the database, the DBMS and the work in general?
14. Do you think you would benefit from using another database technology, like graph DB?

Appendix B
Queries

#	Query
1	SELECT *CONVERT_FROM*(*decrypt*(*CAST*(email_encrypted AS bytea), *CAST*(:passCode AS bytea), 'aes'), 'utf-8') AS emailEncrypted FROM auth.user WHERE email = :email;
2	SELECT id, email_id_hash, *CONVERT_FROM*(*decrypt*(*CAST*(email_id_encrypted AS bytea), *CAST*(:passCode AS bytea), 'aes'), 'utf-8') AS email_id_encrypted, insert_dtime FROM auth.user_t WHERE email_id_hash = ?;
3	SELECT * FROM auth.user_role AS userRole WHERE userRole.user_id = :userId AND userRole.active = true;
4	SELECT (*ARRAY_AGG*(country.cty_name ORDER BY CasE lang_code_iso WHEN :langCode THEN 1 WHEN 'en' THEN 2 ELSE 3 END))[1] AS data, cty_code AS value FROM (SELECT lang_code_iso, cty_code, cty_name FROM common.cty_lang_t WHERE cty_code IN (:authorizedCountries) ORDER BY cty_code) country GROUP BY country.cty_code;
5	UPDATE auth.user set last_login_dtime=:currentDateTime WHERE email=:emailIdHash;
6	SELECT discount_id FROM coupon.coupon_campaign_t WHERE id = :campaignId AND status_code=30 AND type='PRIVATE';

7	SELECT ct.code, ct.schedule ->> 'startDate' AS startDate, ct.schedule ->> 'endDate' AS endDate FROM coupon.coupon_campaign_t cct JOIN coupon.coupon_t ct on cct.id = ct.campaign_id AND cct.schedule ->> 'endDate' >= :endDateValue WHERE cct.id = :campaignId AND cct.status_code = 30 AND cct.type = 'PRIVATE' AND ct.status_code = 30 AND ct.remaining_redemption_limit > 0 AND ct.schedule ->> 'endDate' >= :endDateValue;
8	SELECT ct.code, ct.schedule ->> 'startDate' AS startDate, ct.schedule ->> 'endDate' AS endDate FROM coupon.coupon_campaign_t cct JOIN coupon.coupon_t ct on cct.id = ct.campaign_id WHERE cct.id = :campaignId AND ct.code in (:couponCodes);
9	SELECT * FROM auth.user_preference_t WHERE user_id = :id;
10	SELECT * FROM auth.user WHERE active = true AND email = :email AND delete_dtime is null;
11	SELECT * FROM common.country_t AS country WHERE country.cty_code = :countryCode AND country.active=true;
12	SELECT * FROM discount.discount_t WHERE id = :id;
13	SELECT timezone FROM common.TIMEZONE_T WHERE LOCALE = ?;
14	SELECT parent_bu_code FROM common.business_unit_t WHERE bu_code= :buCode limit 1;
15	SELECT * FROM coupon.coupon_campaign_t WHERE id = :id;

Table B.1: The most commonly executed queries compiled in a list with no specific order or prioritization. Each query is numbered for ease of referencing.

Printed by Tryckeriet i E-huset, Lund 2022

LUND
UNIVERSITY

Series of Bachelor´s theses
Department of Electrical and Information Technology
LU/LTH-EIT 2022-877
http://www.eit.lth.se