Comparing the databases MSSQL and MongoDB for the web-based environment Ozlab

Anthony Nokrach

Faculty: Faculty for Health, Science and Technology

Subject: Computer Science

Points: 15 hp

Supvervisor: Leonardo Iwaya

Examiner: Johan Garcia

Januari 2018

En jämförelse mellan databaserna MSSQL och MongoDB för den webbaserade miljön Ozlab

Anthony Nokrach

This report is submitted in partial fulfillment of the requirements for the Bachelor's degree in Computer Science. All material in this report which is not my own work has been identified and no material is included for which a degree has previously been conferred.

Anthony Nokrach

Approved, 16-01-2018

Advisor: Leonardo Iwaya

Examiner: Johan Garcia

Acknowledgments

I would like to thank my supervisor Leonardo Iwaya, course coordinator Katarina Asplund and family members for the great feedback and encouragement during the process of this project.

Abstract

The debate over whether software system should use a NoSQL or a SQL database, have been going on for a long time, however there are still not that many studies defining which of the two databases that is the preferable one. This essay presents a comparative analysis of the databases MSSQL and MongoDB for the web-based environment Ozlab. Ozlab allows users and developers to experiment with graphical user interfaces to obtain acceptable design solutions. We compared MSSQL and MongoDB based on the basic CRUD operations on both the databases, evaluating their performance and sustainability with respect to the Ozlab environment. This analysis was based on existing research experiments and my own experience using the MongoDB and MSSQL databases. As a conclusion, our result show that MSSQL is the most suitable database for the Ozlab environment.

Contents

List of Figures

List of tables

1 Introduction

Current applications and software systems place more demands for reliable and flexible databases to manage large and medium size datasets. Most information system can have their data structured in typical relational databases, but upcoming systems require changes in the data structure paradigm. This is the case of “big data” applications that handled large amounts of unstructured data. In this thesis, the MSSQL (Microsoft Structured Query Language) and the MongoDB (Structural database), are analyzed and compared in the context of the Ozlab software system. Ozlab is a tool for designing user interfaces, that allows users and developers to easily improvise and test a variety of Graphical User Interfaces(GUIs), being useful for discussing interaction with stakeholders and for demonstrating the intended behavior to the software developers. Given that this research aims to evaluate which of these two databases is the most suitable for the Ozlab environment regarding the database performance for their most used operations (i.e., selection, insertion, update and deletion) and compatibility with the existing integrated development environment (i.e., Microsoft Visual Studio)

This bachelor thesis is structured as follows. The motivation and goals of this thesis are introduced in section 2. The relevant background about the Ozlab project, MSSQL and MongoDB are discussed in section 3. The comparative analysis between the MSSQL and MongoDB is presented in section 4, followed by the results in section 5. Lastly the discussion about the advantages and disadvantages with using MSSQL and MongoDB for the Ozlab environment, is presented in section 6. Followed by the conclusion about the project, in section 7.

2 Motivation and Goals

This project was initiated by the research group within KAU's department of Information Systems, also main developers of Ozlab. Since the Ozlab software system contains information of various kinds of datatypes, the developers of Ozlab would like to verify if an alternative database would be more efficient for accessing, inserting, updating and deleting data. Therefore, the goals for this project is to compare the performance of using the MongoDB(NoSQL) and MSSQL(SQL) databases for the Ozlab environment, where Ozlab today are using the SQL database for managing its information.

2.1 Task

The task is to study and compare two databases, MSSQL (a structured database) and MongoDB (a non-structured database), to provide a consistent recommendation about the best alternative for managing the information for the Ozlab environment. To do so, this analysis is based on the review of existing comparative studies regarding relational and non-relational databases, as well as our own benchmarks for the MongoDB and MSSQL regarding conventional database operations (i.e., select, insert, update and delete)

2.2 Expectation

The expectation for this project was that using the MongoDB for the Ozlab environment would make it simpler for accessing, insert and managing the data for Ozlab project because of the non-structural design of MongoDB which often is used for managing and sorting databases for systems that operates with larger amount of data.

2.3 Motive

This project was selected to enhance and develop the student's skills and knowledge. The exercise of a solid comparative analysis of technologies for real and fully developed systems is

recurrent practice in the industry that is constantly required to update its information system. In this project, this is done by analyzing databases for applications and software systems, considering the importance role of choosing an effective database. Lastly, we believe that by researching and practicing with structural and non-structural databases such as the MSSQL and MongoDB is also beneficial during the student's career in information systems.

2.4 Related Work

For this project, the benefits of using a SQL and a NoSQL database for software applications in the operations of updating, accessing and inserting data where analyzed. Given that, SQL language was designed for a structural and relationship-based database, this kind of databases are often used in software systems with well-defined data structures and that operate with low/middle data volumes. On the other hand, MongoDB is a non-structural and non-relational database that is often used in systems and applications that handle and store vast volumes of data. For instance, MongoDB uses the map-reduce technique for processing large volumes of data [3] [4].

Analyzing and comparing databases is important when adapting software systems/applications to new databases that are available on the market. In [3], the authors provide a comparison between an SQL and NoSQL databases (i.e., MongoDB) for a modest size structured system, by evaluating the processing time of operations such as select, insert and update. The experiment consists of four tests with 100 runs each, where they conclude that the MongoDB have a superior performance for insertions, updates and simple queries, while instead the SQL performed better when updating the non-key attributes and aggregate queries.

Similarly, in [4] the authors compare MongoDB and SQL databases by using a MongoDB database application for the Taiwan Water Corporation, where they applied the Map Reduce method for processing the information, by comparing the insert, reading and search performance. In the report they concluded that the MongoDB does have a better efficiency than the SQL database on these areas. The authors believe that the future trend for non-structural databases will be based on integration both on new technologies and between other structural databases, for better managing of application with a large amount of data. As in [3], have the

researchers in [27] estimated the performance between SQL and NoSQL databases. In this case, the researchers estimated have the time (in milliseconds) taken to instantiate, write, insert, fetch and delete data on a set of structured and non-structural databases.

3 Background

This section presents the relevant technologies and systems used and analyzed throughout the research, i.e., (1) the Ozlab environment, and the database (2) MSSQL, and (3) MongoDB. All the technologies are briefly explained, and their main characteristics emphasized, especially regarding requirements on data handling and sorting, as well as the existing integration of Ozlab with MSSQL and the possible use of MongoDB.

3.1 Ozlab

The Ozlab system is a GUI web-based environment that manages user interfaces and a platform for developers to prototype their own design. The Ozlab environment enables for external partners (researchers and practitioners) to experiment and elaborate with pre-implemented (unwritten code) user interfaces before starting the implementation of general code.

The research group at Karlstad University's department of Information Systems is responsible for the design and development of the Ozlab environment. Ozlab, which has it origin from the "Wizard-of-Oz" project, uses a method that allows the user to believe that he/she implements their functions with the program, when they are only communicating with a test-manager that action-responds to the user. The Ozlab system can be used by researchers/users both at home or at their own respective workplaces [6]. Currently the Ozlab system utilizes the MSSQL database for managing its data.

The **Figure 1** below displays the first GUI components of the Ozlab environment.

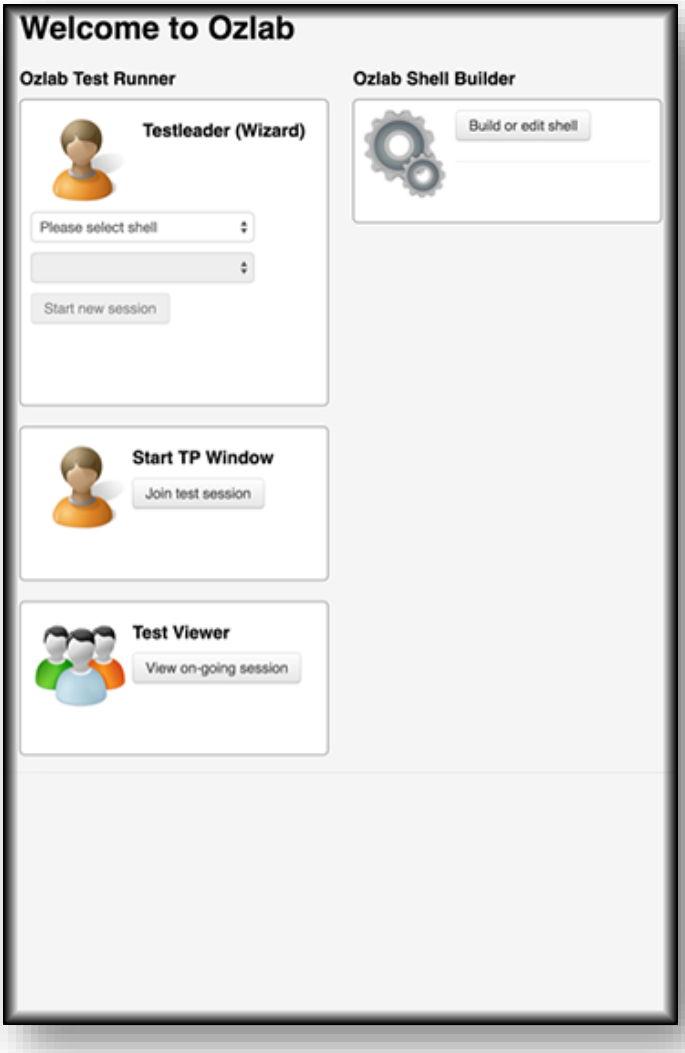

Figure 1 The GUI page of the Ozlab environment.

3.2 MongoDB

MongoDB is a document-oriented database that stores its data in JSON file objects, where the programmer/developer conducts script commands through command lines on windows or Linux (e.g., shell CLI) to manipulate and search the information in the database. MongoDB differs from the SQL database, because instead of using a data-structure of tables, row and columns when handling the data, it uses collections instead of tables and documents instead of rows. The MongoDB database organizes its information through key-value pairs in which uniquely primary keys are used to identify and pull data from the database [1]. The MongoDB is designed and often used for applications and software systems with large volumes of non-structured data/information [1][3].

The **Figure 2** displays an image of a MongoDB database in windows terminal that possesses a small amount of data.

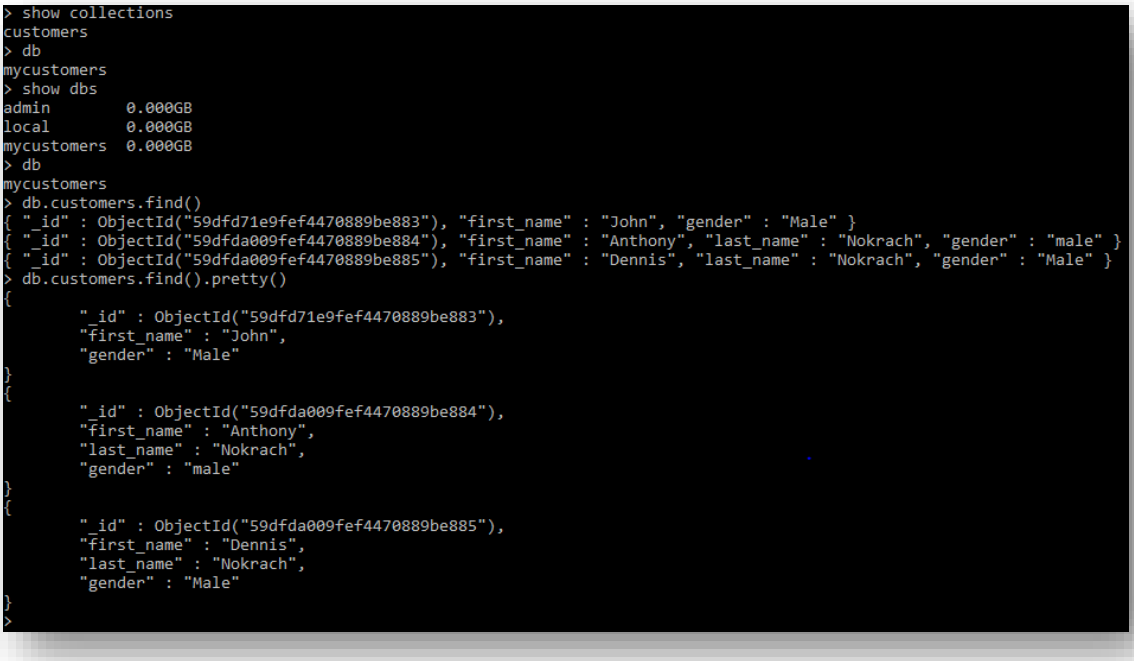

Figure 2 A picture demonstrating a running MongoDB with documents representing data/information.

3.3 MSSQL

MSSQL is a proprietary software created by Microsoft Inc that uses the SQL language, which normally is used when developing analytic statistic for applications and business intelligence systems. MSSQL is a relational database, meaning that the handling of the information in the database is based on the relationship the data stored in memory-optimized tables have for each other [2]. SQL database is built in a way to capture the semantics of the database, where the datatypes, characteristics and format are grouped together which makes the database structured. The **Figure 3** displays an example of an SQL database that contains tables with columns and rows representing data.

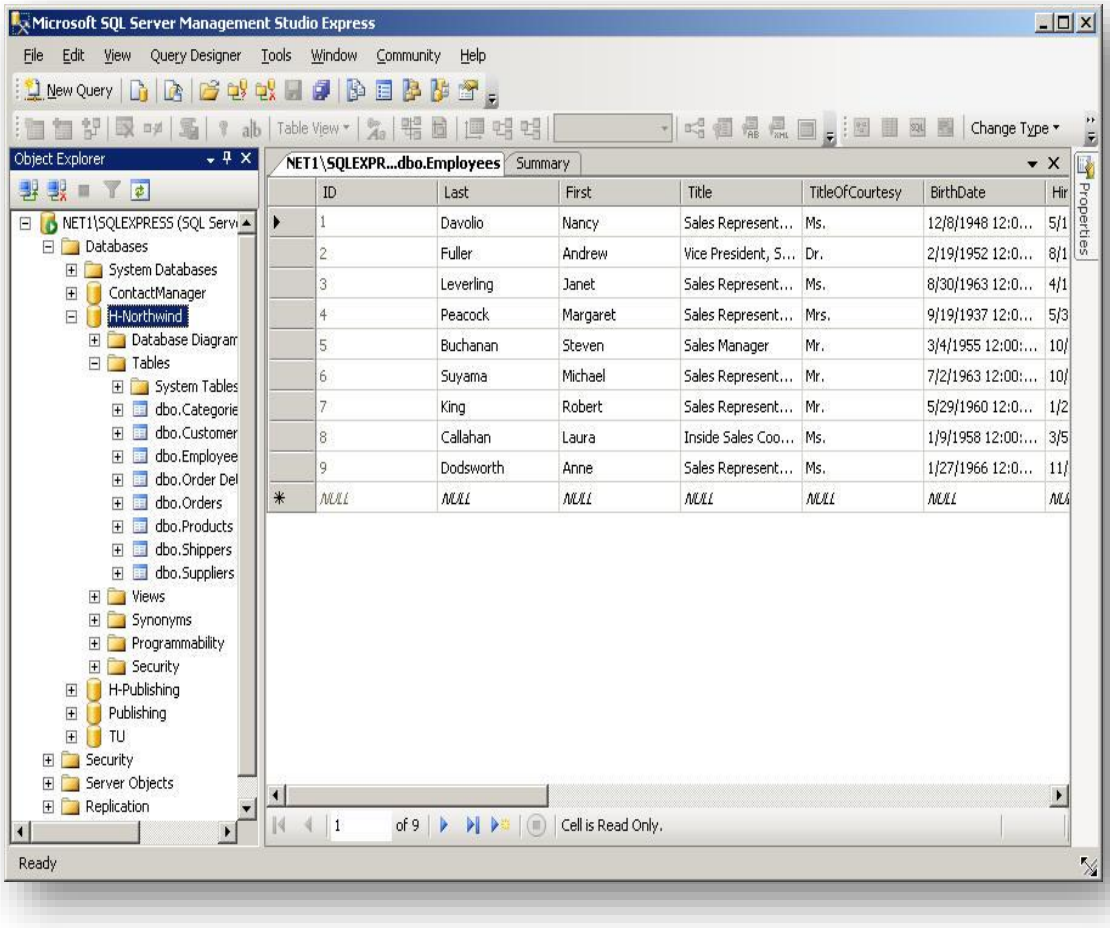

Figure 3 A picture demonstrating a running MSSQL database with table, columns and rows.

4 Comparative Analysis

The importance and understanding of data/information is critical when analyzing databases. This section presents the fundamentals of operating with CRUD on the MSSQL and MongoDB database, (1) MSSQL and the integration process with Microsoft Visual Studio, (2) MongoDB and the integrations process with Microsoft Visual Studio, (3) The CRUD operations on MSSQL database, (4) The CRUD operations on MongoDB database, (5) Comparison table between MongoDB and MSSQL.

4.1 MSSQL

MSSQL often called SQL Server is a proprietary software with its first launch 1989 by Microsoft Inc. MSSQL is a relational database that organize the data in a structural order with a set of tables that contains data in a predefined category [22]. The tables in a SQL database are structured in a way that the tables contain columns which includes a set of data values of a certain datatype. For example, numeric and textual data, that is stored in rows for which are defined in the columns. [22] [23].

4.1.1 MSSQL and Microsoft Visual Studio

Both the programming platform Visual Studio and the MSSQL database are created by Microsoft, which facilitates for an effective cooperation between these two platforms.

Connection to the database.

When connecting a project in Visual Studio to the MSSQL database, you will first need to create a database and a table in the SQL Server Studio Management. In **Figure 4**, I display an image where I have created a database in the SQL Server Management Studio with the database name ‘University ‘.

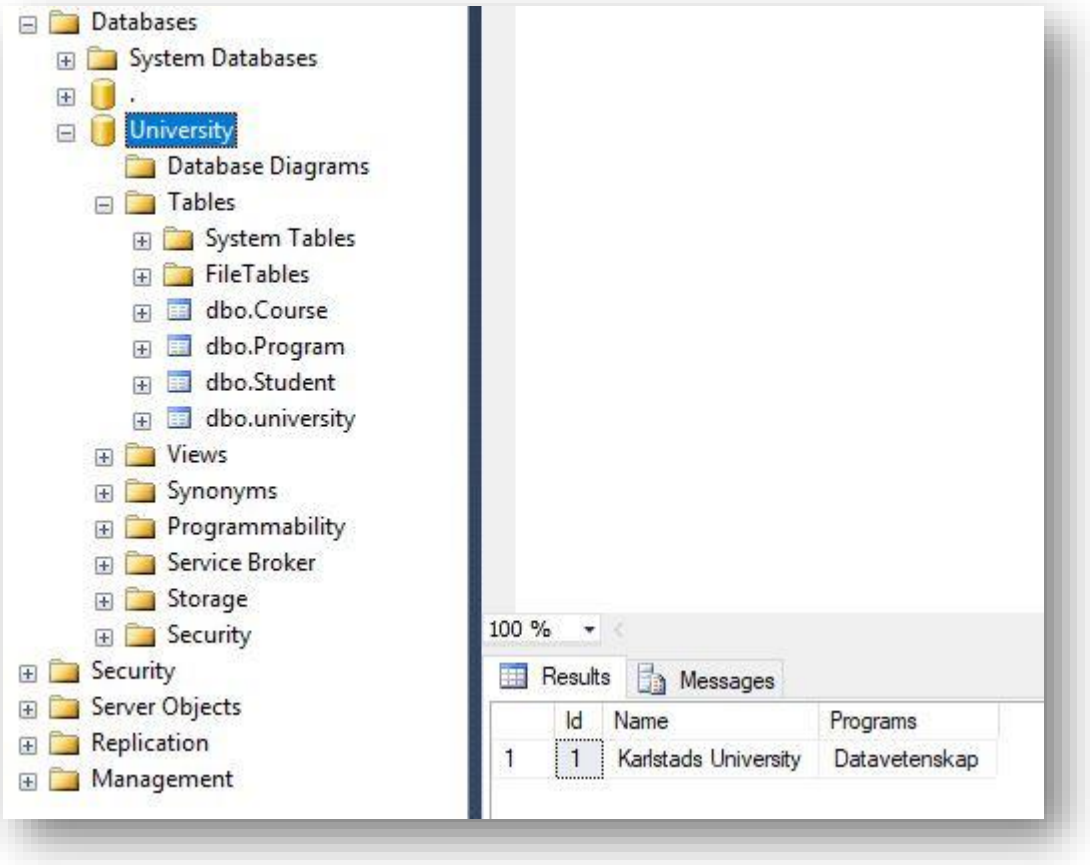

Figure 4 A created University database in the SQL Server Management Studio.

When a database has been created in the SQL, you can then connect the created database to the Visual Studio project through the SQL Server Explorer in Visual Studio.

In **Figure 5** I demonstrate an image where I have connected the database in SQL Server Management Studio to the Microsoft Visual Studio.

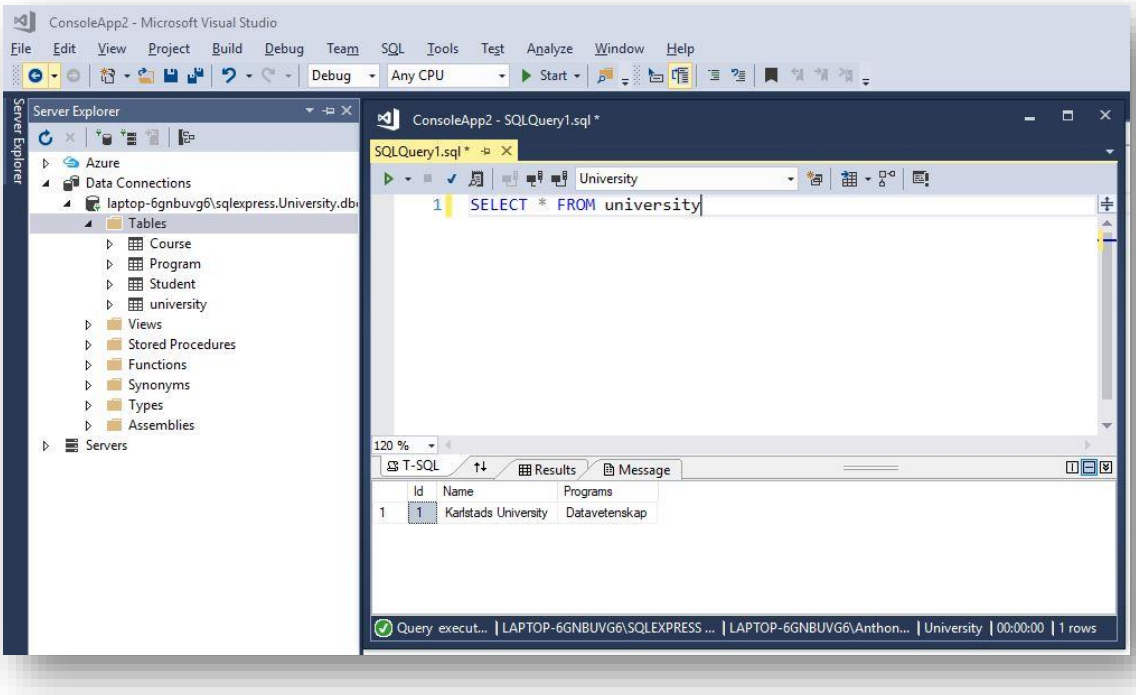

Figure 5 A created database(University) from SQL Server Management with Microsoft Visual Studio.

4.1.2 Basic CRUD operations in the MSSQL

CRUD which stands for Create, Read, Update and Delete are the basic functions used in a database (i.e., insert, select, update and delete). I will in the following subsections describe how the CRUD functions, operates in a SQL database.

4.1.2.1 Insert

The insert function in a SQL database adds a new record in table with a unique id/primary key by using the 'INSERT' statement. Where the columns in the created table has its own parameter [9] [10].

4.1.2.2 Select

The Read function in the database reads the primary keys on every table, which helps the database cooperate with the relationships methods used to find and read data that is generated in the database. Methods such as ‘SELECT’, selects and find data from the database.
[9] [12]

4.1.2.3 Update

The SQL update function renews the tables in the database based on the primary keys selected through the ‘WHERE’ statements in the SQL code [9].

4.1.2.4 Delete

The delete statement/function removes one or more selected rows in the SQL database through the ‘DELETE’ SQL statement. [9].

4.2 MongoDB

MongoDB is a non-structural database created by the MongoDB Inc, which stores its information in a document-oriented structure. The MongoDB is an open-source database published under the 'GNU Affero General Public License', which means that any users of the product can inspect, modify and enhance the platform for its own purpose, under the terms of MongoDB's policy.

4.2.1 MongoDB and Microsoft Visual Studio

Before you integrate the MongoDB to a project in the Visual Studio, there are some pre-adjustment that needs to be include in the process. First step is to create a data and a log folder in the MongoDB folder where the information generated in the database will be stored. Second step would be to create a directory to the data and log folder for the data to be stored by using the command lines, in my case the windows terminal. **The Figure 6** demonstrates an example of how a directory command in the windows terminal could look like.

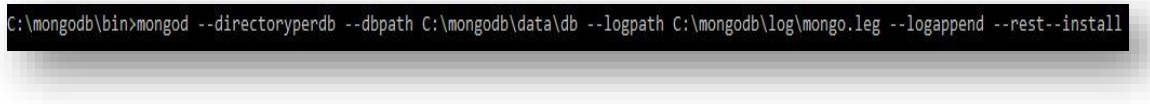

Figure 6 Directory command lines in windows terminal for data and log files.

The final step in the process is to connect the MongoDB database to the Visual Studio using implementation code in programming language C#. In **Figure 7,** I demonstrate how the connection implementation to the MongoDB database could like in C# Visual Studio. Where I in the example below create a university, program and student database in MongoDB by using C# code written in Visual Studio.

```
MongoClient client = new MongoClient();
var server = client.GetServer();
var db = server.GetDatabase("University");
var db1 = server.GetDatabase("Programs");
var db2 = server.GetDatabase("Students");
var collection = db.GetCollection<University>("univeristy");
var collection1 = db1.GetCollection<program>("program");
var collection2 = db2.GetCollection<Students>("students");
University un = new University();
program pr = new program();
Students st = new Students();
collection.Save(un);
collection1.Save(pr);
collection2.Save(st);
```

Figure 7 Connection code to the MongoDB database in the programming language C# from Microsoft Visual Studio.

```
MongoClient client = new MongoClient();
var server = client.GetServer();
```

Figure 8 Connection between the client and server.

The first two code lines in **Figure 8** represent the connection between the Mongo client and the server.

```
var db = server.GetDatabase("University");
var db1 = server.GetDatabase("Programs");
var db2 = server.GetDatabase("Students");
var collection = db.GetCollection<University>("univeristy");
var collection1 = db1.GetCollection<program>("program");
var collection2 = db2.GetCollection<Students>("students");
University un = new University();
program pr = new program();
Students st = new Students();
collection.Save(un);
collection1.Save(pr);
collection2.Save(st);
```

Figure 9 Creation of the MongoDB databases.

The image in **Figure 9** represents the connection to the MongoDB and the creation of the databases 'University', 'Programs', 'Students' which automatically will be created in the MongoDB, alongside with the collections 'university', 'program' and 'students' for the respective created databases.

4.2.2 Robo3T

Robo3T (formerly known as RoboMongo) is a free open-source GUI for the MongoDB database where the platform facilitates for developers to manage and update the data created in the application, instead of using the terminal to display the data in the MongoDB database [21]. In **Figure 10,** I show an image representing the Robo3T platform which contains a database, collection and documents created in Visual Studio.

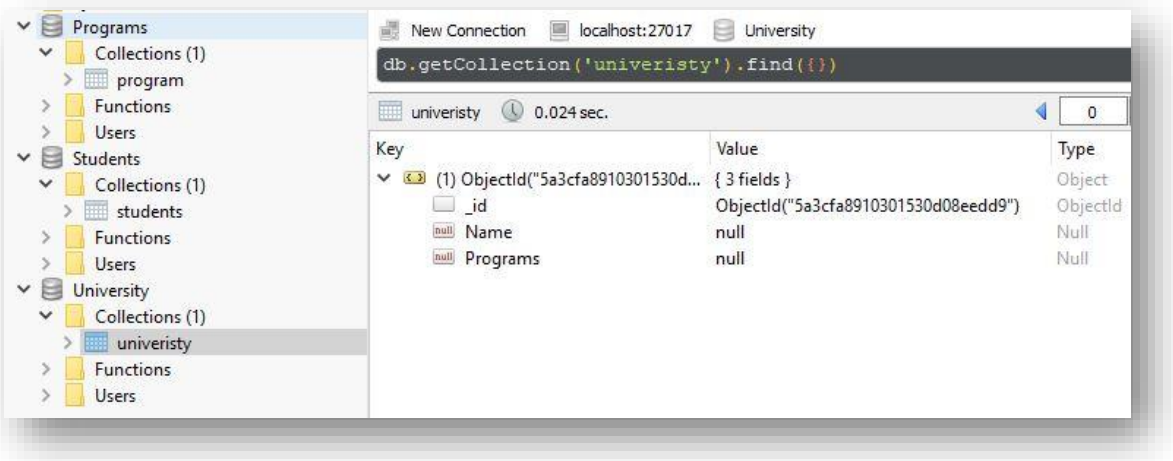

Figure 10 The Robo3T GUI platform displaying the 'University', 'Students' and 'Programs' databases.

4.2.3 CRUD operations in the MongoDB

I have in the previous subsections described how MSSQL operates with CRUD. I will in the following subsections, describe how the MongoDB functions with the CRUD operations.

4.2.3.1 Create

The insert function in MongoDB adds a document to a selected collection in the database. If it occurs a situation where there is no collection for an added document to the database, then the current document will become a collection. The following example of a code snip, implements a document to a collection: Db.collection.insertOne() and db.collection.insertMany(). [9]

4.2.3.2 Read

The read operations or queries in the MongoDB databases, selects documents from a single collection by retrieving data that are stored in the MongoDB. The database adds a query by using the db.collection.find() method/statement. The query process exists of a collection identifier, query criteria and a modifier.

4.2.3.3 Update

When the database performs an updating on its machine, then the machine updates the existing documents in a collection by methods with the example of db.collection.updateOne();, db.collection.updateMany() or db.collection.replaceOne() [9].

4.2.3.4 Delete

Delete functions removes documents in a collection by using the methods db.collections.deleteOne() or db.collection.deleteMany().

4.3 The Insert, Select, Update and Delete benchmarks on MSSQL and MongoDB

Analyzing the performance of the insert and update function is an important area when studying the efficiency of a database. In the project [3], the authors/developers made an experiment between a relational database SQL and the non-relation database MongoDB to determine the insert and update speed performance on the databases. The experiment consisted of three tables representing a department, user, project and an additional table that manages M: N, where queries involving more than one table requires a 'JOIN' function. All the objects in the experiment have a relation to each other.

4.3.1 The insert performance

To make the experiment more credible, the researches have made 100-200 runs for each of the separate tests. In **Figure 11** they compare the insert average time in milliseconds between a relational(MSSQL) and non-relational database (MongoDB) [3].

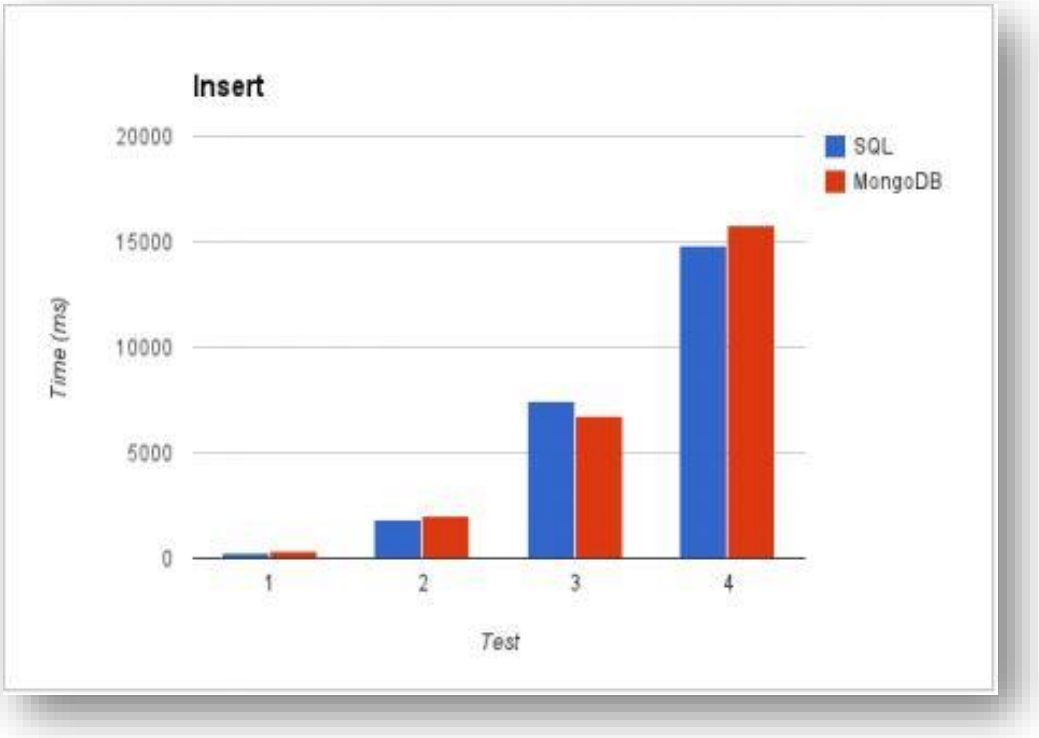

Figure 11 Diagram of the test result from the 'Insert' experiment in [3].

The **Figure 11** above shows that there is no considerable difference between using a relational and a non-relational database regarding the insert performance. The writing performance in [27] measures the time taken to write key-value pairs from the bucket on the respective databases. As shown in the Tables 1, 2 and 3 below, the time is taken on a set of NoSQL and SQL databases, where the number of operations refers to the number of times a given operation is executed in the test. I will in this case only focus on the MongoDB and MSSQL databases.

Database	Number of operations					
	10	50	100	1000	10000	100000
MongoDB	61	75	84	387	2693	23354
RavenDB	570	898	1213	6939	71343	740450
CouchDB	90	374	616	6211	67216	932038
Cassandra	117	160	212	1200	9801	88197
Hypertable	55	90	184	1035	10938	114872
Couchbase	60	76	63	142	936	8492
MS SQL Express	30	94	129	1790	15588	216479

TABLE III
TIME FOR WRITING (MS)

Table 1 Results from the 'Writing/Insert' performance in [27].

The result from the writing experiment in [27], shows almost the same result as the previous experiment in [3]. Where in this the case, the MongoDB performs better than its counterpart MSSQL as the number of operations increases.

4.3.2 The Select performance

The selecting performance is of fundamental importance both in a structural and non-structural database where the user selects a table/collection in a database to find a requested data. In this test the developers have divided the select operations into two categories: complex queries and simple queries. The *complex queries* involved multiple objects, aggregated functions and nested queries where the *simple queries* involved selecting data on only on object type [3].

In **Figure 12,** I show a diagram of the results from the selection speed test in milliseconds, which is based on names on the "Departments" created by the developers in [3].

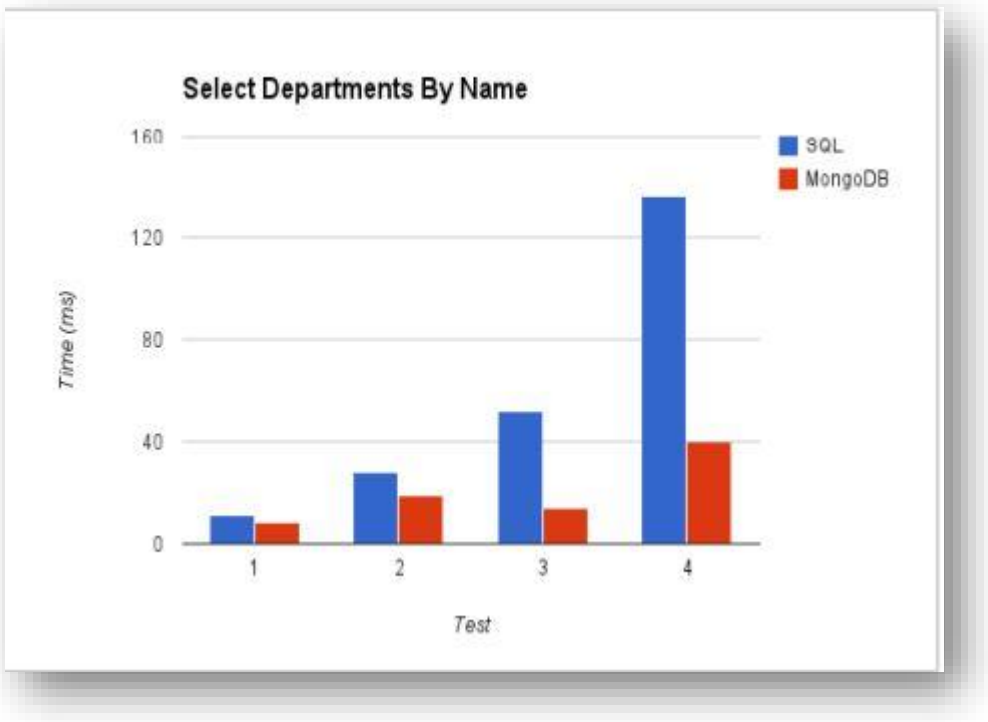

Figure 12 Diagram of the result from the 'Select departments' experiment in [3].

The diagram shows that the MongoDB outperforms the SQL database, which the researchers believes is due to the structure on how the two databases stores the data. MongoDB uses memory mapped files to store its data unlike the SQL database that must retrieve its data from the disk [3].

The reading performance in [27] measures the time taken to read values corresponding to given keys on the bucket from the respective databases. I will only focus on the MongoDB and MSSQL databases that are displayed in **Table 2** below.

Database	Number of operations					
	10	50	100	1000	10000	100000
MongoDB	8	14	23	138	1085	10201
RavenDB	140	351	539	4730	47459	426505
CouchDB	23	101	196	1819	19508	176098
Cassandra	115	230	354	2385	19758	228096
Hypertable	60	83	103	420	3427	63036
Couchbase	15	22	23	86	811	7244
MS SQL Express	13	23	46	277	1968	17214

TABLE II
TIME FOR READING (MS)

Table 2 Results from the 'Reading performance' experiment in [27].

Table 2, from the experiment in [27], shows almost the same results as in the previous example from [3], where MongoDB outperforms the MSSQL database, as the number of operations increases. Which I believe is due to the index used by the MongoDB and its use of memory [3].

4.3.3 Update performance

Non-relational/non-structural and relational/structural databases operate differently when it comes to the update performances. The SQL database operates with relations between the tables which often contains foreign keys, primary keys and other constraint to the tables for example UNIQUE and NOT NULL. [2]. The MongoDB(non-relational) stores its information using collection, meaning that the collections have no constraint and that fields can contain different datatypes [26].

Figure 13 shows a diagram displaying the 'Update' comparison between the MongoDB and the SQL database. The diagram shows the comparison of the update speed in milliseconds on "Projects" tables/collections based on the projects id references.

Figure 13 Diagram of the test result from the 'Update Projects by Id' experiment in [3].

The diagram shows that the MongoDB outperforms the SQL database when updating the databases based on the Id. The explanation for the result could be due to the MongoDB having a pre-built index on the primary on its data which is faster than SQL Servers primary key clustered index [3]. **Figure 14** shows where the MongoDB performs inferior to the SQL database. The explanation is because of the MongoDB unstructured nature, which leads to complex lockups on each data when updating item in the MongoDB [3]. The diagram shows the comparison of the update speed in milliseconds on the "Frist Name", which is not based on the id references.

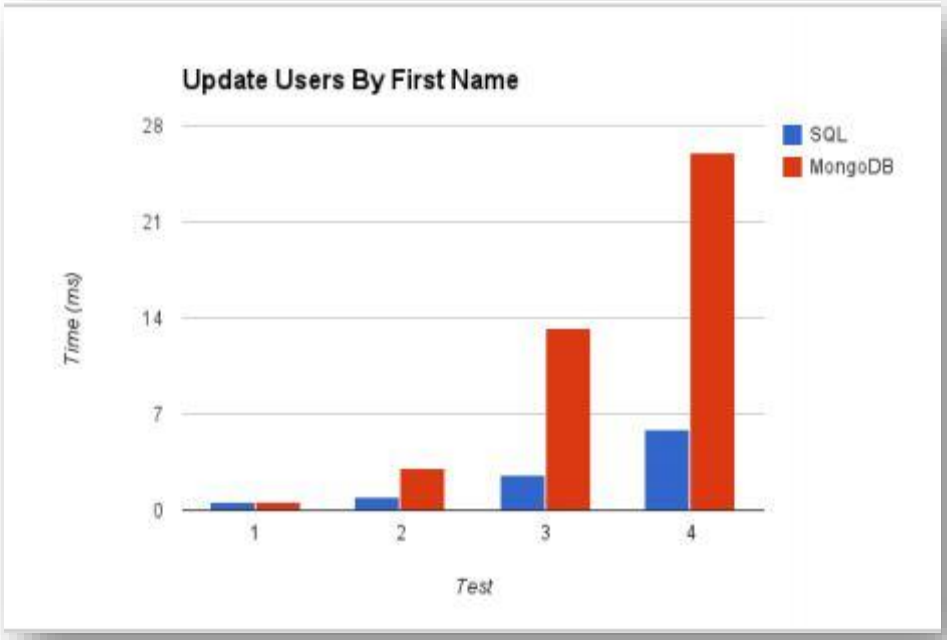

Figure 14 Diagram of the test result from the 'Update users by first name' experiment in [3].

Here the diagram in **Figure 14**, shows that the SQL database outperforms the MongoDB database, where the explanation for the result, is that the update experiment was on a non-indexed column. Which forces the MongoDB to perform complex lockups on the item because of the unstructured nature of the database [3].

4.3.4 The delete performance

Table 3, which represents the deleting experiment in [27], measures the time taken to delete key-values pair from a bucket in the respective databases on the table, where I only focus on the MongoDB and MSSQL databases.

Database	Number of operations					
	10	50	100	1000	10000	100000
MongoDB	4	15	29	235	2115	18688
RavenDB	90	499	809	8342	87562	799409
CouchDB	71	260	597	5945	67952	705684
Cassandra	33	95	130	1061	9230	83694
Hypertable	19	63	110	1001	10324	130858
Couchbase	6	12	14	81	805	7634
MS SQL Express	11	32	57	360	3571	32741

TABLE IV
TIME FOR DELETING (MS)

Table 3 Result from the 'Delete performance' experiment in [27].

The *Table 3* shows that the MongoDB database has a better deleting performance with both small and large number of operations than its counterpart MSSQL database.

4.4 Comparison tables

In this section I compare the fundamental differences between MongoDB and MSSQL, by using a comparison tables. In *Tables 4 and 5* shows a comparison of the fundamentals between the MongoDB and MSSQL.

MongoDB	MSSQL
None structural	Structural
None-relational	Relational
Collection	Table
Fields	Columns
Documents	rows

Table 4 Comparison of structure.

MongoDB	MSSQL
High write loads	Low writes loads
Big data	Low data
Not Microsoft owned	Microsoft owned
Longer setup	Shorter setup
Free and Open-Source	Proprietary and Closed - Source

Table 5 Comparison of user related aspects.

Table 4 compares the areas on the structure between the two databases. For example, the content of MSSQL is structured and relational, which enables the user to manage and manipulate the data that the database possesses. MongoDB on the contrary, is non-structured and non-relational which facilitates for its main purpose of storing large volume of data. *Table 5* compare the user related aspects between MSSQL and MongoDB. For example, The MSSQL is owned and developed by Microsoft, which facilitates for a shorter setup, since the Ozlab project exists in the Microsoft owned platform Visual Studio. MongoDB is open-source database, while MSSQL is not.

5 Results

This section presents the results and experiences from using the CRUD operations on MSSQL and MongoDB. (1) The create performance on MongoDB and MSSQL, (2) the read performance on MongoDB and MSSQL, (3) the update performance on MongoDB and MSSQL, (4) the delete performance on MSSQL and MongoDB.

5.1 Create (Insert/Writing)

Own experience on MSSQL

The insert procedure in the MSSQL database do not require much effort implementing, and because the MSSQL and Microsoft Visual Studio both are created by Microsoft, leading to a more efficient cooperation and interaction between the two platforms. One example is the SQL platform in Visual Studio where SQL language statements can be implemented. The insert procedure in MSSQL is performed using "INSERT INTO TableName (the identification of the effected table) and 'VALUE' (the specified data that will be inserted)" statements.

Own experience on MongoDB

The insert process in the MongoDB database requires almost the same amount of effort implementing as for the MSSQL database. The MongoDB requires more pre-installations processes between the MongoDB and Microsoft Visual Studio than its counterpart MSSQL in terms of directory commands in the terminal and folder creation for data and log files to be stored. The insert statement for document in MongoDB is "db.collection.insert({})."

Existing results from the literature

In reference with the insert speed experiment in [3], shows the research results, where the MSSQL have almost the same insert speed performance in comparison with the non-structured MongoDB database. The results from another insert experiment in [27], shows that the MSSQL performs inferior to the MongoDB database, as the number of operations and data storage grows larger.

The results from [3] are based on the average insert speed in milliseconds between MSSQL and MongoDB.
The results from [27] are based on the time taken to write key-value pairs to a bucket on a set of NoSQL and SQL databases

5.2 Select/Read

Own experience on MSSQL

The reading and selecting procedures in MSSQL are quite structured and self-explanatory. The information in the MSSQL database is divided into tables(Title), columns(Subtitles) and rows(data), which facilitates unexperienced user to understand the concept of the database. The select process in SQL is performed using "SELECT column (the name of the column) FROM TableName (the name of the effected table);"

Own experience on MongoDB

Because of the non-structured nature of MongoDB, the way of reading data in the database can be quite complex, when finding and searching for information in the database. To read a specific data requires a searching for a whole document in a collection, by using the "db.collection.find();" statement.

Existing results from the literature

The select experiment in [3], shows where the MSSQL database performance a slightly worse in comparison with the non-structured MongoDB. The test results in [27], shows that the MongoDB have a slightly more efficient reading performance than its counterpart MSSQL.

The results in [3] are based on the average time to perform select queries on a modest size database based on the items Id.
The results in [27] are based on the time taken to read values corresponding to given keys on the buckets in the databases.

5.3 Update

Own experience on MSSQL

The update procedure on the SQL language in MSSQL is very self-explanatory, is performed using: “Update TableName’ (which indicates the table that is going to be updated)”, “SET (the data that is going to be updated)” and the “WHERE”, to declare a more precise update localization, usually the Id for the row in the table.

Own experience on MongoDB

Due to the non-structured nature of the MongoDB, is it important that the user identifies the Objects Id reference when updating a collection. But the procedure is still just as self-explanatory as its counterpart MSSQL and performs equally well as the MSSQL database on a small data-storage. The update procedure in MongoDb is performed using: db.collection.update();

Existing results from the literature

The update performance experiment in [3], shows where the MSSQL have a more efficient update performance in comparison with its counterpart MongoDB, when updating data which do not required an update procedure based on the items Id references.

The update experiment in [3], also shows that the MongoDB surpasses the MSSQL when the update performance involves using the primary keys.

5.4 Delete

Own experience on MSSQL

Based on my own experience operating with MSSQL, my indications are that the delete operation performs equally well as the MongoDB on a small data-stored database. The delete procedure in MSSQL exists of “DELETE FROM TableName WHERE (The data which is going to be deleted)”

Own experience on MongoDB

The results from my own interaction with the MongoDB database, is that the deleting process performs equally well as the MSSQL on a small data-stored database. The delete procedure in MongoDB is performed using: “db.collection.deleteOne(); ”

Existing results from the literature

Based on the results from the delete experiment in [27], which is based on the time taken to delete key-value pairs from a bucket of data. Shows that the deleting performance in MSSQL have a slight inferior performance compared to MongoDB.

6 Discussion

In this chapter, I discuss the advantages and disadvantages for using the two databases for the Ozlab environment. I will analyze the areas where the two databases can make a difference for the Ozlab project, in terms of flexibility and performance, that can be of a benefit when further developing the Ozlab environment.

This chapter starts with, (1) the advantages and disadvantages with using MSSQL database, followed by, (2) the advantages and disadvantages with using MongoDB database, and (3) the conclusion of which of the two databases that would most appropriate for the Ozlab environment.

6.1 The MSSQL

6.1.1 The advantages with MSSQL

The advantages with using MSSQL database is the relationship-based structured. That enables the system to use the SQL statements to manipulate tables and columns to get a more precise and satisfactory match of the information, that exist in a MSSQL database.

6.1.1.1 Visual Studio

The MSSQL and Visual Studio are both created by Microsoft, allowing an effective cooperation between the MSSQL database and the Ozlab project, which is implemented in the .Net Visual Studio. The benefits with this cooperation is the small amount of connection code(C#) that is needed between the MSSQL database and Microsoft Visual Studio programming platform.

6.1.1.2 Software System

Based on the scientific research, some developers have indicated that the MSSQL database functions more efficiently with software systems that do not maintain large amount of data. An advantage for the Ozlab project, which for the moment do not need to handle larger volumes of information.

6.1.1.3 Functionalities

The MSSQL database has another advantage over MongoDB in the areas of searching functionalities where the structured nature of MSSQL allows broad query statements in SQL language to manipulate the data in the database.

Some examples of the SQL queries statements used on my own created database, that demonstrate the benefits of using SQL when manipulating the information in the MSSQL database are:

Inner-Join – "SELECT Student.Name, Program.CourseName From Student INNER JOIN Program on Student.ID = Program.ID;"

Which allows the database to connect two or more tables, in this case a connection of the 'Student' and 'Program tables,' which enables for a more precise, satisfactory and readable searching match.

Full-join – "SELECT Program.CourseName, Student.Name FROM Program FULL OUTER JOIN Student ON Student.ID = Program.ID; "

Full-join statement returns all the data in the database that have a match with a value in either the Student.ID table or the Program.ID table.

Union – SELECT Program.Id FROM Program UNION ALL SELECT Student.Id FROM Student;

The Union combines two or more selects statements in the table, in this case the program and student table.
As shown on the examples above, shows the benefits of using the SQL queries to manipulate the data in the MSSQL.

6.1.1.4 The update performance in MSSQL

In **Figure 14** taken from the research experiment in [3], shows the result where MSSQL have a superior update performance in comparison with the MongoDB database.

6.1.2 The disadvantages with MSSQL

I have in the previous subsections discussed the advantages with using the MSSQL in relation with the Ozlab environment. In this section, I will discuss the disadvantages with MSSQL.

6.1.2.1 Unstructured data

MSSQL and relational databases in general does not function well when collecting large volume of unstructured information that can be of any datatype [3].

6.1.2.2 The select performance in MSSQL

In **Figure 12** and *Table 2* taken from the research experiment in [3] [27], shows the result where the MSSQL have an inferior selecting performance in comparison with its counterpart MongoDB.

6.1.2.3 The update performance in MSSQL by id

In **Figure 13** taken from the research experiment in [3], does it show the result where the MSSQL have an inferior update performance in comparison to the MongoDB.

6.2 The MongoDB

6.2.1 The advantages with MongoDB

Although that the NoSQL MongoDB is new to the market, the database is increasingly being consider as a viable alternative to the MSSQL. In this section, I discuss the advantages of using the MongoDB database in relation with the Ozlab environment.

6.2.1.1 Consistency

The MongoDB database is built to operate well with large amounts of data [3], which leads to a consistency of well-functioning cooperation between the database and the software system when the data-storage expands.

6.2.1.2 Open-Source

MongoDB is an open-source database that can be used for free in the implementation and distribution of systems, which facilitates for new software systems that want to adopt a NoSQL database rather than using the normally used SQL database.

6.2.1.3 The update performance in MongoDB by id

In **Figure 13** taken from the research experiment in [3] displays the result where the MongoDB has a superior update performance based on the items Id in comparison to the MSSQL.

6.2.1.4 The Select/Read performance in MongoDB

In **Figure 12** and *Table 2* taken from the research experiment in [3], shows the result where the MongoDB has a superior selecting performance in comparison with its counterpart MSSQL.

6.2.1.5 The delete performance in MongoDB

Table 3 which represents the results from the research experiment in [27], shows where the deleting performance on MongoDB performs better than its counterpart MSSQL.

6.2.2 The disadvantages with MongoDB

I have presented the advantages with using the MongoDB in the previous subsections. In this section, I will discuss the disadvantages with using the MongoDB in relation with the Ozlab environment.

6.2.2.1 Not Microsoft owned

Microsoft Visual Studio is created and owned by Microsoft, which MongoDB is not, in a comparison to the MSSQL database. That leads to a separate maintenance where the developers for the Ozlab project will have to use two or more platforms to implement, read and manage the information generated in the Ozlab environment.

6.2.2.2 Non-relationship structure

Because of the non-relationship structure, statements such as Full-join, Union and Inner-Join, where you can connect two tables in the database to get a more satisfactory and precise match of a query statement. Do not operate in the same way with the MongoDB system.

6.2.2.3 The update performance in MongoDB

In **Figure 14** taken from the research experiment in [3], shows the result where the MongoDB has an inferior update performance to the MSSQL database when the data-storage in the database grows larger. Where the explanation in [3] is that the non-structure nature of MongoDB forces unindexed queries to perform complex lookups on each data item, when making an update performance that is not based on the items Id.

6.3 The database for Ozlab environment

After a fair comparative analysis between the MongoDB and MSSQL databases, I have concluded that the MSSQL is the most suitable database for the Ozlab environment. This decision is based on the following reasons:

1. **Data requirements and volume**

   The Ozlab environment is designed to work with well-structured data in relational databases. It does not need to handle large volumes of unstructured data, which would make MongoDB more beneficial.

2. **Microsoft and Visual Studio**

   As mention earlier, because the MSSQL is developed by Microsoft, I believe that would facilitate for further development of Ozlab environment, since the Ozlab system is being developed in the Microsoft programming platform Visual Studio.

3. **Shorter setup**

   The second reason is the shorter setup between the MSSQL, Ozlab and Microsoft Visual Studio, that can be of benefit if the current database-connection between Ozlab system and MSSQL would disintegrate (MSSQL →| Ozlab ← Microsoft Visual Studio).

4. **SQL language**

   Although the research experiments in [3] and [27] shows where the MongoDB outperforms the MSSQL in terms of select and reading data. The SQL language offers statements such as Full-join, Union and Inner-Join, where you can connect two tables in the database to get a more satisfactory and precise match of a query statement.

5. **SQL's popularity and ongoing improvements**

   Since the first release of MSSQL (1989), the software systems popularity has increased over the years and established a dominant place in the market place. Where the SQL language and MongoDB database are used in large organizations for the example of Google and Facebook [28] [29].

Another reason to keep using the MSSQL are the continuous investment and improvements that are being made to the SQL standard, one, for example, is that SQL now supports JSON as the JSON file format becomes a more popular data interchange format. [28]

7 Conclusion

In this chapter, I discuss the overall conclusion of this comparative analysis for the Ozlab environment, which includes: (1) the overall conclusion and content of this comparative analysis, (2) the challenges and unforeseen events that I faced in this project, (3) a few important remarks about the project in general.

This thesis has presented the databases MSSQL and MongoDB along with the Ozlab environment. I have presented the structure of the MSSQL and MongoDB based on my own implemented database content, through figures and textural explanations. I also demonstrated how the two databases connects with the Microsoft Visual Studio. The CRUD operations for MSSQL and MongoDB, that are implemented with example methods used on my own created MSSQL and MongoDB database, alongside with test results from existing scientific CRUD experiments. And lastly, in the essay explain the advantages and disadvantages of using MSSQL and MongoDB for the Ozlab environment. The final section of the essay concludes with me electing the database that I believe the Ozlab environment should use.

7.1 Facing and overcoming challenges

The challenges from this project

There have been numerous challenges in this project where I for example have been I forced to find alternative ways of analyzing the two databases for the Ozlab environment, from what was originally planned.

The lessons learnt from this project

This project allowed me to learn how to find alternative ways of handling situation when projects cannot be executed as planned. I was also able to exploit the opportunity of learning valuable knowledge about the differences between a non-structure and a structural database. This project has provided me the knowledge of how to operate within these two databases as a user and how the MSSQL and MongoDB can be integrated with the Microsoft Visual Studio.

7.2 Remarks

I am grateful for the opportunity to write this comparative thesis about MSSQL and MongoDB databases for the Ozlab environment and I‘am overall satisfied with this project and hoping to continuously develop my competence and knowledge of managing databases for software systems.

References

[1] MongoDB (u.å). *What is MongoDB?*. Available at: https://www.mongodb.com/compare/mongodb-mysql?jmp=docs [Accessed 20 Jan. 2018].

[2] Rouse, Margaret (2017). Microsoft SQL Server. Available at:http://searchsqlserver.techtarget.com/definition/SQL-Server [Accessed 20 Jan. 2018].

[3] Parker, Zachary. Poe, Scott. Vrbsky, V Susan (2013). *Comparing noSQL MongoDB to an SQL DB.* Proceedings of the 51st ACM Southeast Conference. DOI: 10.1145/2498328.2500047

[4] Wu, Ming Chieh. Huang, Fu Yin. Lee John (2015). *Comparison Between MongoDB and MS-SQL Databases on the TWC Website.* American Journal of Software Engineering and Applications. DOI: 10.11648/j.ajsea.20150402.12

[5] Pettersson Sören, John. Wik Malin. (2014). *Perspectives on Ozlab in the cloud.2. Karlstad: Universitetstryckeriet. ISBN 978-91-7063-587-8.*

[6] Åhs, Malin. (2017). *The Ozlab system – Underlying principles and a system overview.* . Available at: https://www.kau.se/en/ozlab/research-and-development/ozlab-system-underlying-principles-and-system-overview [Accessed 20 Jan. 2018].

[7] Rouse, Margarete (2008). *CRUD cycle (Create, Read, Update and Delete Cycle).* Available at: http://searchdatamanagement.techtarget.com/definition/CRUD-cycle [Accessed 20 Jan. 2018].

[8] MongoDB (2017). *MongoDB CRUD Operations*. Available at: https://docs.mongodb.com/manual/crud/ [Accessed 20 Jan. 2018].

[9] Medic, Milica (2014). *Creating and using CRUD stored procedures.* Available at: https://www.sqlshack.com/creating-using-crud-stored-procedures/ [Accessed 20 Jan. 2018].

[10] Tutorialspoint (u.å). *PL/SQL – Records*. Available at: https://www.tutorialspoint.com/plsql/plsql_records.htm [Accessed 20 Jan. 2018].

[11] MongoDB (u.å). *MongoDB Download Center*. Available at: https://www.mongodb.com/download-center?jmp=nav#community [Accessed 20 Jan. 2018].

[12] Wikipedia (2017). *Columns(database)*. Available at: https://en.wikipedia.org/wiki/Column_(database) [Accessed 20 Jan. 2018].

[13] Wikipedia (2017). *SQL Server Express*. Available at: https://en.wikipedia.org/wiki/SQL_Server_Express [Accessed 20 Jan. 2018].

[14]Microsoft (2017). *Download SQL Server Management Studio(SSMS)*. Available at: https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms [Accessed 20 Jan. 2018].

[15] Microsoft. (u.å). *ASP.NET*. Available at: https://www.asp.net/ [Accessed 20 Jan. 2018].

[16] Wikipedia (2017). *Internet Information Services*. Available at: https://sv.wikipedia.org/wiki/Internet_Information_Services [Accessed 20 Jan. 2018].

[17] Microsoft (2010). *IIS 7.NET Extensibility component is required*. Available at: https://technet.microsoft.com/en-us/library/dd421841(v=exchg.80).aspx [Accessed 20 Jan. 2018].

[18] Microsoft (u.å). *ISAPI Extension Overview*. Available at: https://msdn.microsoft.com/en-us/library/ms525172(v=vs.90).aspx [Accessed 20 Jan. 2018].

[19] Microsoft (u.å). *ISAP Filter Overview*. Available at: https://msdn.microsoft.com/en-us/library/ms524610(v=vs.90).aspx [Accessed 20 Jan. 2018].

[20] Wikipedia. (2017). *Websocket*. Available at: https://sv.wikipedia.org/wiki/Websocket [Accessed 20 Jan. 2018].

[21] Robomongo (u.å). *Robomongo is Robo 3T*. Available at: https://robomongo.org/ [Accessed 20 Jan. 2018].

[22] TechTerms (u.å). *Data*. Available at: https://techterms.com/definition/data [Accessed 20 Jan. 2018].

[23] Rouse, Margaret (2017). *database(DB).* Available at: http://searchsqlserver.techtarget.com/definition/database [Accessed 20 Jan. 2018].

[24] Essential SQL (u.å). *What is a Database Table?* Available at: https://www.essentialsql.com/what-is-a-database-table/ [Accessed 20 Jan. 2018].

25] ApexSQL Solution Center (2017). [Online. *How to create and use CRUD stored procedures in SQL Server.* Available at: https://solutioncenter.apexsql.com/how-to-create-and-use-crud-stored-procedures-in-sql-server/ [Accessed 20 Jan. 2018].

[26] Boicea, Alexandru. Radulescu, Florin. Agapin, loana Laura (2012). *MongoDB vs Oracle – Database Comparison.* 3rd International Conference on Emerging Intelligent Data and Web Technologies. DOI: 10.1109/EIDWT.2012.32

[27] Li Yishan, Manoharan, Sathiamoorthy (2013). *A performance comparison of SQL and NoSQL databases.* Conference: Communications, Computers and Signal Processing(PACRIM), 2013 IEEE Pacific Rim Conference. DOI: 10.1109/PACRIM.2013.6625441

[28] Dix John. (2014). *What's better for your big data application, SQL or NoSQL?* Available at: https://www.networkworld.com/article/2226514/tech-debates/what-s-better-for-your-big-data-application--sql-or-nosql-.html [Accessed 20 Jan. 2018].

[29] MongoDB (u.å). *Flexible enough to fit any industry*. Available at: https://www.mongodb.com/who-uses-mongodb [Accessed 20 Jan. 2018].