5-2-2022

Performance Comparison Between Relational and Non-Relational Databases in TCMS

Ahmad Yousef Imam

Performance Comparison Between Relational and Non-Relational Databases in TCMS

by

Ahmad Yousef Imam

An Honors Capstone

submitted in partial fulfillment of the requirements

for the Honors Diploma

to

The Honors College

of

The University of Alabama in Huntsville

5/2/2022

Honors Capstone Director: Dr. Dan Schrimpsher

DocuSigned by:
Ahmad Imam
D1D057E06D42445...

4/28/2022

Student (signature) Date

4/29/2022

Director (signature) Date

Dr. Letha Etzkorn Digitally signed by Dr. Letha Etzkorn
Date: 2022.05.02 08:52:02 -05'00'

Department Chair (signature) Date

William Wilkerson Digitally signed by William Wilkerson
Date: 2022.05.03 09:41:12 -05'00'

Honors College Dean (signature) Date

HONORS COLLEGE
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Honors College
Frank Franz Hall
+1 (256) 824-6450 (voice)
+1 (256) 824-7339 (fax)
honors@uah.edu

Honors Thesis Copyright Permission

This form must be signed by the student and submitted as a bound part of the thesis.

In presenting this thesis in partial fulfillment of the requirements for Honors Diploma or Certificate from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department, Director of the Program, or the Dean of the Honors College. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

Ahmad Imam

Student Name (printed)

DocuSigned by:
Ahmad Imam
D1D057E06D42445...

Student Signature

4/28/2022

Date

Table of Contents

Abstract

This project encapsulates and extends the existing work in CS 499, the Computer Science Capstone Course that entails the analysis and solving of a complex software project. The original project, the Trucking Company Management System, describes a request to design a system that simulates the storage, presentation, and maintenance of information relevant to a trucking company. This includes different branches of information, such as Shipping and Maintenance, that should not only be maintained by the system, but also only offered to authorized users. Such a type of system, where a large set of data must be maintained for a company, is commonly stored within a database. However, two major types of databases exist: relational and non-relational. Relational databases, including the vast array of SQL database languages, provide strict, tabular restrictions to the way data is stored and maintained, with links between tables. Meanwhile, non-relational databases, such as MongoDB, provide no such structure or links, promoting flexibility and speed. Since relational databases are more popular and have been in industry longer, it is more commonly chosen for software problems, and our project followed this trend, using MySQL, a relational database, as our database of choice. In order to ascertain the difference in speeds between queries (requested operations on the database) for each model, I developed a second database in MongoDB, a non-relational database, in order to promote a direct comparison between each model. Across the history of computing, speed has always been one of the principal concerns that leads to significant efforts in research and development. Being able to determine not only which model is faster, but by what exact degree in a specific instance of a database-related software problem, allows one to consider the costs and benefits of switching away from a more familiar model to a new design.

Introduction

The process of completing the project entailed first following the normal course of project flow for the course. Our project first required several foundations in the code base regardless of what type of database was used, including the flow of execution, user permissions, layout of the Graphical User Interface, and navigation of the interface. Thus, the beginning of development mostly focused on creating the skeletons for these aspects of the project. Once these were completed to the best of their ability without a database present, the first database was created. Since the group was focused on using MySQL, this database was created first, along with its backend model. Once this was created, tasks were then focused on populating the database with appropriate data and generating the necessary queries to fulfill the necessary requirements of the project. After the database was set up and queries were starting to develop, I then created my second database in MongoDB, populating it with the same data and setting it up in a similar manner to that of the existing MySQL database. I then created a second data model that correlated with this database. At this point, I also began to rewrite the MySQL queries in MongoDB, so that they each returned the same expected values.

For these comparisons, one has to make sure that the scenarios for each are equivalent, allowing for the test to only take into account the time necessary to access the database and perform the desired operation, controlling for confounding variables. This includes latency to connect to the database and the algorithm used in C# that accessed the database. Thus, for these comparisons, both databases were hosted on the localhost of the same machine, removing any latency endemic to an internet connection. The algorithms that were used in C# were also exactly

the same for each, ensuring that the only difference was the database that was being connected to.

Data

Logging in:

-MySql:

1. 1,595 milliseconds
2. 1,614 milliseconds
3. 1,555 milliseconds
4. 1,504 milliseconds

Average Time: 1,567 milliseconds

-MongoDB:

1. 652 milliseconds
2. 661 milliseconds
3. 593 milliseconds
4. 663 milliseconds

Average Time: 643 milliseconds

Average Difference: MongoDB is 925 milliseconds faster

Loading Monthly Maintenance Report:

-MySql:

1. 1172 milliseconds
2. 1181 milliseconds

3. 1509 milliseconds

4. 1219 milliseconds

Average Time: 1270 milliseconds

-MongoDB:

1. 17 milliseconds

2. 32 milliseconds

3. 17 milliseconds

4. 17 milliseconds

Average Time: 21 milliseconds

Average Difference: MongoDB is 1249 milliseconds faster

Loading Full Maintenance Report:

-MySql:

1. 1345 milliseconds

2. 1324 milliseconds

3. 1384 milliseconds

4. 1317 milliseconds

Average Time: 1343 milliseconds

-MongoDB:

1. 22 milliseconds

2. 35 milliseconds

3. 61 milliseconds

4. 52 milliseconds

Average Time: 43 milliseconds

Average Difference: MongoDB is 1300 milliseconds faster

Loading Personnel Page:

-MySql:

1. 1608 milliseconds

2. 1602 milliseconds

3. 1321 milliseconds

4. 1324 milliseconds

Average Time: 1464 milliseconds

-MongoDB:

1. 1 millisecond

2. 1 millisecond

3. 1 millisecond

4. 2 milliseconds

Average Time: 1 millisecond

Average Difference: MongoDB is 1463 milliseconds faster

Inserting New Personnel:

-MySql:

1. 270 milliseconds

2. 285 milliseconds

3. 279 milliseconds

4. 266 milliseconds

Average Time: 275 milliseconds

-MongoDB:

1. 100 milliseconds

2. 90 milliseconds

3. 86 milliseconds

4. 87 milliseconds

Average Time: 91 milliseconds

Average Difference: MongoDB is 184 milliseconds faster

Methods and Results

The method used to measure the amount of time taken accessing and returning values for each database involved the Stopwatch class in the Systems.Diagnostics library endemic to C#. This class allows one to surround calls to the database in start and stop commands by a stopwatch object, then returning the elapsed amount of time in milliseconds. This method was used over several different types of database calls and used in trials of 4, where the average time over those trials was used for comparison. The first comparison involved loading the login page, which requires calling the database with username and password parameters, checking the parameters against the database, and returning the row that contains those parameters. In MySql,

the average time for this operation was 1,567 milliseconds (1.6 seconds), while it only took an average 643 milliseconds in MongoDB; thus, nearly identical operations were about 925 milliseconds faster in MongoDB than in MySql. The next comparison involved loading the monthly maintenance report page. This involves grabbing an entire list of maintenance records from the database, then returning filtered records based on only those that occurred during the current month. In MySql, the average time for this operation was 1,270 milliseconds, while in MongoDB, the average time was a staggering 21 milliseconds. Thus, MongoDB was an entire 1,249 milliseconds (1.2 seconds) faster than MySql, where the speed difference was obvious to the user accessing the GUI. The next comparison was similar, based on loading the full maintenance report page. This simply returned the entire list of maintenance records rather than filtering for those of the current month. In MySql, the average time for this operation was 1,343 milliseconds, while for MongoDB, the average time was 43 milliseconds. Thus, MongoDB was able to load the full maintenance page an entire 1,300 milliseconds (1.3) seconds faster than MySql. The final comparison involved a write operation involving the Adding Personnel page. In this comparison, each database was meant to add a new user to its records that could then be accessed in the database similar to any other existing record. In MySql, the average time for this operation was 275 milliseconds, while for MongoDB, the average time was 91 milliseconds. Thus, MongoDB was able to insert a new personnel record 184 milliseconds faster than MySql.

Thus, across all operations, MongoDB was an average 1,024 milliseconds (1 second) faster than MySql. While 1 second may seem like a short amount of time, this can make a noticeable difference for users, especially when comparing the software side by side. When using the GUI, there is a noticeable lag when clicking on buttons associated with a MySql query, while it appeared instantaneous for a MongoDB user. It should also be noted that this example involved

using a relatively small database with minimal records, yet in an industrial setting, one would expect a much greater load of data, as well as number of connections trying to access it. Furthermore, industrial databases will typically require their databases to be hosted on a web service, which comes with its own latency to connect to it across the internet. Thus, utilizing a Non-Relational database such as MongoDB would help to alleviate further speed issues associated with such infrastructures.