ANALYSIS ON THE PERFORMANCE COMPARISON BETWEEN A RELATIONAL DATABASE AND A NON-RELATIONAL DATABASE FOR BIG DATA APPLICATION

by

Md. Ahsanul Hasan
18101018
Fardin Ahmed
18101203
Abul Hossain Chowdhury
18101587
Md. Kamruzzaman Leion
18101417
Md. Rafsan Ferdous Prince
18101299

A thesis submitted to the Department of Computer Science and Engineering in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
January 2022

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Md. Ahsanul Hasan	Fardin Ahmed
18101018	18101203
Abul Hossain Chowdhury	Md. Kamruzzaman Leion
18101587	18101417

Md. Rafsan Ferdous Prince

18101299

Approval

The thesis titled "Analysis on the performance comparison between a Relational Database and a Non-Relational Database for Big Data Application" submitted by

1. Md. Ahsanul Hasan (18101018)
2. Fardin Ahmed (18101203)
3. Abul Hossain Chowdhury (18101587)
4. Md. Kamruzzaman Leion (18101417)
5. Md. Rafsan Ferdous Prince (18101299)

of Spring, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 20, 2022.

Examining Committee:

Supervisor:
(Member)

Hossain Arif

Assistant Professor
Department of Computer Science and Engineering
BRAC University

Program Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD

Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi

Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Ethics Statement

We, hereby declare that this thesis is based on the findings of our research. All other materials used have been properly noted in the document. This work, neither in whole nor in part, has never been submitted by anyone to any other university or institution for any degree.

Abstract

Database plays a vital role in modern life of technology. Large scale applications such as social media, E-commerce platforms generate a really big amount of data that needs to be stored and worked with properly. And it is important to get efficient performance from database management systems that are used for this purpose. So, in this thesis, we compared the performance between a Relational database management system (RDMS) and a Non-relational database management system. We used PostgreSQL as a relational database and MongoDB as a Non-relational database in JavaScript server side platform. And we generated a large amount of records with different numbers using JavaScript functionalities that needs to be processed and stored for scalable and optimized performance. Then, we performed select, update, delete, insert operations in different level of complexities in both the databases. Finally, we analyzed and compared the performance using statistical ANOVA process of these two databases. From our experiment in big data application, the update, delete and select operation was faster in PostgreSQL database and insert operation was faster in MongoDB database

Keywords: Big data; MongoDB; PostgreSQL; RDMS; Technology;

Dedication

We would like to dedicate this research work to our dearest supervisor Hossain Arif Sir. Beyond his absolute support and motivation, we could not have succeed this far.

Acknowledgement

First of all, all praise to the Great Allah for whom our thesis have been completed without any interruption. Now we would like to thank our honorable supervisor Hossain Arif sir for the support and guidance he provided during our research time. We would also like to thank sir for providing us with necessary resources during our research phase.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Relational model is the basis for traditional databases. They are called SQL databases. But in recent times, NoSQL databases are highly recommended as internet usage has increased dramatically which creates large amounts of data which are structurally different . Different applications create such types of data commonly known as big data [4]. Big data [5]is becoming more and more important to the modern world day by day. The modern databases are getting larger with every second which is creating problems.. Such problems are:

1. With this much rapidly growing data, organizations often find it difficult to store them properly.

2. The next problem is to choose the right tools and techniques to deal with this huge amount of data.

3. The last but not the least is securing the data with data model [7]. While everyone gives all their attention to managing and analyzing the data properly, often times no one remembers that the data also needs to be secured properly. Unprotected data can create a lot of problems.

Many sources have unraveled by stating NASA can just produce 1TB of data just observing the earth on a daily basis [2]. SQL databases are known as traditional databases which store data based on relational models compared to those of non-relational databases or NoSQL which has gained popularity [19].

SQL databases manages data in structured tables. Conventional relational database management system (RDBMS) has strict static schema and because of this, it is not flexible. SQL queries are declarative and expressive. RDBMS always tries to use the relational algebra to optimize executions of queries. SQL databases scale vertically. So, in order to increase load in a server system upgrades would be needed such as, more powerful CPU, more ram, faster and larger storage capacity. SQL databases follow ACID property. ACID stands for Atomicity, Consistency, Isolation and Durability [12]. Because, conventional relational database management systems (RDBMS) work using relations among different data tables, they can only work properly if the database is small, but the problem is nowadays databases are getting huge in size and volume day by day. Another problem is because of SQL's strict static schema and structured table based model, SQL cannot always meet the increasing growth of unstructured datasets. And because of these reasons, working with big datasets in RDBMS, can be a lot hard, resource heavy and tiring also for the whole system. SQL databases are known as traditional databases which store data based on relational models but non-relational databases or NoSQL have gained popularity [19]. NoSQL existed since late 1960s [4].

NoSQL term was introduced by Carlo Strozzi in 1998 and referred to non-relational database in 2009 by eric Evans. NoSQL database have been around since 1960s [4].NoSQL database is not a working tool rather it is a methodology containing several competing tools [8]. It offers a different mechanic to store and retrieve data. It can also handle unstructured data unlike relational database. Non-relational databases do not use RDBMS and also do not use complex storage of database. Rather they uses keys to find data based on keys attached to the data. NoSQL database becomes important to us where SQL counterpart can not do so. A conventional relational database management system (RDBMS) is not flexible because it has a static schema. NoSQL databases have different usage. For example, Redis provides useful way for changing data quickly. Cassandra is appropriate for real-time analysis operation, i.e. Web analytics [13]. A conventional relational database management system (RDBMS) could work properly if the database was small, but now the databases are getting robust day by day. So, in order to process huge data, we need machines that work very fast, database schemas that are flexible and distributed architectures. Each rule must have a user query. That will result in a serious decrease of the performance process optimization techniques [1]. It is claimed that NoSQL databases can provide simple access, efficient speed, and development facilities for big data application purposes. Google uses big table databases which was worlds first NoSQL database [6].

The big data success came courtesy of NoSQL database. NoSQL is a grouping of several other technologies that are defined as any database technology that is not SQL. Such can be named as Key-value store. NoSQL uses key-value always roots for saving data. Key-value databases are mainly divided into two parts one is column-oriented databases and the other is document-oriented databases. Besides, there are a lot of options that are deemed as commercial and also deemed to be open-source NoSQL databases [5]. Good review articles [3] and [6] gave analysis in such conditions and a lot of resources online have given significance in this matter for higher performance and good user experience.

MongoDB is a source available-document based platform-based database program, commonly known as the NoSQL database program. Even though MongoDB is non-relational, it inherits many relational database system. Namely sorting, secondary indexing, range queries. NoSQL has high-level scalability alongside scalability and a better structure for better performance. In 2007 MongoDB was discovered. But a couple of years later the company shifted to the open-source development model. In 2013 MongoDB name was proposed. The MongoDB database, which was released in 2009 was written in C++. It helps with a high percentage with replication sets which refers to two or more copies of data. MongoDB documents can be Pararally worked as JavaScript Object Notation (JSON) objects and stored internally using a binary encoding of JSON called BSON [19]. As counterpart of relational database, these objects do not have to have a common structure to store but these can be nestled. Basically, they represent only a specific key-value in the database. In MongoDB engine, "ACID" properties are being replaced by "BASE" architecture [3]. MongoDB is good at scaling workloads. MongoDB databases is document-oriented databases program. The data given by the user gets divided into ranges across multiple shards. MongoDB can also perform aggregation by pipeline, map- reduction and single-purpose method.

The supporting nestling data removes the need to join the data the same way relational database does. This feature provides help to large datasets like an e-commerce company data. Their data can be segmented into several parts and each of the parts are stored in a different server to equally divide the work load. The default engine is "WiredTiger". WiredTiger storing data in BSON(Binary JSON) [18]. MongoDB can pre define the data files sizes to avoid any system mulfuction. MongoDB is preferred because it serves better performance despite the fact that it is relatively new in the modern age technology market of NoSQL databases. Many projects and products are based on this engine. Such products are namely Disney Media group which is basically open source code. Each of them justified the use of MongoDB for specific reasons. For instance The new York Time uses MongoDB simply because it will be very easy for them to store documents. Most of them prefers MongoDB because of its scalability and usability in environments like clusters. MongoDB can easily handle the scalability problem in three dimension like cluster scale, performance, data scale

PostgreSQL is an open-source relational database management system (RDBMS). Previously named POSTGRES. In 1996 PostgreSQL name was proposed as an experimental mean. But in 2007 PostgreSQL's name was permanently given. For the separation of client and server in PostgreSQL, client's library becomes lighter, and if any changes are made in the database, that won't affect the client while the process runs [12]. PostgreSQL uses built-in replicas based on making the changes which are called Write-Ahead Logging (WAL). PostgreSQL also has built-in support B-tree and hash table index. In the schema, it holds objects, except for roles as well as tablespace. This schema acts like name places by allowing objects who have the same name to co-exist in the data table. PostgreSQL can handle a wide variety of data types including Boolean, Character, Binary, Bit String, Text search types are some of them. PostgreSQL has a special extension called "PostGIS". PostGIS integrates several geofunctions and supports geographic objects for any object purpose. PostGIS is based on "light Weight" geometrics and index optimization to reduce memory and disk storage [19].

1.1 Research Problem

Big data generating platform or application like ecommerce site has huge volume of data. Millions of users are generating the amount of data and in this way it is increasing day by day. Efficient performance is important issue for the customers as well as the owners of such kind of big data platform in the recent time. A big enterprise solution like ecommerce application has millions of users and they use those application in their day to day life for purchasing, renting, even selling their items. Social media application has a lot of functionalities and every day the data size is getting bigger and bigger. For example, Amazon, Facebook, Twitter, Instagram etc. has billions of users and they constantly use their products and services. Generally, ecommerce platform have big data with huge amount of records and information. This information and data must need to store and maintain properly. Besides, there are different types of data like structured, semi structured, unstructured data is creating regularly. In order to process big data where millions or even billions of user, proper database schema is important. It is also important to think about the features of ecommerce platform data as well as the type, structure of the data that fits and performs very well in databases. So, the problem occurs which database is appropriate to solve the issue. The research problem is to find an effective solution for querying and processing big amount of structured, unstructured and semi structured data in a shortest amount of time to give a great user experience when millions of users are using the system.

In this regard, a step by step design and experimentation of database is important to get the idea of performance between databases. Database plays a key role for processing large amount of data and application performance issues. Big data processing is a complex issues in relational databases. In this case, SQL databases are vertically scalable but on the other hand, NoSQL databases are horizontally scalable [15]. Generally, databases perform queries in the big data applications having millions of data. Most of the common operations that we can see from the application programming interface (API) is the crud operation which stands for (create, read, update and delete), When any new user create any account, or add new data like text and images, that time insert operation are performed in the database through the application programming interface and successfully, users data are stored in the database. When user wants to update some of his/her information on the application, a database update operation is performed through the API and user's information is updated. User wants to access or read some information, find or read operation is performed and users get the response through API from the database. And finally, when use wants to delete something from the application, delete operation is performed in the database and the data will be deleted from the database. We can see that a lot of insert, update, delete, read operations are performed constantly and it is more frequent in the big enterprise application like ecommerce. So, designing an experiment to perform insert, update, delete, select queries in various scales of data and measure the time that the database has taken can give proper understanding and insights to see which database schema can fit such large application and give better performance to millions of users. So, the research about the query execution time of both SQL and NoSQL databases can give a better view and solution to the problem.

The databases that we will be using in the example is PostgreSQL which is SQL database and MongoDB which is NoSQL database. SQL is the standard language to deal with relational model. In a relational model, data records are stored in a schema and grouped in tables. There are different numbers of rows, columns but as it is a structured model, each row must have the same number of columns. The advantage of this model is to avoid data redundancy. PostgreSQL is well known as object-relational database and stores data as structured objects. Like SQL relational model, PostgreSQL follows same syntax and schema to perform operations in applications. The schema has different objects like tables, columns, keys, etc. PostgreSQL has various use cases such as bank systems, multi repository data app, Business Intelligence , production , as well as different business applications. It is ideal to select the PostgreSQL for understanding performance in big E-commerce data.

On the other hand, NoSQL database works well when the dataset has no specific structure and ability to store huge amounts of data. NoSQL databases use key value pairs which are divided into two parts. They are column-oriented databases and document-oriented databases. MongoDB is known as a document oriented database and can easily store objects in XML, JSON, or BSON (binary coded JSON), graph oriented formatting. The advantage of BSON format is that it does not require the same structure and schema to store objects. In this way, MongoDB can solve the problem of large ecommerce platform data by dividing different numbers of parts and processing each part in a different server. MongoDB can give better insights of its performance of query execution time analysis in this research.

1.2 Research Objectives

In this research we will compare the performance between a Relational database management system (RDMS) and a Non-relational database management system. We use PostgreSQL as a relational database and MongoDB as a Non-relational database. In the case of big data analysis, we want to compare which databases perform better. Sometimes insert, select,update and delete operations take too much time in one database in a simple case so we want to find those reasons in this research. The objectives of this research are:

1. To know which feature work better in big data analysis

2. To compare time efficiency between this databases

3. To understand the workloads that specify the type and the number of operations that are executed on a database

4. To perform various queries of complexity

5. To understand in depth analysis of various structure of data in big applications.

6. To gain better view of designing big applications with proper schema model using database.

7. To understand which database works long time and avoid server crash.

Chapter 2

Literature Review

In recent years, there have been many debates and discussions going on in many unofficial online forums and groups about whether Relational Database Management System (RDBMS) is better than non-Relational Database Management System (Non-RDBMS) or not in business and E-commerce fields. There have been some official research papers too that offer some knowledge about the comparison of RDBMS and Non-RDBMS. In this section, we will look at some of those works that have helped us during our research.

Puntheeranurak in [16] used Redis as NoSQL database and MariaDB as relational database. They executed the experiment in a windows operating system and used PHP framework and XAMPP server. They used the predis library for connecting the PHP framework with Redis database. They compared the efficiency between these two database management systems by executing four simple yet most important operations in a DBMS. They were, INSERT, DELETE, UPDATE and SELECT operation. Insert was in two methods, firstly single data insert and then multiple inserts. The experiment with Delete was also done twice, first with just normal data deletion and then deletion while considering specific conditions. Select operation was also done in this manner, first with no conditions but in the second experiment, they divided the queries into simple and complex queries. Their results show that Redis outperformed MongoDB in both styles of Insert operation. Similarly, in both styles of Delete operation, Redis was faster than MariaDB. But, in the first Update operation (the one without any conditions) MariaDB is faster than Redis. When using an Update operation with specific conditions for their experiment, Redis was performed faster than MariaDB. In the Select operation without any conditions, MongoDB performs better than Redis. The results of the second experiment are different from other results. Here, Redis performed better when complex queries were used because MariaDB spent more time to process the queries. Though it was slightly faster than Redis when simple queries were used.

Aboutorabi et al [9] used MongoDB as the NoSQL database which is similar to us and as SQL the Microsoft SQL server was used. They used five scales of records consisting of 100, 1,000, 10,000, 100,000 and 1,000,000 records. With the fifth scale of 1,000,000 records, this is the work with the most number of records at a time. They also executed the operation in a windows system but they used JDBC which is Java Database Connectivity, to connect with both the databases and execute the experiment. Usual SQL queries were used on the Microsoft SQL server and then the same queries were changed to MongoDB syntax and used in the Java driver for MongoDB. They used the insert, delete, update and select operations for the evaluations of these two databases. But they also added some evaluations for queries with or without aggregate functions. The insert operation shows that MongoDB performs at a higher speed than the SQL server and the more number of records, the more efficient it will be over the SQL server. The delete operation was also a lot faster with MongoDB. The results of the update operation shows that, at the highest scale of 1,000,000 records, MongoDB is almost four times faster than the SQL server. So, basically their evaluations show that with more records, NoSQL database MongoDB is faster and more efficient than the SQL server. But then they divided the queries into 2 parts. One without aggregate functions and one with the aggregate functions. The result of the first part is that MongoDB performs better than SQL server in queries without aggregate functions. For the second part, the SQL server outperforms MongoDB. Queries with aggregate functions were the only part in which the SQL server performed better than MongoDB. MongoDB is better for very large data and databases with constant change. But MongoDB falls weak when aggregate functions are executed.

[20] by Shetty and Akshay is another one of the related works where the performance of MongoDB was compared with a relational database. Here they used Oracle as the relational database. They performed the basic four operations of Insert, Delete, Update and Select and compared both the databases. The results shown in their hypothesis testing is , the performance of Insert, Delete and Select operations are better in MongoDB but for Update operation, Oracle is still outperforming MongoDB in the environment and dataset that they were given. In this paper, our focus is mainly on evaluating the performance of the SQL database PostgreSQL and the NoSQL database MongoDB for a relatively large database with a real-world representation that is common to E-commerce applications. And comparing them both to determine which is better for this purpose.

[17] by Sharma, Vishal and Bundele is one of the related works where they analyze the performance of relational database with Non- relational database and graph database. They used MongoDB for NoSQL, PostgreSQL for relational model and Neo4J as graph database system. They used all the open source database in their experiment and they used OpenStreetMap data in "osm" formant¿ The data was geographical which described as nodes, relation and tags. They created the necessary tables in PostgreSQL and Neo4j and collection in MongoDB. They used java program to perform queries for finding user tagged pont, nodes etc. The performance of MongoDB was best in terms as it takes least time to execute the query, then PostgreSQL and neo4j has least performance.

[10] Chickerur, Goudar, Kinnerkar also did some works where the performance of MongoDB with the help of MySQL. They considered a huge database with 105000 records. The purpose was to preserve all the data in a old system using database migration tools. Implementation had 2 main steps namely, "Extracting the data from MySQL to csv files" and "Dumping the extracted data in the csv files to Mon-

goDB". For all the implementations, MongoDB took less time from MySQL in their operations. MongoDB outperformed MySQL in the given environment. This paper mainly focused on performance of MongoDB respect with MySQL. The authors performed the operations for a larger database record for practical use in real world (E-commerce) for better purpose.

[12] Jung, Youn, Bae, Choi compared PostgreSQL and MongoDB in big data environment. They implemented a program capable of insert, select, update, and delete operations. They have taken 300000 data records based on card milage information. With the help of unstructured data model, authors executed these operations. The authors thought it would be right not to wait until the data is inserted and to be available. Speed of performing operations have been measured by time taken from selection of data all the way to processing of data. The end result stated that MongoDB was faster then PostgreSQL in general. The authors stated that using MongoDB with unstructured data model will surely improve the performance for any real life usage. However withing structured data environment PostgreSQL will perform better then MongoDB.

In [11] Gyorodi et al did a comparative study between MongoDB and MySQL. The operating system they used was 64 bit Windows 7 Ultimate. By implementing various operations on these SQL and NoSQL databases, they justified that MongoDB is more efficient and advantageous than MySQL. They built a forum application to evaluate these databases. The forum application they built was developed using Symfony2. It is an open source PHP framework and allows easy integration of the MongoDB database. To test the performance of these two databases they performed various operations of the four basic operations that are, Insert, Select, Update and Delete. For the insertion operation, they had different tables or documents with some common data fields and having the same number of fields or columns of data. Some of the common elements between the two databases were: table/document User with the columns: id, username, password and email, table/document Forum with the columns: id, title, author, info (short description) and so on. The insertion began with inserting 10,000 users in both the databases. The id of the users were auto generated by the databases and for the username, password and email other PHP functions were used. For recording the time also they used the PHP function microtime. This function recorded the time from the beginning of the script running to the completion of the script. For the inserting of the 10,000 users, MySQL took 440 seconds while MongoDB took 0.29 seconds. After the users, they inserted 5000 forums, 5000 subforums, 5000 discussions and 5000 comments respectively. For inserting these data, MySQL took 1010 seconds while MongoDB took only 3.3331 seconds. Thus, for insertion of large amount of information in the database, MongoDB is better performing than MySQL. For the select query operations, they made two select queries, selection of all discussions a user attended and with a date different than a certain one and selection of all the users from the database and the number of discussions started by each other. The first select in MySQL took 0.0018 seconds while MongoDB executed that in 0.0011seconds. The second select took 0.6478 seconds in MySQL and 0.0052 seconds in MongoDB. So, for the select queries operation also, MongoDB is better performing than MySQL. Now, for the update operation also they used two update queries. First one was updating a comment written by a specific user and the second one was updating the email address of a user. The first update operation took 0.0987 seconds in MySQL and 0.0021 seconds in MongoDB. The second update took 0.0428 seconds in MySQL and 0.0013 seconds in MongoDB. These timings concluded MongoDB being the more performant than MySQL for the update operation. Now, for the delete operation, here too they used two delete queries for both the databases. The first query was used for deleting all the comments posted by a user and the second one deleted all the forums created by a specific user. With the deletion of the forum, all the sub forums, comments and discussions contained in the forum were also be deleted. The first delete took 0.3524 seconds in MySQL and 0.0028 seconds in MongoDB and the second delete took 0.8231 seconds in MySQL and 0.0064 seconds in MongoDB. So, for the delete operation, they found that, MongoDB was better performing than MySQL. So, overall, according to their implementations, they found that MongoDB had faster execution times in all the operations compared to MySQL.

[14] by Andersson and Beggren is a research work where they performed queries and measured performance in SQL database MySQL server and NoSQL database MogoDB. In MongoDB they store documents by creating different number of JSON objects which is known as JavaScript Object Notation. They used AMD FX 8350 CPU having 8 Gb RAM and 1 TB hardrive. Beides they used MySQL server version 5.718 and MongoDB version 3.4.4. The test were done by executing JavaScript files having different queries. They performed the insert, update, delete and select operations in both database. The types of data were mostly string and integer in table's attribute. The insert operation was performed having (0, 10000, 100000, 1000000, 10000000) amount of data and other operations were performed without having records empty in the database. MongoDB takes less time in every query execution like single-insert, multi-insert, single-update, multi-update, single-select, multi-select, single-delete, multi-delete operations.

Chapter 3

Methodology

3.1 Experiment Model Design

The main purpose of the experiment is to find the answer of the questions that most of the users regarding the performance of relational SQL and non-relational NoSQL databases. But there are some major facts about this two databases. The SQL database has the structural nature where NoSQL database like MongoDB do not have the structural nature. MongoDB do not follow the nature of relational model. Another thing is that MongoDB call easily store every form of data even there is empty or missing value or any field of the collection in NULL field. For that reason, if we want to evaluate the performance of the two database, we have to understand the feature and model of the database. So, we can design the experiment by understanding the feature and model of each database.

Besides, another important factor is that we are designing the experiment that can give us important decision making process for large scale of data or big data application. So, we must follow the actual model of large scale business application or enterprise solution that have millions of users. So, for our experiment, we are following a data model of E-commerce application. Using the data of large enterprise E-commerce application, we can perform operations such as insert, update, delete and select queries. Besides, we know the fact the data do not insert all together into the database tables, so we have to avoid instructions that insert the all data at one time. In the SQL, each table of data on the database is inserted using rows. On the other hand, in the NoSQL database, like MongoDB, each document of data is inserted.

Basic operation that can perform on the database is:
1. Insert
2. Update
3. Select
4. Update

So in our experiment a server that exists and it can handle thousands of users at a time. Besides, an important factor is that researchers want to find comparisons between two or more systems in big data applications so it is important for the experiment to use a common interface between both systems to get appropriate results. So no matter what the interface is and how it works it is important that in the design system we should use a common interface which can process different queries with different amounts of data. For our experiment we have used JavaScript runtime Node JS platform and node package manager such as npm packages in this case and used JavaScript functions to provide the optimal amount of time of processing the queries of two databases.

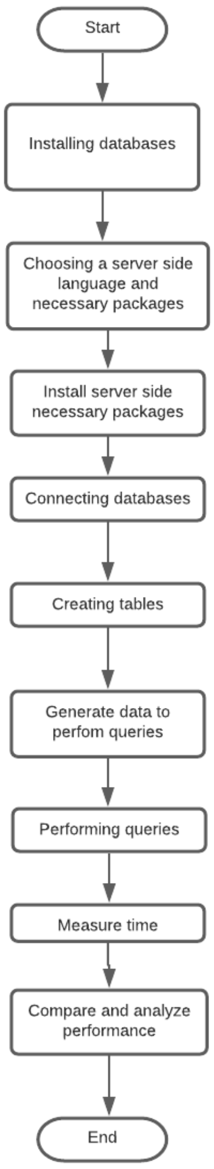

Start
Installing databases
Choosing a server side language and necessary packages
Install server side necessary packages
Connecting databases
Creating tables
Generate data to perfom queries
Performing queries
Measure time
Compare and analyze performance
End

Figure 3.1: performance evaluation process for SQL and NoSQL

In the above, there is a figure which describes the step by step implementation and analysis process for performing the experiment, getting the results data and analyze the performance in both database.
The step by step algorithm for the process is

Step 1: Start
Step 2: Install MongoDB and PostgreSQL database
Step 3: Choosing Javascript server side language Node Js and node packages
Step 4: Install necessary packages
Step 5: Connect both database to the experiment
Step 6: Create tables
Step 7: Generate various amount of data for operation
Step 8: Execute queries
Step 9: Measure the time for executing queries
Step10: Compare and analyze the results of performance

3.2 Implementation

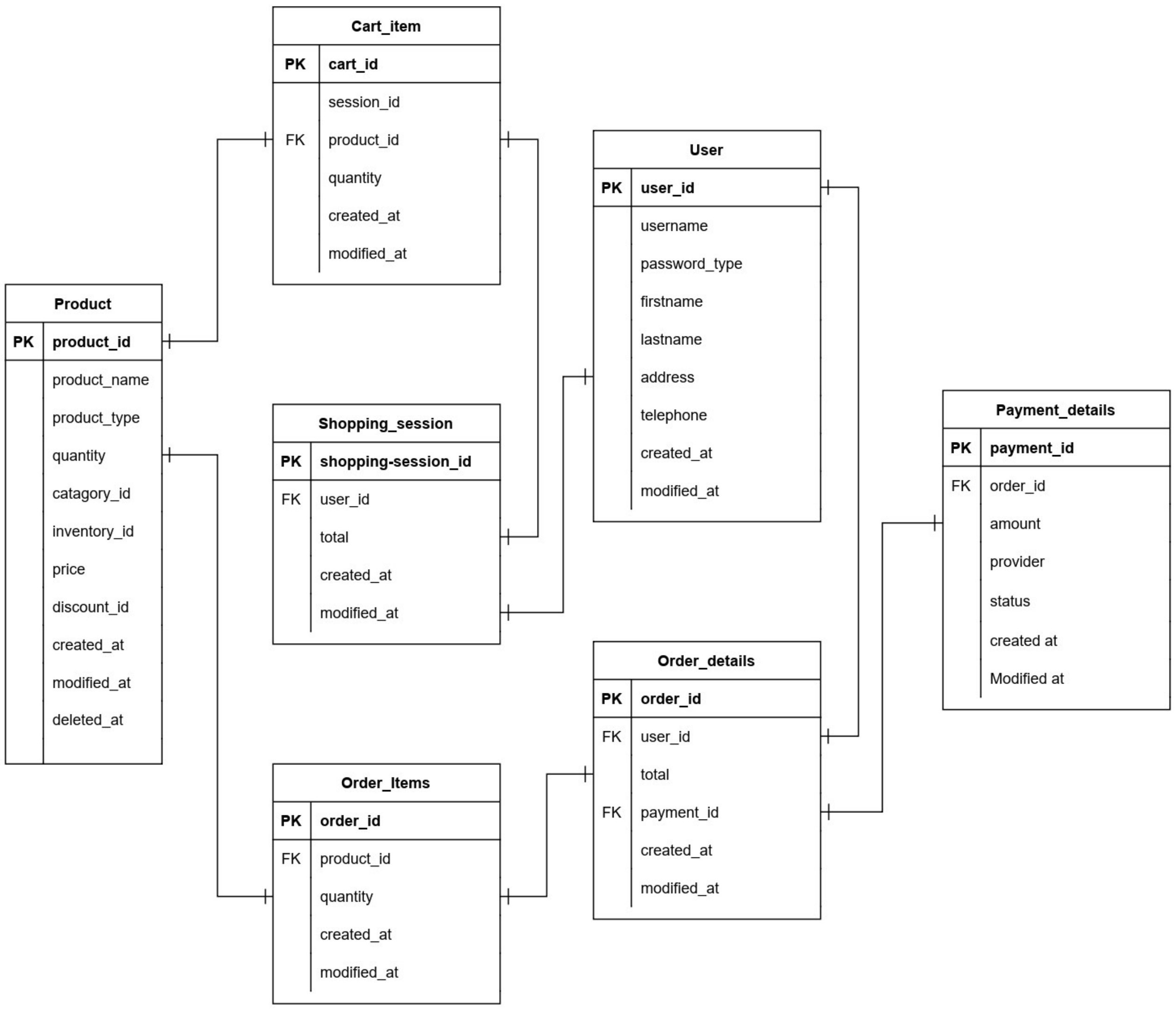


Figure 3.2: Schema Diagram of E-commerce Application

This is the model of the database . We can see the 7 tables that exist in the database.

The tables are :

1. Product table.
2. Cart item table.
3. Shopping session table.
4. Order items table.
5. Order details table.
6. User table.
7. Payment details table.

```
The schema of tables in SQL is following below:

Product table:

    product_id int PRIMARY KEY,
    product_name varchar NOT NULL,
    product_type varchar NOT NULL,
    quantity int NOT NULL,
    category_id int NOT NULL,
    inventory_id int NOT NULL,
    price decimal NOT NULL,
    discount_id int NOT NULL,
    created_at timestamp NOT NULL,
    modified_at timestamp NOT NULL,
    deleted_at timestamp NOT NUL

Cart_item table:

    cart_id int NOT NULL PRIMARY KEY,
    session_id int NOT NULL,
    product_id int NOT NULL,
    quantity int NOT NULL,
    created_at timestamp NOT NULL,
    modified_at timestamp NOT NULL,
      FOREIGN KEY (product_id)
      REFERENCES product (product_id)


Order_item table:


    order_id int NOT NULL PRIMARY KEY,
    product_id int NOT NULL,
    quantity int NOT NULL,
    created_at timestamp NOT NULL,
    modified_at timestamp NOT NULL,
      FOREIGN KEY (product_id)
      REFERENCES product (product_id)
```

Order_details table:

```
   id int NOT NULL PRIMARY KEY,
    user_id varchar NOT NULL,
    total decimal NOT NULL,
    payment_id int NOT NULL,
    created_at timestamp NOT NULL,
    modified_at timestamp NOT NULL,
      FOREIGN KEY (id)
      REFERENCES order_items (order_id)
      FOREIGN KEY (user_id)
      REFERENCES user (id)
```

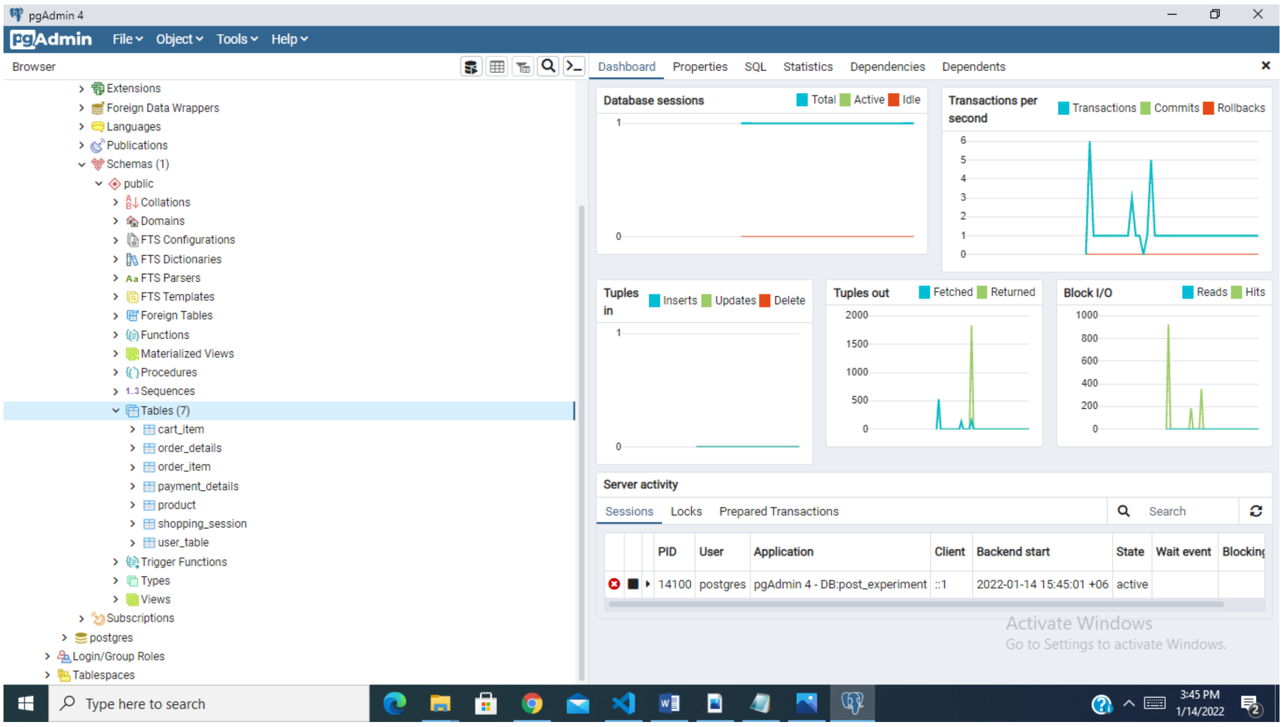
User table:

```
    user_id int NOT NULL PRIMARY KEY,
    username varchar NOT NULL,
    password_type text NOT NULL,
    firstname varchar NOT NULL,
    lastname varchar NOT NULL,
    address varchar NOT NULL,
    telephone int NOT NULL,
    created_at timestamp NOT NULL,
    modified_at timestamp NOT NULL
```

Shopping_session table:

```
    shopping_session_id int NOT NULL PRIMARY KEY,
    user_id int NOT NULL,
    total int NOT NULL,
    created_at timestamp NOT NULL,
    modified_at timestamp NOT NULL,
      FOREIGN KEY (user_id)
      REFERENCES user_table (user_id)
```

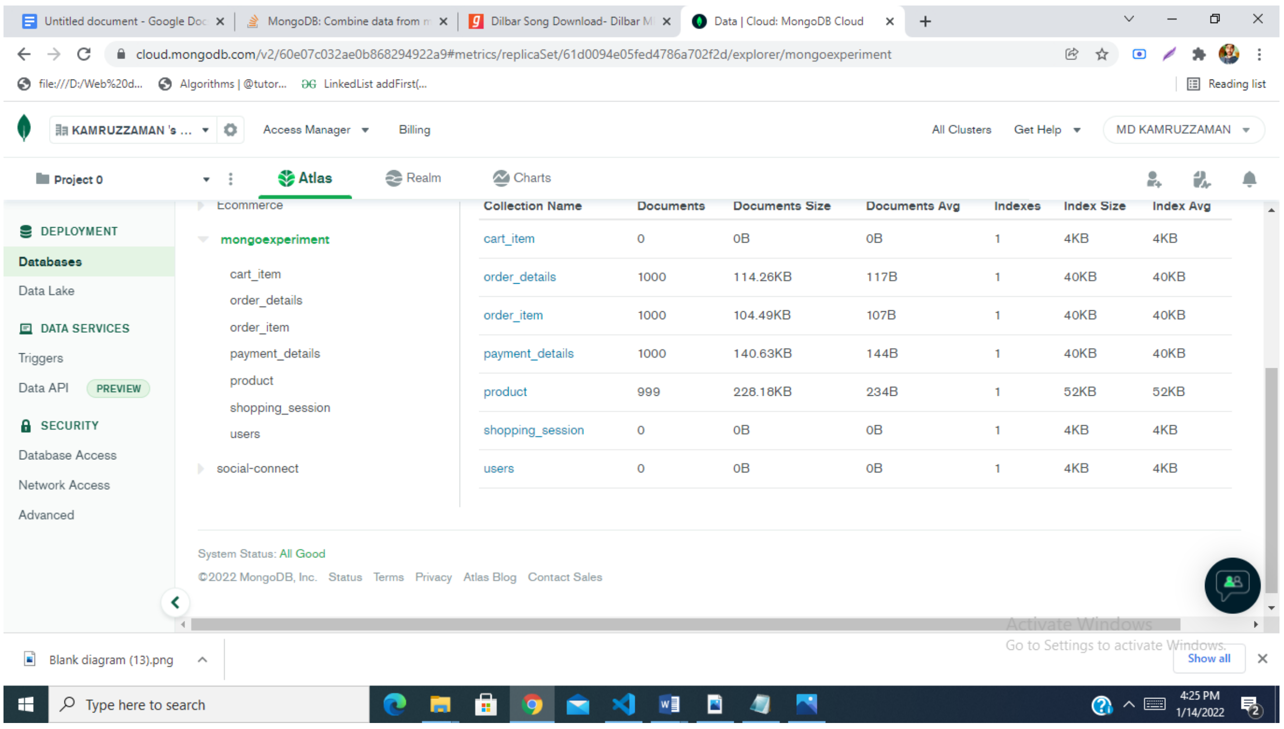
```
Payment_details table:


payment_id int NOT NULL PRIMARY KEY,
    order_id int NOT NULL,
    amount int NOT NULL,
    provider varchar NOT NULL,
          status varchar NOT NULL,
    created_at timestamp NOT NULL,
    modified_at timestamp NOT NULL
```



PID	User	Application	Client	Backend start	State	Wait event	Blocking
14100	postgres	pgAdmin 4 - DB:post_experiment	::1	2022-01-14 15:45:01 +06	active		

Figure 3.3: Table Creation in PostgreSQL

To design schema in MongoDB for every collection is very easy and it can easily accept data having null or missing value or empty objects. So, we do not have to design the schema or types of data that accept MongoDB. We can manually create collection in MongoDB official server and those collection can accept any types of data.
Product table has one to one relationships with the cart item table and order item table. Order items have a one to one relationships with order details table. Order details table has a one to one relationship with the payment details table. One to one relationship is used when we create a relationship between two tables where a single row of the first table can only be related to one and only one record of a second table. In the product table product id is present in the cart item table which is a foreign key in car iteim table. The user table has a one to one relationship with



Collection Name	Documents	Documents Size	Documents Avg	Indexes	Index Size	Index Avg
cart_item	0	0B	0B	1	4KB	4KB
order_details	1000	114.26KB	117B	1	40KB	40KB
order_item	1000	104.49KB	107B	1	40KB	40KB
payment_details	1000	140.63KB	144B	1	40KB	40KB
product	999	228.18KB	234B	1	52KB	52KB
shopping_session	0	0B	0B	1	4KB	4KB
users	0	0B	0B	1	4KB	4KB

System Status: All Good

©2022 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

Figure 3.4: Table Creation in MongoDB

the shopping session table and order details table. In this way the design of the database is created in a relational model.

```
    {
    product_id: 1,
    product_name: 'GTX_1',
    product_type: 'Graphics_Card',
    quantity: 207,
    category_id: 24,
    inventory_id: 466,
    price: 192,
    discount_id: 489,
    created_at: 2013-03-28T10:28:10.034Z,
    modified_at: 2012-05-14T05:34:16.208Z,
    deleted_at: 2012-04-19T08:33:57.619Z
  }
```

Similar experiments were performed on SQL database. The JavaScript programming language was used to run the desired operations and measure the time of each operation. In the experiment as we know that there is no join concept in MongoDB. But MongoDB 3.2 allows us to combine data from multiple collections into one collection through the lookup aggregation function.

So if we use multiple lookup functions we can combine multiple collections in aggregation. This is the way when we join multiple collections just like joining multiple tables in relational databases using MongoDB.

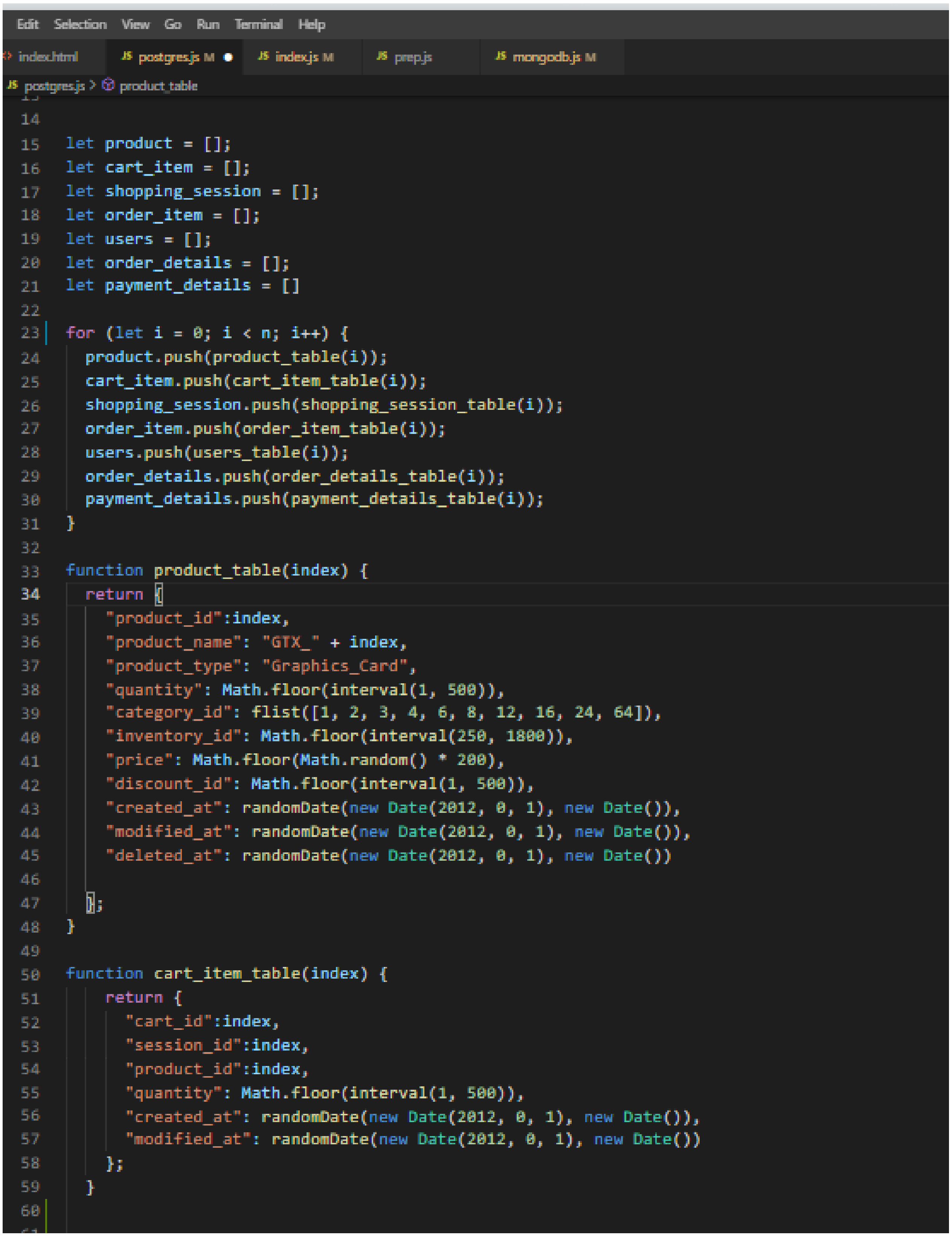
First the SQL language queries were designed to run on PostgreSQL server database then the same queries were converted to do similar operations in MongoDB syntax in order to run in JavaScript for MongoDB. In the insert operations we inserted 1000, 10000, 100000 and 500000 records into a collection and each record was read separately and was inserted into the collection in the form of a document in MongoDB database and in the form of rows in PostgreSQL database. In join operations where several tables communicate with each other one query must be performed for each collection in MongoDB and each table for postgres and get the desired result. Joining the tables can be achieved by common fields that have collections in each other. So in this way we experiment and perform Complex queries that several table communicate and participate each other

In this experiment we had to download and install both MongoDB and PostgreSQL databases in our system. The experiment was designed in Intel core i5 processor having 4 gigabytes of ram and 64 bit operating system. In this experiment we have used JavaScript as a backend language. We have used server side javascript runtime such as node js in the experiment. It is an open–source cross-platform JavaScript platform that helps building the real-time network application. Some advantages of Node.js is that it is used writing the server-side applications in the JavaScript. It can scale systems in both vertical and horizontal manner.
. As we are developing the experiment based on PostgreSQL and MongoDB database, it is important to have the node packages that execute the SQL and NoSQL databases. We have installed necessary node package library mongodb, pg, dotenv in our experiment.

Dotenv package is used for hiding the database connection url, port number, password and important features to ensure safety and security in the experiment.
At first we had to create database and table in SQL database PostgreSQL and similarly create a database and collection inside the database in MongoDB.
Then we had to connect the javascript runtime with MongoDB and PostgreSQL databases. The tests were done by executing using JavaScript files that were contained with different queries to the database. To make sure the tests were not affected by any kinds of ram usage like other background apps, tabs etc. the script were run in an environment where there were no other apps running in the background to use ram, in order to maximize the full usage to the script. The environment was made as fair as possible for both management system to get the best outcome. To show the performance between the database was made by executing and looking into how fast each one of the JavaScript could finish the execution of different types of queries such as inserting data updating data removing data and selecting data. These are the basic and main operations used in the experiment.

3.2.1 Data Creation Process

In this experiment we have created an empty array of object and generate data by executing function. We can create various amount of data by executing a loop of different amount of data that needed in the experiment. The data will be stored in different amount of array of object in the temporary array.


Edit Selection View Go Run Terminal Help
index.html
postgres.js M
index.js M
prep.js
mongodb.js M
postgres.js > product_table

```
let product = [];
let cart_item = [];
let shopping_session = [];
let order_item = [];
let users = [];
let order_details = [];
let payment_details = []

for (let i = 0; i < n; i++) {
  product.push(product_table(i));
  cart_item.push(cart_item_table(i));
  shopping_session.push(shopping_session_table(i));
  order_item.push(order_item_table(i));
  users.push(users_table(i));
  order_details.push(order_details_table(i));
  payment_details.push(payment_details_table(i));
}

function product_table(index) {
  return {
    "product_id":index,
    "product_name": "GTX_" + index,
    "product_type": "Graphics_Card",
    "quantity": Math.floor(interval(1, 500)),
    "category_id": flist([1, 2, 3, 4, 6, 8, 12, 16, 24, 64]),
    "inventory_id": Math.floor(interval(250, 1800)),
    "price": Math.floor(Math.random() * 200),
    "discount_id": Math.floor(interval(1, 500)),
    "created_at": randomDate(new Date(2012, 0, 1), new Date()),
    "modified_at": randomDate(new Date(2012, 0, 1), new Date()),
    "deleted_at": randomDate(new Date(2012, 0, 1), new Date())

  };
}

function cart_item_table(index) {
    return {
      "cart_id":index,
      "session_id":index,
      "product_id":index,
      "quantity": Math.floor(interval(1, 500)),
      "created_at": randomDate(new Date(2012, 0, 1), new Date()),
      "modified_at": randomDate(new Date(2012, 0, 1), new Date())
    };
  }
```

Figure 3.5: Data Creation Process In SQL NoSQL

3.2.2 Query Execution Process

After generating the data, we had pushed the array into the database table in PostgreSQL and MongoDB database collection. After inserting the different amount of rows and objects of data in the both database, we have performed the other queries such as select, update and delete operations.
During the period of executing queries in different amounts of data, we had measured the time that had taken in the both database. We had used JavaScript function new Date() which had helped us to measure the execution of queries in milliseconds. Then we had created a table of execution time of insert, update, delete, select operations in 1000, 10000, 100000, 500000 data. Based on the results that we had found we created graph to visualize the performance statistically.

Time Measuring Technique:

As we are using the JavaScript platform in our experiment, so JavaScript has a function named new Date() which is used to find the current timestamp in milliseconds. So before, query execution process, we have used first timestamp to get the current time and after executing queries we have used second timestamp. Then we subtract the second timestamp to first timestamp. In this way, we get the query execution time in milliseconds.

List of Queries

1. Insert product details data (1000 products, 10000 products, 100000 products, 500000 products)

2. Update each of product name field in product data

3. Select all products details data

4. Delete all products details data

5. Find product name, product type, order id, payment id joining order item table and order details table

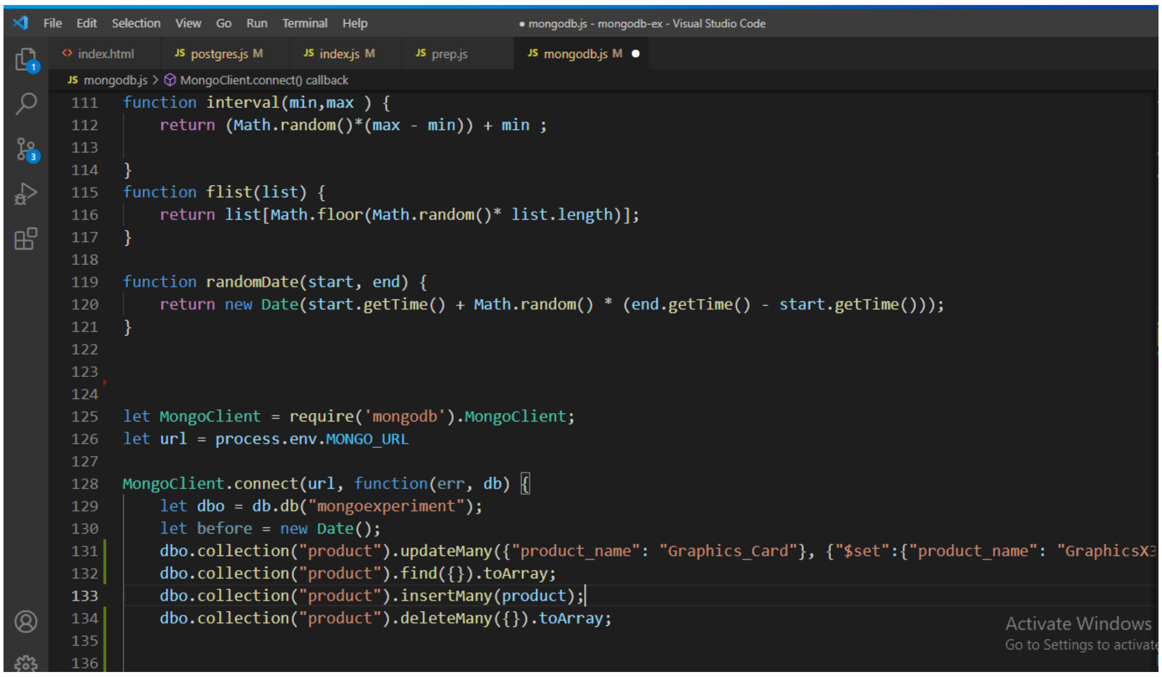
6. Find product name, product type, order id, payment id, provider where product name = “GTX 0” joining order item table and order details table and payment details table

Mongoexperiment is the name of the database where all the MongoDB works were done and. “npm install mongoDB” is used to perform JavaScript with MongoDB. Then we had connected the Mongoexperiment database with Node Js using Mongoclient.connect function. We had created empty arraya named product, cart item, shopping session, order item, users, order details, payment details as same as the collections of MongoDB database and used function to store different amount of data in the those arrays. Then we have created another variable named before which act as the initial timestamp for our experiment.

MongoDB has the insertMany function which helps to insert many documents at a time. So, we pushed the array to perform insert operation in the MongoDB database and created a final timestamp and finally we subtracted the final timestamp from the initial timestamp to get the time in milliseconds.

After inserting documents in MongoDB, we had performed the update operations and measure the time. The updateMany function helps to update any number of records and fields in MongoDB. We updated the field product name field in our data. After inserting documents, the product name field was “Graphics Card” and we updated the record as “GraphicsX4 Card”. The updateMany function updated all the documents in the mongo schema. We measured the time in the same process of insertion of data.

After updating documents in MongoDB, we had performed the select operations and measure the time. The find function in MongoDB helps to find any number of documents or any documents from the schema of the database. By passing the in find function, we were successfully capable to select all the documents from the schema of the database. We measured the time in the same process of insertion of data and updating data.

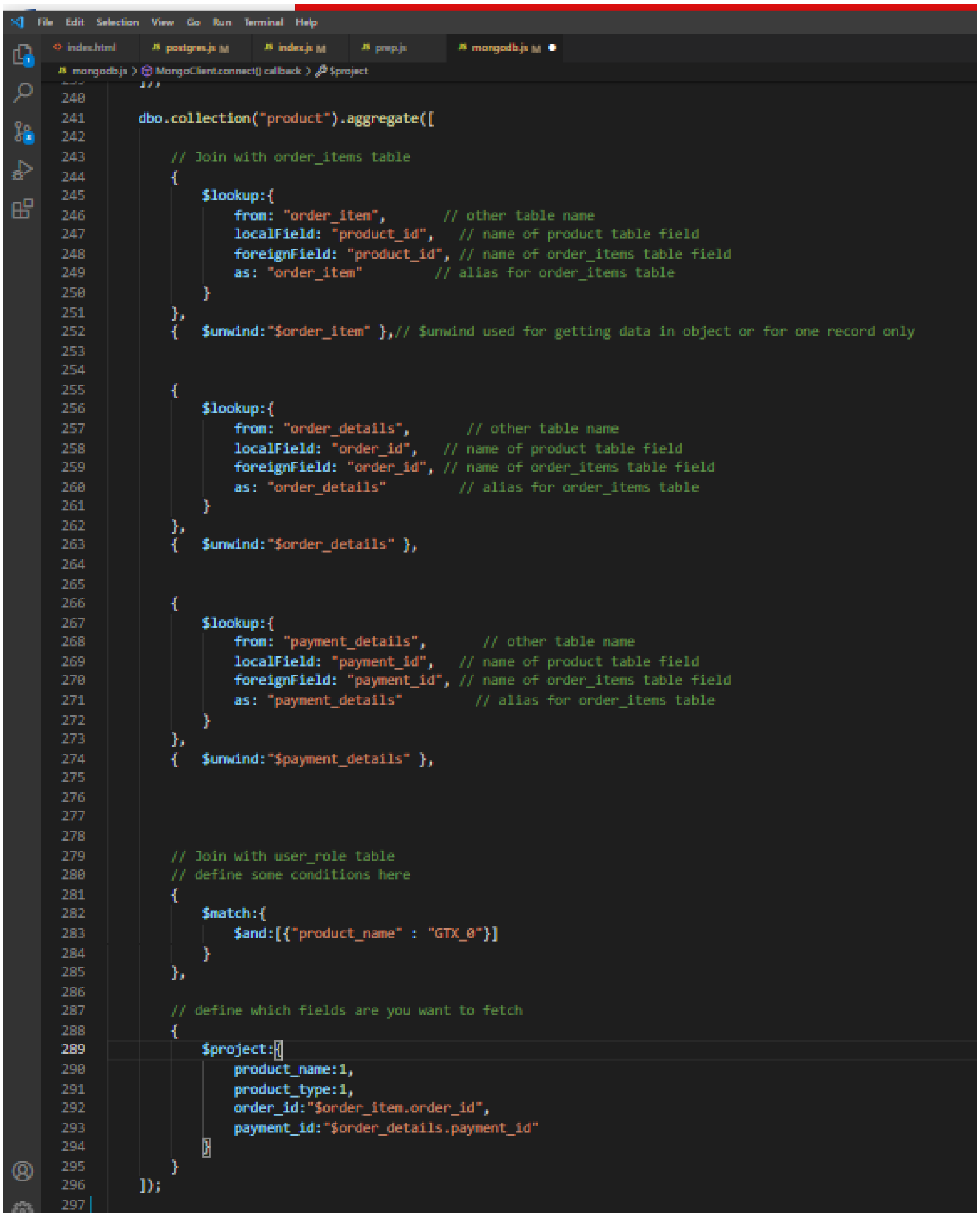

File Edit Selection View Go Run Terminal Help
• mongodb.js - mongodb-ex - Visual Studio Code
index.html
postgres.js M
index.js M
prep.js
mongodb.js M
mongodb.js > MongoClient.connect() callback
Activate Windows
Go to Settings to activate

```
function interval(min,max ) {
    return (Math.random()*(max - min)) + min ;

}
function flist(list) {
    return list[Math.floor(Math.random()* list.length)];
}

function randomDate(start, end) {
    return new Date(start.getTime() + Math.random() * (end.getTime() - start.getTime()));
}



let MongoClient = require('mongodb').MongoClient;
let url = process.env.MONGO_URL

MongoClient.connect(url, function(err, db) {
    let dbo = db.db("mongoexperiment");
    let before = new Date();
    dbo.collection("product").updateMany({"product_name": "Graphics_Card"}, {"$set":{"product_name": "GraphicsX
    dbo.collection("product").find({}).toArray;
    dbo.collection("product").insertMany(product);
    dbo.collection("product").deleteMany({}).toArray;
```

Figure 3.6: insert , update, delete, find, operations in MongoDB

After selecting documents in MongoDB, we had performed the delete operations and measure the time. The deleteMany function in MongoDB helps to delete any number of documents or any documents from the schema. By passing the in deleteMany function, we were successfully capable to delete all the documents from the schema of the database. We measured the time in the same process of other operations.
After doing all those operation, we perform joining collections to join multiple collections to find the desired outcome. The lookup property is used in MongoDB for joining more than one collection. We joined product collection with order items

and order details collection to find the product name, product type, order id and payment id fields.
Then we joined product collection with order items ,order details and payment details collection to find the product name, product type, order id and payment id, provider fields where product name = “GTX 0



```
});

dbo.collection("product").aggregate([

    // Join with order_items table
    {
        $lookup:{
            from: "order_item",        // other table name
            localField: "product_id",   // name of product table field
            foreignField: "product_id", // name of order_items table field
            as: "order_item"          // alias for order_items table
        }
    },
    {   $unwind:"$order_item" },// $unwind used for getting data in object or for one record only


    {
        $lookup:{
            from: "order_details",        // other table name
            localField: "order_id",    // name of product table field
            foreignField: "order_id", // name of order_items table field
            as: "order_details"          // alias for order_items table
        }
    },
    {   $unwind:"$order_details" },


    {
        $lookup:{
            from: "payment_details",        // other table name
            localField: "payment_id",   // name of product table field
            foreignField: "payment_id", // name of order_items table field
            as: "payment_details"          // alias for order_items table
        }
    },
    {   $unwind:"$payment_details" },




    // Join with user_role table
    // define some conditions here
    {
        $match:{
            $and:[{"product_name" : "GTX_0"}]
        }
    },

    // define which fields are you want to fetch
    {
        $project:{
            product_name:1,
            product_type:1,
            order_id:"$order_item.order_id",
            payment_id:"$order_details.payment_id"
        }
    }
]);
```

Figure 3.7: Join Collection Using MongoDB

For PostgreSQL,“post experiment” is the database where all the PostgreSQL works were done and product table, cart item table, shoppinh session table, order item table, users table, order details table, and payment details table. In order to create those tables we had to define the structure of data and records in the database. “npm install pg” is used to perform javascript with PostgreSQL . Host id, user id, port, password and database name was selected and client.connect function helped to connect JavaScript with PostgreSQL for the experiment.

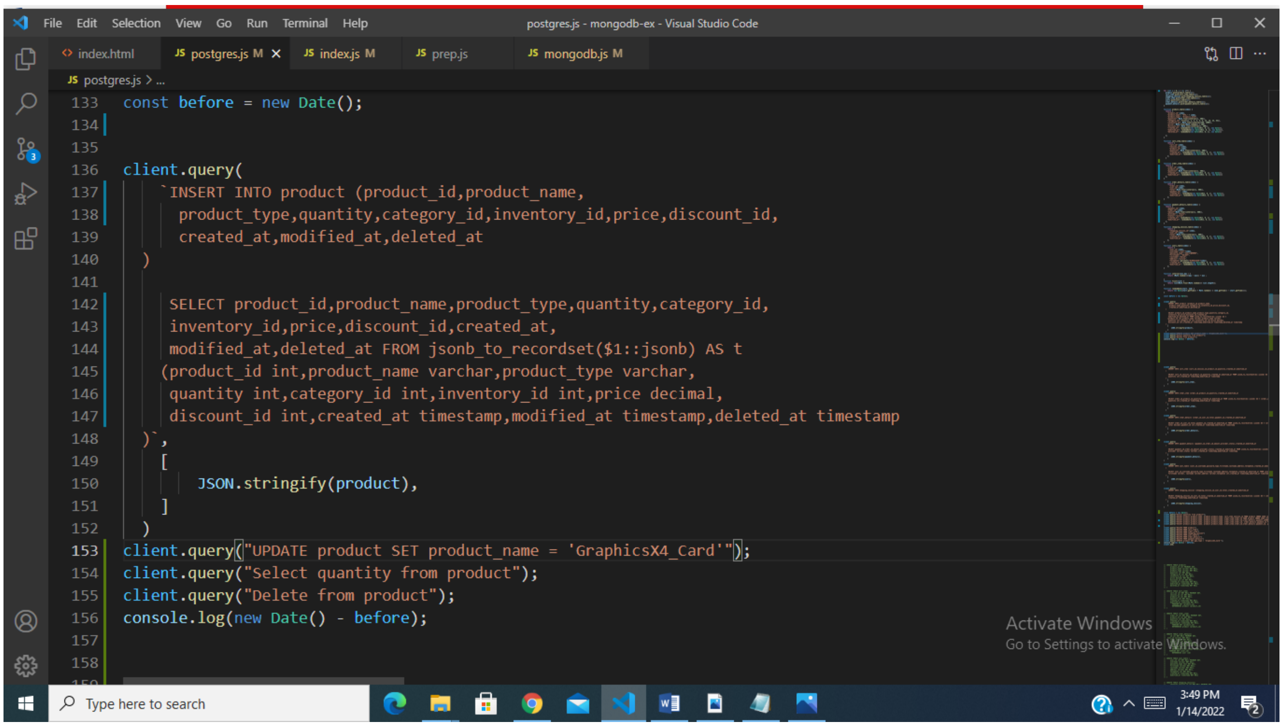
The data is inserted into the database with ‘Insert Into “Table name” and selected all of the fields from the table and with the help of

```
\jsonb to recordset($1::jsonb)" and \JSON.stringify(array)"
```

function we were able to insert the data into the database. So, we pushed the array to perform insert operation in the database and similar as MongoDB, we created a final timestamp and finally we subtracted the final timestamp from the initial timestamp to get the time in milliseconds.

After inserting documents in PostgreSQL, we had performed the update operations and measure time. We updated the field similar as MongoDB product name field in our data. After inserting documents, the product name field was “Graphics Card” and we updated the record as “GraphicsX4 Card”. “Update table name set updated field” helped us to perform update operation to update all row’s product name field successfully. We measured the time in the same process of other operations.

After updating documents in PostgreSQL, we had performed the select operations and measure the time. “Select from table name” helped us to perform select operation to select all rows successfully. We measured the time in the same process of other operations.
After selecting documents in PostgreSQL, we had performed the delete operations and measure the time. “Delete from table name” helped us to perform delete operation to delete all rows successfully. We measured the time in the same process of other operations.


File Edit Selection View Go Run Terminal Help
postgres.js - mongodb-ex - Visual Studio Code
index.html
postgres.js M
index.js M
prep.js
mongodb.js M
postgres.js > ...
Activate Windows
Go to Settings to activate Windows.
Type here to search
3:49 PM
1/14/2022

```
const before = new Date();


client.query(
    `INSERT INTO product (product_id,product_name,
     product_type,quantity,category_id,inventory_id,price,discount_id,
     created_at,modified_at,deleted_at
  )

    SELECT product_id,product_name,product_type,quantity,category_id,
    inventory_id,price,discount_id,created_at,
    modified_at,deleted_at FROM jsonb_to_recordset($1::jsonb) AS t
   (product_id int,product_name varchar,product_type varchar,
    quantity int,category_id int,inventory_id int,price decimal,
    discount_id int,created_at timestamp,modified_at timestamp,deleted_at timestamp
  )`,
    [
       JSON.stringify(product),
    ]
  )
client.query("UPDATE product SET product_name = 'GraphicsX4_Card'");
client.query("Select quantity from product");
client.query("Delete from product");
console.log(new Date() - before);
```

Figure 3.8: insert , update, delete, select, operations in PostgreSQL

After doing all those operation, we perform joining table to join multiple table to find the desired outcome. The inner join property is used in SQL for joining more than one table. We joined product table with order items and order details table to find the product name, product type, order id and payment id fields.
Sample code for joining three tables:
SELECT product.product name, product.product type, order item.order id,order details.payment id FROM product INNER JOIN order item ON product.product id=order item.product id INNER JOIN order details ON order item.order id = order details.order id

Then we joined product table with order items ,order details and payment details table to find the product name, product type, order id and payment id, provider fields where product name = "GTX 0".
Sample code for joining four tables with condition:
SELECT product.product name, product.product type, order item.order id,order details.payment id, payment details.provider FROM product INNER JOIN order item ON product.product id=order item.product id INNER JOIN order details ON order item.order id = order details.order id INNER JOIN payment details ON order details.payment id = payment details.payment id WHERE product.product name = 'GTX 0

In this way, we have performed queries both of the databases.

Chapter 4

Analysis

	No. of Records	1000	10,000	100,000	500,000
PostgreSQL	Insert	16	131	1323	6671
	Update	1	1	1	1
	Simple select	1	1	1	1
	Complex select	1	1	1	1
	Very complex select	1	1	1	1
	Delete	1	1	2	2
MongoDB	Insert	35	125	576	2591
	Update	1	1	1	1
	Simple select	3	3	3	3
	Complex select	3	3	3	4
	Very complex select	9	9	9	9
	Delete	1	1	2	2

Table 4.1: Table of Execution Time For MongoDB and PostgreSQL

In the above, there is a table which is the query execution time table that we have found based on the implementation. So, the table describes the insert, update, simple select, complex select, very complex select and delete operations on the 1000,10000,100000,500000 amount of data for both in PosgreSQL and MongoDB database.

4.1 INSERT OPERATION

In Insert operation, PostgreSQL took 16 millisecond for inserting 1000 data and 131 milliseconds for inserting 10000 data. And it took 1323 milliseconds for 100000 data insertions and 6671 milliseconds for inserting 500000 data. MongoDB on the other hand, took 35 millisecond for inserting 1000 data and 125 milliseconds for inserting 10000 data. And it took 576 milliseconds for 100000 data insertions and 2591 milliseconds for inserting 500000 data.From the above states tics we can see that MongoDB took less time then PostgreSQL to perform. We can state that PostgreSQL is not performing better than MongoDB

Insert Operation	insert 1000 data	insert 10000 data	insert 100000 data	insert 500000 data
PostgreSQL	16 milliseconds	131 milliseconds	1323 milliseconds	6671 milliseconds
MongoDB	35 millisecond	125 millisecond	576 millisecond	2591 milisecond

Table 4.2: Insert operation query execution time table

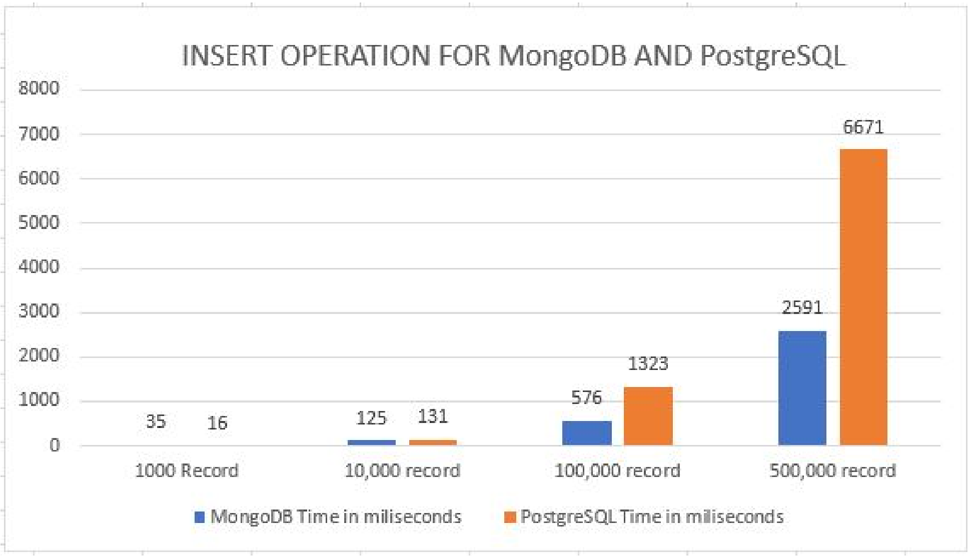


Figure 4.1: INSERT Operation Query Execution Graph

Hypothesis Testing For each operation can be performed. Procedure Taken to perform the hypothesis testing is "ANOVA" for all the operations.
Following hypothesis can be stated: H0: Performance of Postgresql for insert is not better than the performance of MongoDB. H1: Performance of Postgresql for insert is better than the performance of MongoDB
Calculation:

$$\begin{aligned} \text{Grand Total T} &= \sum x1 + \sum x2 \\ &= 8041 + 3327 \\ &= 11368 \end{aligned} \tag{4.1}$$

$$\begin{aligned} \text{Connection Factor } &= \mathrm{T}^2/\mathrm{N} \\ &= 11368^2/8 \\ &= 16153928 \end{aligned} \tag{4.2}$$

$$\text{Sum of Square between sample, } \begin{aligned} \text{SSC} &= 8041^2/4 + 3327^2/4 - 16153928 \\ &= 2777724.5 \end{aligned} \tag{4.3}$$

$Df = c - 1 \quad = 1$

$$\begin{aligned} SST &= \sum x1^2 + \sum x2^2 - T^2/N \\ &= 46253787 - 7061907 \\ &= 37161766 \end{aligned} \tag{4.4}$$

$$\begin{aligned} \text{Sum of square within sample SSE} &= \text{ SST-SSC} \\ &= 37161766 - 2777724.5 \\ &= 34384041 \end{aligned}$$

One way Anova table:

Source of variation	Sum of square	Degree of freedom	Mean sum of square
Between sample	SSC=2777724.5	c-1=1	MSC=277772.5/1=277772.5
Within sample	SSE=34384041.5	n-c=6	MSE=34354041.5/6=5730673.583
	SST=37161766	n-1=7	

Table 4.3: Anova Table for INSERT Operation

$$\begin{aligned} F_{(1,8)} &= 277772.5/5730673.583 \\ &= 0.05 \end{aligned} \tag{4.5}$$

Hence, the F(calculated) is 0.05. F(tabulated) is 5.99 F(calculated) is less than F(tabulated). SO, the H0 is completed meaning the performance of PostgreSQL for insert is not better than the performance of MongoDB.

4.2 SIMPLE SELECT OPERATION

In simple select operation, PostgreSQL took 1 millisecond for slecting 1000, 10000, 100000 and. 500000 data. MongoDB on the other hand, took 3 millisecond for selecting 1000, 10000, 100000 and. 500000 data.Here we can see that PostgreSQL took less time then MongoDB for both select and Very complex select operation. So PostgreSQL is performing better than MongoDB.

Select Operation	insert 1000 data	insert 10000 data	insert 100000 data	insert 500000 data
MongoDB	3 milliseconds	3 milliseconds	3 milliseconds	3 milliseconds
PostgreSQL	1 millisecond	1 millisecond	1 millisecond	1 milisecond

Table 4.4: Select operation query execution time table

Starting the following hypothesis: H0: Performance of Mongodb for very simple select is not better than the performance of Postgresql. H1: Performance of Mongodb for simple select operation is better than Postgresql
Calculation:

$$\begin{aligned}\text{Grand Total T} &= \sum x1 + \sum x2 \\ &= 4 + 12 \\ &= 16\end{aligned} \tag{4.6}$$

$$\begin{aligned}\text{Connection Factor } &= \mathrm{T}^2/\mathrm{N} \\ &= 16^2/8 \\ &= 32\end{aligned} \tag{4.7}$$

$$\begin{gathered}\text{Sum of Square between sample } \begin{aligned}\text{ssc} &= 4^2/4 + 32^2/4 - 32 \\ &= 8\end{aligned} \\ \begin{aligned}\text{Df } &= \text{c} - 1 \\ &= 1\end{aligned}\end{gathered} \tag{4.8}$$

$$\begin{aligned}SST &= \sum x1^2 + \sum x2^2 - T^2/N \\ &= 4 + 36 - 32 \\ &= 8\end{aligned}$$

$Df = n - 1 = 8 - 1 = 7$

Sum of square within sample SSE=SST-SSC =8-8 =0 One way Anova table:

Source of variation	Sum of square	Degree of freedom	Mean sum of square
Between sample	SSC=8	c-1=1	MSC=8/1=8
Within sample	SSE=0	n-c=6	MSE=0/6=0
	SST=8	n-1=7	

Table 4.5: Anova Table for Simple SELECT Operation

F(1,6)=8/0 =0
Hence, the F(calculated) is 0. (Ftabulated) is 5.99 F(calculated) is less than F(tabulated). Therefore the H0 is completed meaning the performance of Mongodb is not better than PostgreSQL for simple select.

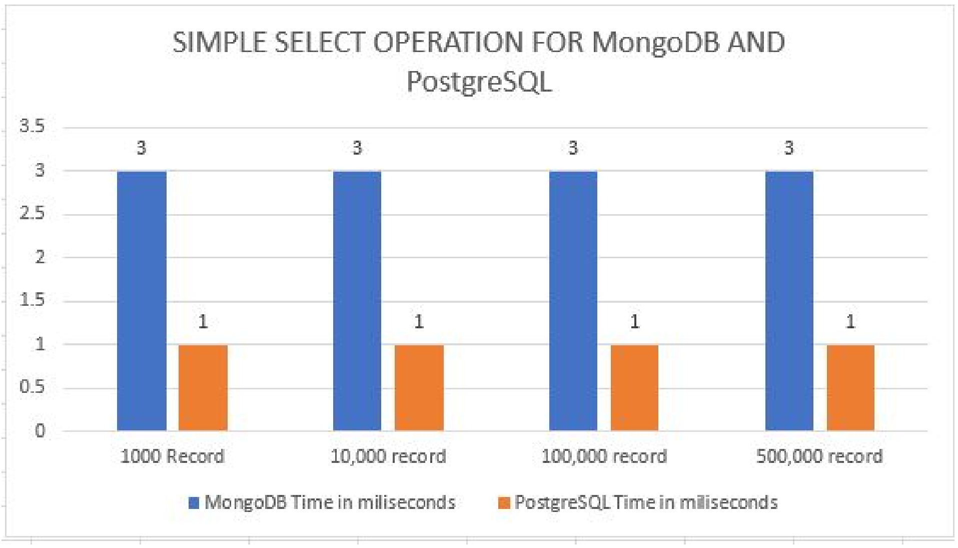


Figure 4.2: SELECT Operation Query Execution Graph

4.3 COMPLEX SELECT JOIN OPERATION

In complex select operation, PostgreSQL took 1 millisecond for slecting 1000, 10000, 100000 and. 500000 data. MongoDB on the other hand, took 3 millisecond for selecting 1000, 10000, 100000 and 4 milliseconds for selecting 500000 data. From the above states tics we can see that PostgreSQL took less time then MongoDB to perform. We can state that mongoDB is not performing better than PostgreSQL

Select Operation	insert 1000 data	insert 10000 data	insert 100000 data	insert 500000 data
MongoDB	3 milliseconds	3 milliseconds	3 milliseconds	4 milliseconds
PostgreSQL	1 millisecond	1 millisecond	1 millisecond	1 milisecond

Table 4.6: Complex Select operation query execution time table

Starting the next hypothesis: H0: Performance of MongoDB for complex is better than the performance of PostgreSQL. H1: Performance of MongoDB for complex select is not better than the performance of PostgreSQL
Calculation:

$$\begin{aligned}\text{Grand Total T} &= \sum x1 + \sum x2 \\ &= 4 + 13 \\ &= 17\end{aligned} \tag{4.9}$$

$$\begin{aligned}\text{Connection Factor } &= \mathrm{T}^2/\mathrm{N} \\ &= 17^2/8 \\ &= 36.128\end{aligned} \tag{4.10}$$

$$\begin{aligned}\text{Sum of Square between sample ssc } &= 4^{2/4} + 13^2/4 - 36.128 \\ &= 10.125\end{aligned} \quad Df = c - 1 = 1$$

$$\begin{aligned} SST &= \sum x1^2 + \sum x2^2 - T^2/N \\ &= 4 + 43 - 36.128 \\ &= 10.875 \\ Df &= n - 1 = 8 - 1 = 7 \end{aligned} \tag{4.11}$$

$$\begin{aligned} \text{Sum of square within sample SSE} &= \text{SST-SSC} \\ &= 10.875 - 10.125 \\ &= 0.75 \end{aligned} \tag{4.12}$$

One way Anova table:

Source of variation	Sum of square	Degree of freedom	Mean sum of square
Between sample	SSC=10.125	c-1=1	10.125/1=10.125
Within sample	SSE=0.75	n-c=6	MSE=0.75/6=0.125
	SST=10.875	n-1=7	

Table 4.7: Anova Table for Complex SELECT Operation

F(1,6)=10.125/0.125 =81
Hence, the F(calculated) is 81. F(tabulated) is 5.99 F(calculated) is greater than F(tabulated). Therefore the H1 is completed which means the performance of MongoDB is not better than PostgreSQL for complex select.

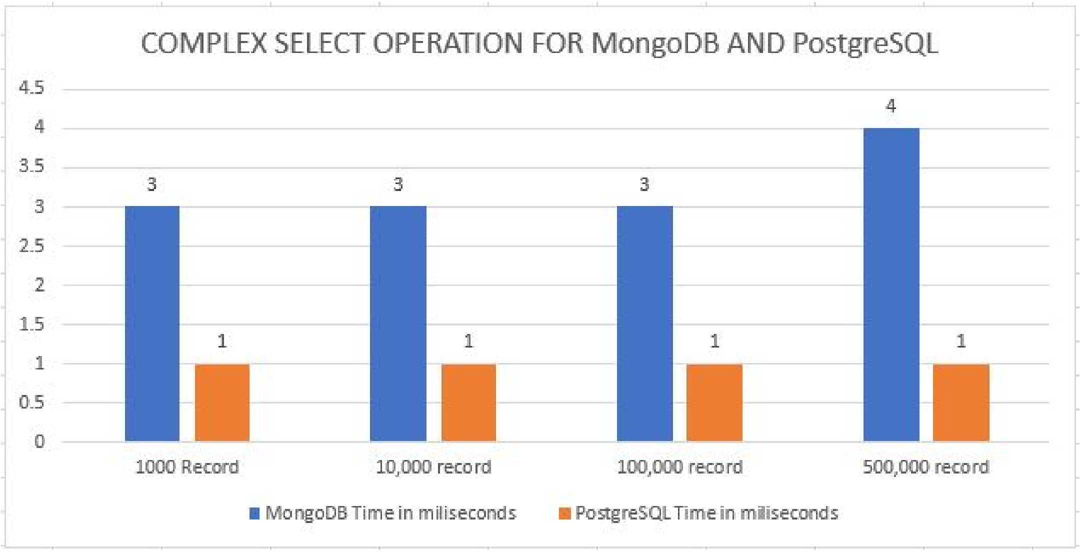


Figure 4.3: Complex Select operation query execution

4.4 VERY COMPLEX SELECT JOIN OPERATION

In very complex select operation, PostgreSQL took 1 millisecond for slecting 1000, 10000, 100000 and. 500000 data. MongoDB on the other hand, took 4 millisecond for selecting 1000, 10000, 100000, 500000 data. Here we can see that PostgreSQL took less time then MongoDB for both select and Very complex select operation. So PostgreSQL is performing better than MongoDB.

Select Operation	insert 1000 data	insert 10000 data	insert 100000 data	insert 500000 data
MongoDB	4 milliseconds	4 milliseconds	4 milliseconds	4 milliseconds
PostgreSQL	1 millisecond	1 millisecond	1 millisecond	1 milisecond

Table 4.8: VERY Complex Select operation query execution time table

Following hypothesis can be stated: H0: Performance of PostgreSQL for very complex select is better than MongoDB. H1: Performance of PostgreSQL for very complex select is not better than MongoDB
Calculation:

$$\begin{aligned}\text{Grand Total T} &= \sum x1 + \sum x2 \\ &= 4 + 16 \\ &= 20\end{aligned} \tag{4.13}$$

$$\begin{aligned}\text{Connection Factor } &= \mathrm{T}^2/\mathrm{N} \\ &= 20^2/8 \\ &= 50\end{aligned} \tag{4.14}$$

$$\begin{aligned}\text{Sum of Square between sample } \text{ssc} &= 4^2/4 + 16^2/4 - 50 \\ &= 18 \\ \text{Df} &= \text{c} - 1 \\ &= 1\end{aligned} \tag{4.15}$$

$$\begin{aligned}SST &= \sum x1^2 + \sum x2^2 - T^2/N \\ &= 4 + 46 - 50 \\ &= 18\end{aligned}$$

$Df = n - 1 = 8 - 1 = 7$

Sum of square within sample SSE=SST-SSC =18-18 =0

Source of variation	Sum of square	Degree of freedom	Mean sum of square
Between sample	SSC=18	c-1=1	MSC=18/1=18
Within sample	SSE=0	n-c=6	MSE=0/6=0
	SST=18	n-1=7	

Table 4.9: Anova Table for Very Complex SELECT Operation

F(1,6)=18/0 =0

So, the F(calculated) is 0. F(tabulated) is 5.99 F(calculated) is less than F(tabulated). Therefore the H0 is performed meaning the performance of PostgreSQL is better than MongoDB for very complex select operation.

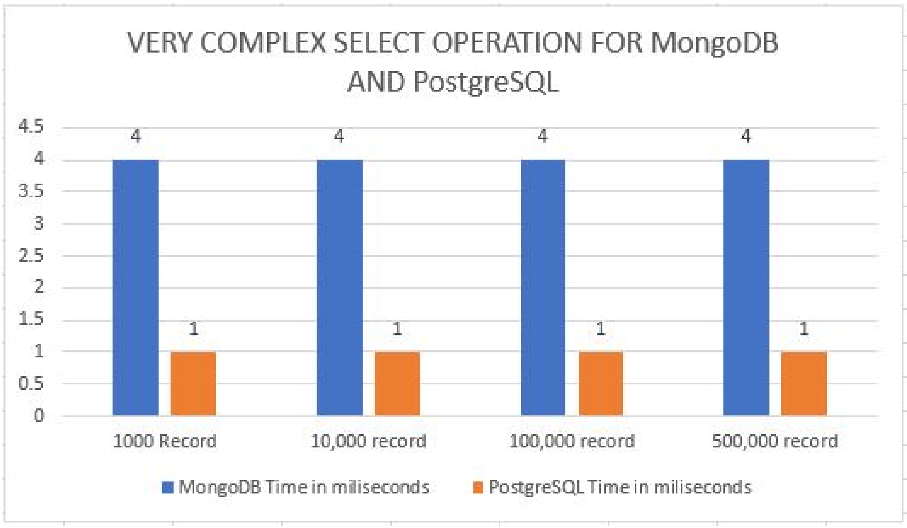


Figure 4.4: VERY COMPLX SELECT operation query execution

4.5 UPDATE OPERATION

In update operation, PostgreSQL took 1 millisecond for updating 1000, 10000, 100000 and. 500000 data. MongoDB on the other hand, took 9 millisecond for inserting 1000, 10000, 100000 and. 500000 data. From the above states tics we can see that PostgreSQL took less time then MongoDB to perform. We can state that mongoDB is not performing better than PostgreSQL

Select Operation	insert 1000 data	insert 10000 data	insert 100000 data	insert 500000 data
MongoDB	9 milliseconds	9 milliseconds	9 milliseconds	9 milliseconds
PostgreSQL	1 millisecond	1 millisecond	1 millisecond	1 milisecond

Table 4.10: Update operation query execution time table

Starting the following hypothesis: H0: Performance of PostgreSQL for very Update is better than MongoDB. H1: Performance of PostgreSQL for update select is not better than MongoDB
Calculation:

$$\begin{aligned}\text{Grand Total T} &= \sum \times 1 + \sum \times 2 \\ &= 4 + 36 \\ &= 40\end{aligned} \tag{4.16}$$

$$\text{Connection Factor } = \text{T}^2/\text{N} \ = 40^2/8 = 200 \tag{4.17}$$

$$\text{Sum of Square between sample, } \begin{aligned} \text{ssc} &= 4^2/4 + 36^2/4 - 200 \\ &= 128 \end{aligned} \tag{4.18}$$

$$\text{Df } = \text{c} - 1 = 1$$

$$\begin{aligned} SST &= \sum x1^2 + \sum x2^2 - T^2/N \\ &= 4 + 328 - 200 \\ &= 128 \end{aligned}$$

Df $= n - 1 = 8 - 1 = 7$

Sum of square within sample SSE=SST-SSC $= 128 - 128 = 0$
One way Anova table:

Source of variation	Sum of square	Degree of freedom	Mean sum of square
Between sample	SSC=128	c-1=1	MSC=128/1=128
Within sample	SSE=0	n-c=6	MSE=0/6=0
	SST=128	n-1=7	

Table 4.11: Anova Table for UPDATE Operation

F(1,6)=128/0 =0
Hence, the F(calculated) is 0. F(tabulated) is 5.99 F(calculated) is less than F(tabulated). Therefore the H0 is completed meaning the performance of PostgreSQL is better than MongoDB for Update.

4.6 DELETE OPERATION

In delete operation, PostgreSQL took 1 millisecond for updating 1000, 10000 and 2 millisecond for 100000 and. 500000 data. MongoDB on the other hand, took 9 millisecond for deleting 1000, 10000, 100000 and. 500000 data. Here we can see that PostgreSQL took less time then MongoDB for both select and Very complex select operation. So PostgreSQL is performing better than MongoDB.

Select Operation	insert 1000 data	insert 10000 data	insert 100000 data	insert 500000 data
MongoDB	9 milliseconds	9 milliseconds	9 milliseconds	9 milliseconds
PostgreSQL	1 millisecond	1 millisecond	2 millisecond	2 milisecond

Table 4.12: DELETE operation query execution time table

Starting the following hypothesis: H0: Performance of MongoDB for delete is better than the performance of PostgreSQL. H1: Performance of MongoDB for delete is not better than the performance of PostgreSQL

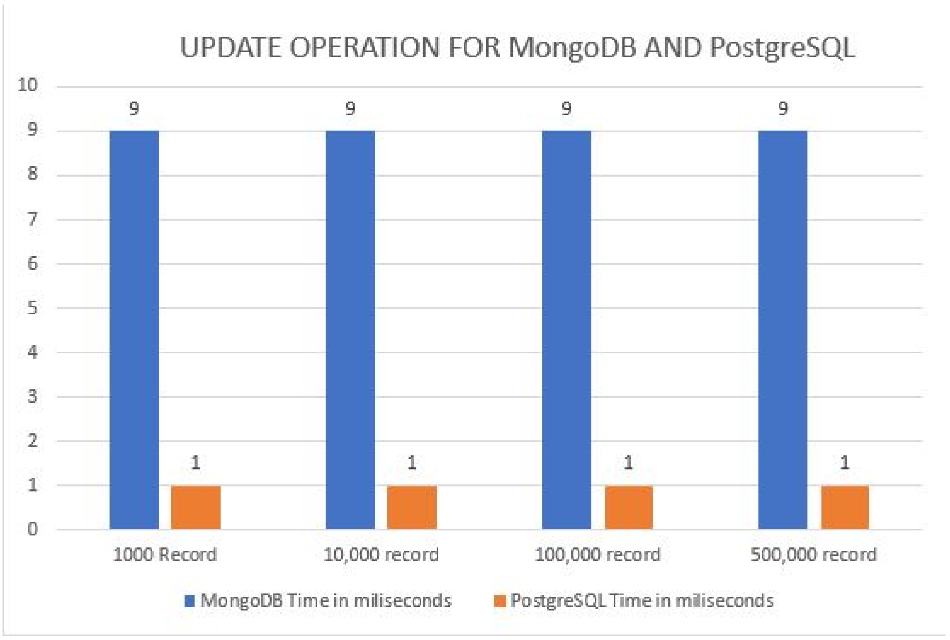


Figure 4.5: UPDATE operation query execution

Calculation:

$$\begin{aligned}\text{Grand Total } T &= \sum x1 + \sum x2 \\ &= 6+36 \\ &= 42\end{aligned} \tag{4.19}$$

$$\begin{aligned}\text{Connection Factor } &= \mathrm{T}^2/\mathrm{N} \\ &= 42^2/8 \\ &= 220.5\end{aligned} \tag{4.20}$$

$$\text{Sum of Square between sample ssc} = 6^2/4 + 36^2/4 - 220.5 \quad \begin{aligned} &= 112.5 \\ \text{Df} &= \text{c} - 1 \\ &= 1 \end{aligned} \tag{4.21}$$

$$\begin{aligned}SST &= \sum x1^2 + \sum x2^2 - T^2/N \\ &= 10 + 324 - 220.5 \\ &= 113.5\end{aligned} \tag{4.22}$$

$$\begin{aligned}\text{Sum of square within sample SSE } &= \text{ SST-SSC} \\ &= 113.5 - 112.5 \\ &= 1\end{aligned} \tag{4.23}$$

One way Anova table:
F(1,6)=112.5/0.1667 =675
Hence, the F(calculated) is 675. F(tabulated) is 5.99 F(calculated) is greater than F(tabulated). Therefore the H1 is completed meaning the performance of MongoDB is not better than PostgreSQL for delete

Source of variation	Sum of square	Degree of freedom	Mean sum of square
Between sample	SSC=112.5	c-1=1	MSC=112.5/1=112.5
Within sample	SSE=1	n-c=6	MSE=1/6=0.1667
	SST=113.5	n-1=7	

Table 4.13: Anova Table for DELETE Operation

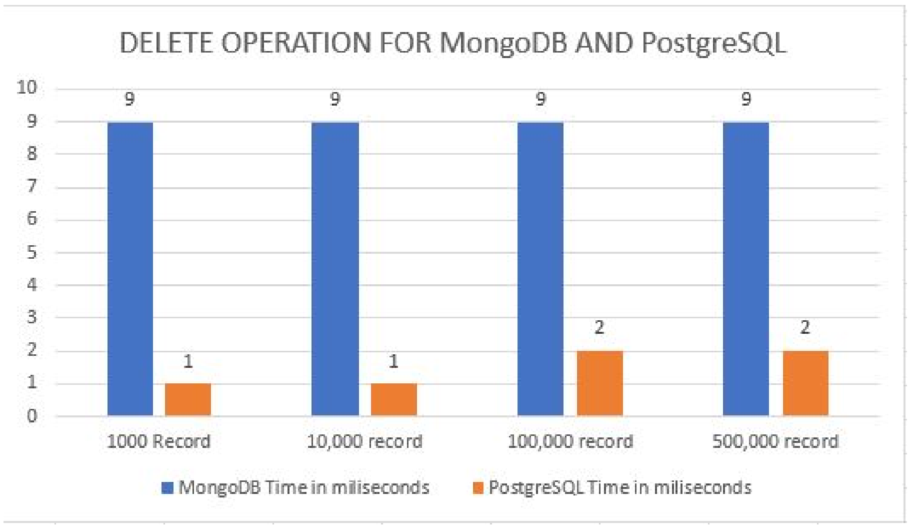


Figure 4.6: DELETE operation query execution

Chapter 5

Comparison

5.1 SIMPLE SELECT OPERATION AND COMPLEX SELECT OPERATION COMPASION

Records in milieconds	1000 records	10,000 records	100,000 records	500,000 records
PostgreSQL Simple Select	1	1	1	1
PostgreSQL Complex Select	1	1	1	1
MongoDB Simple Select	3	3	3	3
MongoDB Complex Select	3	3	3	4

Table 5.1: Select Operation Comparison time table

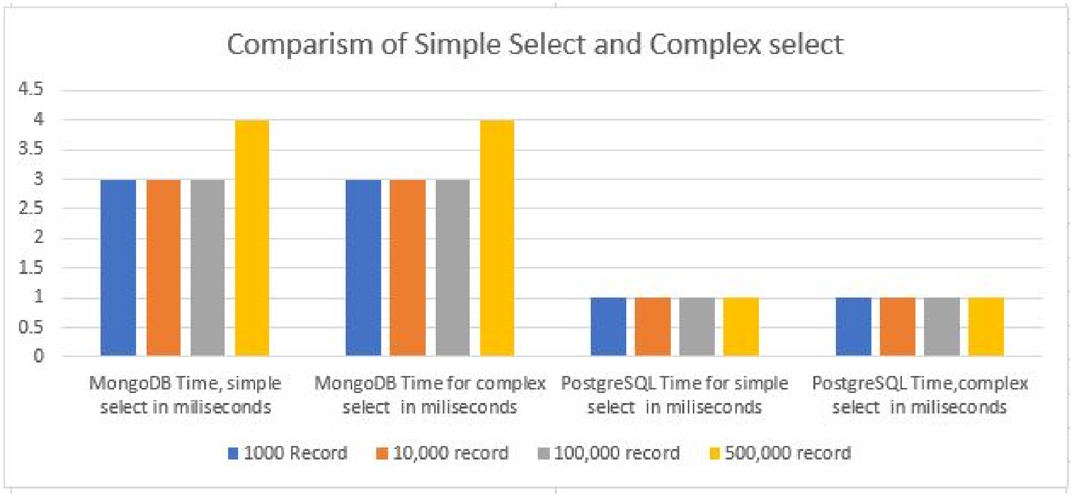


Figure 5.1: Comparison between Simple SELECT and Complex SELECT

In comparison of simple select and complex select operation, we can see that for simple operation PostgreSQL took 1 milliseconds time for 1000 record and for complex select it also took 1 milliseconds for 1000 records. As shown in the table PostgreSQL took 1 milliseconds for performing the select and complex select operation for all 10,000 data, 100,000 data, 500,000 data. On the other hand MongoDB took 3 milliseconds for entering 1000 data for simple select operation and for performing complex selet operation it took 3 milliseconds. For 10,000 data, 100,000 data, 500,000 data respectively it took 3 milliseconds for performing simple select operation and took 3 milliseconds for performing complex operation till 100,000 records. But MongoDB took 4 milliseconds to perform complex select opertation which is 1 milliseconds slower then simple select operation time. Here we can see that PostgreSQL took less time then MongoDB for both select and Very complex select operation. So PostgreSQL is performing better than MongoDB.

5.2 SIMPLE SELECT OPERATION AND VERY COMPLEX SELECT OPERATION COMPASION

Records in milieconds	1000 records	10,000 records	100,000 records	500,000 records
PostgreSQL Simple Select	1	1	1	1
PostgreSQL Very Complex Select	1	1	1	1
MongoDB Simple Select	3	3	3	3
MongoDB Very Complex Select	4	4	4	4

Table 5.2: Select Operation Comparison table

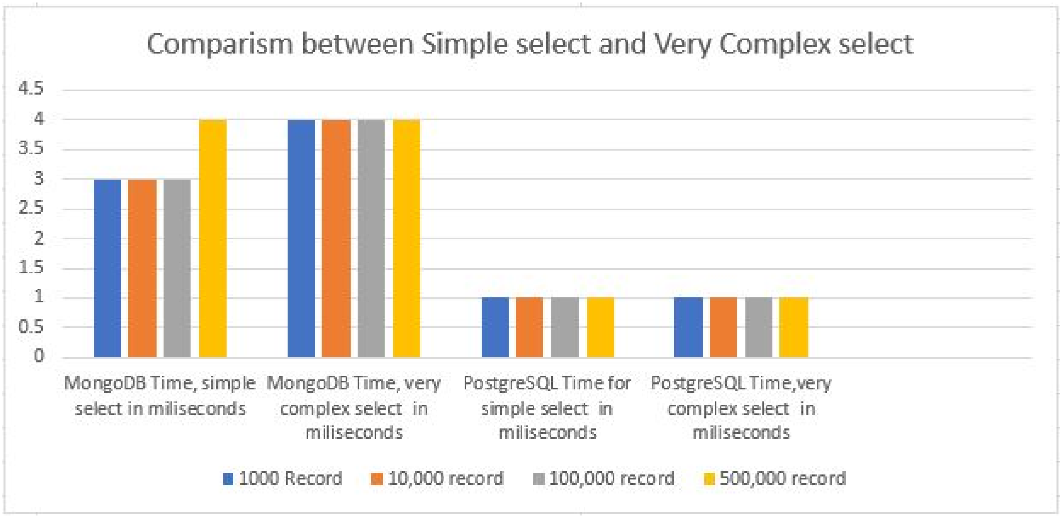


Figure 5.2: Comparison between Simple SELECT and Very Complex SELECT

In comparison of simple select and very complex select operation, we can see that for simple operation PostgreSQL took 1 milliseconds time for 1000 record and for very complex select it also took 1 milliseconds for 1000 records. As shown in the table PostgreSQL took 1 milliseconds for performing the operation for all 10,000 data, 100,000 data, 500,000 data. On the other hand MongoDB took 3 milliseconds for entering 1000 data for simple select operation and for performing very complex selet operation it took 4 milliseconds which is 1 milliseconds slower then simple select operation. For 10,000 data, 100,000 data, 500,000 data respectively it took 3 milliseconds for performing simple select operation and took 4 milliseconds for performing very complex operation. From the above states tics we can see that PostgreSQL took less time then MongoDB to perform. We can state that mongoDB is not performing better than PostgreSQL

5.3 RESULT ANALYSIS

From the experiment, insert operation is faster in MongoDB database then the PostgreSQL database. From our own perspective, it is expected to us because MongoDB database insert and store data in objects documented way. So, when we push large amount of JavaScript objects to the MongoDB database, it processed those objects and stored faster way than PostgreSQL.

The update, delete, select operations is faster in PostgreSQL than in MongoDB. It is because, data is stored in PostgreSQL structured way. SQL database performs fast and better performance when data is stored in structural relation matter. For that reason, PostgreSQL performs fast in select, update, join and delete operations and fast processing speed on those operations.

Designing enterprise level solution and millions of user will use their products or services and big amount of data is generated, and those data that have to be processed or operate frequently in a database or process multiple concurrent requests from the database like as our experiment as we processed large amount of complex queries in our system and measured time, so, if the application's model of data or types of data is structured and maintained in a relational model, SQL database in our example provide better performance.

On the other hand, when the big data follows more unstructured model or not obvious, then NoSQL database give better performance in their system.

So, the goal is to provide the ability of decision making in creating big data application for better performance to their customers or users. In this way, the research helps to make appropriate decision as well as give better performance on the application with great user experience.

Chapter 6

Conclusion

The thesis provides the analysis of MongoDB and PostgreSQL databases. Besides it provides comparative performance results for insert, select , update and delete operations for simple to complex queries. Through this experiment PostgreSQL database is faster than MongoDB for update, delete and read operations. But on the other hand, insert operation is faster in MongoDB database than PostgreSQL database. MongoDB performs well when data is unstructured and can easily be inserted as objects which is in the document form.

Chapter 7

Future Works

There are still a lot of room for imporvement keeping aside above works. For improving the performance of MongoDB then designing an unstructured data model is required. On the other hand, PostgreSQL provides better performance in structured data models. Besides, PostgreSQL performs well in the join operation to find the desired output by joining multiple tables. So, both database performs well depends on the data structure or type of data in the application system. There may be some limitations regarding system configuration, internet speed for getting the desired output. It is useful when we analyze the system, the types of data operation and features complexity to make a decision that which database is useful. So thinking about every operation and feature of a big data application to determine which database can solve the problem efficiently. The result can create an impact in big data applications having a huge collection of data that have to perform in the database. So, the thesis provides the decision making steps when choosing a right database in the big data generating application

Bibliography

[1] T. K. Sellis, "Global query optimization," in *Proceedings of the 1986 ACM SIGMOD international conference on Management of data*, 1986, pp. 191–205.

[2] G. Leptoukh, "Nasa remote sensing data in earth sciences: Processing, archiving, distribution, applications at the ges disc," in *Proc. of the 31st Intl Symposium of Remote Sensing of Environment*, 2005.

[3] D. Pritchett, "Base: An acid alternative: In partitioned databases, trading some consistency for availability can lead to dramatic improvements in scalability.," *Queue*, vol. 6, no. 3, pp. 48–55, 2008.

[4] N. Leavitt, "Will nosql databases live up to their promise?" *Computer*, vol. 43, no. 2, pp. 12–14, 2010.

[5] Z. Wei-Ping, L. Ming-Xin, and C. Huan, "Using mongodb to implement textbook management system instead of mysql," in *2011 IEEE 3rd International Conference on Communication Software and Networks*, IEEE, 2011, pp. 303–305.

[6] A. Boicea, F. Radulescu, and L. I. Agapin, "Mongodb vs oracle–database comparison," in *2012 third international conference on emerging intelligent data and web technologies*, IEEE, 2012, pp. 330–335.

[7] V. Abramova and J. Bernardino, "Nosql databases: Mongodb vs cassandra," in *Proceedings of the international C* conference on computer science and software engineering*, 2013, pp. 14–22.

[8] S. Khan and V. Mane, "Sql support over mongodb using metadata," *International Journal of Scientific and Research Publications*, vol. 3, no. 10, pp. 1–5, 2013.

[9] S. H. Aboutorabi$^{\underline{a}}$, M. Rezapour, M. Moradi, and N. Ghadiri, "Performance evaluation of sql and mongodb databases for big e-commerce data," in *2015 International Symposium on Computer Science and Software Engineering (CSSE)*, IEEE, 2015, pp. 1–7.

[10] S. Chickerur, A. Goudar, and A. Kinnerkar, "Comparison of relational database with document-oriented database (mongodb) for big data applications," in *2015 8th International Conference on Advanced Software Engineering & Its Applications (ASEA)*, IEEE, 2015, pp. 41–47.

[11] C. Győrödi, R. Győrödi, G. Pecherle, and A. Olah, "A comparative study: Mongodb vs. mysql," in *2015 13th International Conference on Engineering of Modern Electric Systems (EMES)*, IEEE, 2015, pp. 1–6.

[12] M.-G. Jung, S.-A. Youn, J. Bae, and Y.-L. Choi, "A study on data input and output performance comparison of mongodb and postgresql in the big data environment," in *2015 8th International Conference on Database Theory and Application (DTA)*, IEEE, 2015, pp. 14–17.

[13] J. Pokorn, "Database technologies in the world of big data," in *Proceedings of the 16th International Conference on Computer Systems and Technologies*, 2015, pp. 1–12.

[14] E. Andersson and Z. Berggren, *A comparison between mongodb and mysql document store considering performance*, 2017.

[15] A. Oussous, F.-Z. Benjelloun, A. A. Lahcen, and S. Belfkih, "Nosql databases for big data," *International Journal of Big Data Intelligence*, vol. 4, no. 3, pp. 171–185, 2017.

[16] W. Puangsaijai and S. Puntheeranurak, "A comparative study of relational database and key-value database for big data applications," in *2017 International Electrical Engineering Congress (iEECON)*, IEEE, 2017, pp. 1–4.

[17] M. Sharma, V. D. Sharma, and M. M. Bundele, "Performance analysis of rdbms and no sql databases: Postgresql, mongodb and neo4j," in *2018 3rd International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE)*, IEEE, 2018, pp. 1–5.

[18] M. Karimi, "Benchmarking of rdbms and nosql performance on unstructured data," Ph.D. dissertation, 2019.

[19] A. Makris, K. Tserpes, G. Spiliopoulos, and D. Anagnostopoulos, "Performance evaluation of mongodb and postgresql for spatio-temporal data.," in *EDBT/ICDT Workshops*, 2019.

[20] B. S. Shetty and K. Akshay, "Performance analysis of queries in rdbms vs nosql," in *2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, IEEE, vol. 1, 2019, pp. 1283–1286.