CS Capstone Technology Review

November 28, 2017

Investment Performance Mobile App

Prepared for

HedgeServ

Edison Tsai ______________________ Signature ________ Date

Prepared by

Group 68

The Cleverly Named Team

Tyler Jones ______________________ Signature ________ Date

Abstract

For our upcoming project, which is an Investment Performance Mobile Application, there are many different pieces of technology that should be considered before development begins. The following paper breaks down 3 main portions of the back end database for our application, and discusses the general idea of relational vs. non-relational databases, compares different database management systems, and compares different hosting options for the database.

Contents

1 Storage and Handling of Data

1.1 Overview

The storage and handling of data, otherwise known as the back end, is of the utmost importance in most mobile applications, and this project is no exception. Quality data maintenance and storage is critical to providing the user with investment performance data for their portfolio which is the core functionality of this application.

1.2 Criteria

The back end for this application must be able to store a wide array of information related to the user's investment portfolio. One of the most important criterion will be to encrypt the user's personal information, such as username and password, and total funds invested. Furthermore, when considering data storage, the industry standard metrics, ACID, are analyzed. ACID stands for Atomicity, Consistency, Isolation, and Durability. Atomicity refers to the need for a database transaction to either completely pass, or completely fail. In the event of a transaction failing, there is no partial data transfer, and the changes are rolled back to the previous state. Consistency considers that any transaction must be valid. This validity is determined in the context of all constraints, cascades, and triggers. Isolation considers the need for concurrent execution of transactions to always lead to a state that could have been reached had said transactions been completed sequentially. Lastly, durability considers the need that once a transaction has been fully completed, the data must be stored permanently, especially in the event of crashes, errors, and even power loss. [1]

1.3 Potential Choices

When considering back end options and data storage, two main options are considered - relational and non-relational databases. There are different types/implementations of relational databases, as well as non-relational databases, but only the groups will be discussed for this piece.

1.3.1 Relational Database

Relational databases, at their core, have a rigid, mathematical approach to their structure. Data is stored in data buckets (tables), and these buckets are referenced by a key or index. Other buckets can then reference said key to establish a link between the data. These buckets are divided into columns, with each column having its own datatype. Moreover, a standardized language, called SQL, is used to store and retrieve data from relational databases.

To consider relational databases in the context of our project, one table may contain login information for a user, another may contain all the holdings that a user has, another may contain data about the investments themselves such as ticker symbol, pricing, etc. The data in our relational database would be more rigid in structure than that of a non-relational database, and the data from these 3 tables could be manipulated and displayed as needed using SQL.

1.3.2 Non-Relational Database

Non-relational databases differ fundamentally from relational databases. As opposed to keys and indexes all spread among various tables, with SQL joining tables and the data within them, non-relational databases function with a variety of different strategies for storing data. These include, key-value cache, key-value store, key-value store(eventually consistent), key-value store (ordered), data structures server, tuple store, object database, and wide column store.[2] Overall, the approach and need for a non-relational database depends on the requirements of the developer, and allows for more flexibility in the way they can be designed. They are less rigid in design than relational databases, and this comes with an array of advantages and disadvantages depending on the specific choice made of non-relational database.

One inherent advantage to the less rigid structure boasted by non-relational databases is the ability to scale well. Non-relational databases typically exist in clusters of nodes. This means that they can scale out horizontally, because multiple servers work together, each sharing part of the load.

1.3.3 Brief Note

For the data management portion of this application, there are only two major families of data storage to consider, and for this reason, a third potential choice is out of the scope of this project. As stated above, non-relational databases, and their respective implementations, differ fundamentally from relational databases, and their respective implementations. To clarify, the scope of this consideration and weighing of options is not considering the differences between, for instance MongoDB and Cassandra, which are two different non-relational implementations, but rather the larger group they belong to. Currently, other solutions in addition to non-relational and relational databases are being developed, but the scope of this project does not include them for consideration.

1.4 Discussion

The main criteria for data storage, as stated earlier, can be abbreviated as ACID. Since the data that this application will be storing is of high importance, as it contains financial data for the user, security and durability are the number one consideration. Relational databases, by the nature of their management systems, fulfill each component of ACID, while non-relational databases don't necessarily. For the scope of this application, non-relational databases would not serve our needs any better. The application will not require the advantage in scaling that NoSQL provides, as our database will likely only need to support a number of users on the order of thousands, which would see no significant increase in efficiency if a non-relational database were to be used. Utilizing a relational database has a wide array of advantages for this project, including the ability to use SQL, as well as fulfilling the requirements set by ACID.

1.5 Conclusion

For this application, we have elected to use a relational database for our back end data. Due to the inherent advantages as far as rigidity of design, as well as being able to guarantee ACID, relational databases will not only fulfill our needs for this project, but be much easier to design and interact with in the process.

2 Database Management System

2.1 Overview

In considering the implementation of the back end relational database for our application, it is necessary to consider which database management system will be used. A database management system (DBMS) is essentially nothing more than a computerized data-keeping system [3]. The users of the DBMS have an array of tools at their disposal in order to achieve various operations for either manipulation of the data that is contained within the database, or for changing the actual structure of the database. These structures could include the changing of tables, indexes, and keys. [3]

2.2 Criteria

When considering the necessary criteria for our selection of DBMS, any decision that is made of selecting between a list of options comes down to preference, ease of use, and availability. Almost all DBMS's in existence share many more similarities than differences, and considering this fact, any harsh criteria for our selection of DBMS is nonexistent, and rather will come down to preference.

2.3 Potential Choices

For this piece, I will be looking at 3 very common DBMS options within the industry. These options are Microsoft SQL Server, MySQL, and PostgreSQL.

2.3.1 Microsoft SQL Server

Regarding Microsoft SQL Server (MSS), it is perhaps most important to note that MSS is an enterprise system, rather than open source. For this reason, companies will often choose to use MSS over something open source like MySQL or PostgreSQL due to its high quality and proprietary, uniquely supported features. Moreover, Microsoft has a variety of editions of SQL Server available to choose from in order to accommodate the client's specific needs and budget. SQL server was originally developed by Microsoft for exclusive use with Windows OS. MSS supports a variety of programming languages, including Java, PHP, C++, Python, Ruby, Visual Basic, Delphi, GO and R. [4] SQL server, due to its enterprise level of quality, supports the option to stop query execution. SQL server, although it is a relational DBMS, which by nature fulfill the ACID requirement, can handle non-atomic processes. This means that a query to the database can be cancelled while it is running, a feature that is a considerable quality of life improvement over its competitors. Additionally, SQL server doesn't allow any process to access or manipulate its database files or binaries, therefore SQL server is also more secure than the other options in consideration.[4]

2.3.2 MySQL

MySQL, to contrast MSS, is open source and is thus often chosen simply for its financial benefits. MySQL, because of its open source nature, is also more portable in the sense that it was not originally developed solely for Windows, and is able to run smoothly on Linux, Mac OS, and FreeBSD, making it the optimal choice for a project such as ours whose users are using an array of operating systems. Additionally, MySQL supports all the languages that are supported by MSS, in addition to Perl, Scheme, Tcl, Haskel, and Eiffel, which gives us more choices with how we interact with our stored data.[4] Furthermore, MySQL allows users to filter out tables, rows, and users in a variety of ways. This is in contrast with MSS which supports row-based filtering. Lastly, as stated previously, MySQL suffers where MSS shines in its inability to cancel running queries without killing the entire process, and is also less secure due to its ability to have its files changed through binaries while running.[4]

2.3.3 PostgreSQL

PostgreSQL PostgreSQL is very similar to MySQL, however there are a handful of differences that are worth considering. Firstly, PostgreSQL (PGSQL) is open source, similar to MySQL and in contrast to MSS. One place that MySQL and PostgreSQL differ significantly is in respect to their partitioning methods, which is how data is stored in the database onto different nodes. MySQL uses a proprietary technology, called MySQL Cluster, to perform horizontal clustering, which consists of creating multiple clusters with a single cluster instance within each node, while PostgreSQL doesn't implement true portioning.[5]

2.4 Discussion

When considering which DBMS to use, specifically relating to a relational database, many of the available features are similar enough that choosing one DBMS over another is not likely to have any large implications for the scope of our project. The consideration of scaling, uptime, and other note-worthy performance metrics is unnecessary for this project. In an enterprise setting, these metrics matter much more than for something relatively small, and an application that is more so a proof of concept than anything else. For this reason, the choice of DBMS that we use will likely come down to financial reasons. As students, we are aiming to be frugal, and thus software that is open source is always appealing and worth a second glance.

2.5 Conclusion

For our application, we will be using MySQL for our DBMS. With its wide array of supported languages, portability across many different operating systems, access to filtering of our data, and open source nature, it is the ideal choice over PostgreSQL, and MSS.

3 Database Hosting

3.1 Overview

In considering how to implement the database for our application, it is necessary to finally choose a host for our MySQL database. Different hosts offer different levels of support in terms of performance, scalability, security, pricing and more.

3.2 Criteria

In choosing a host for our database, we must consider and reference our requirements document for our application. Our client has stated that our database will likely need to support a number of users on the scale of thousands, which is relatively small, so therefore scalability isn't as important for this application as it may be in an enterprise setting. Moreover, security is also extremely important as our users will be storing their financial data within our database. Performance is the last significant aspect of our host selection that we must consider, as we need timely returns on our queries to create a desirable experience for the users of our application.

3.3 Potential Choices

For this piece, I will be considering 3 different MySQL hosts. I will weigh the pros and cons between Google Cloud SQL, Microsoft Azure, and Oregon State hosting.

3.3.1 Google Cloud SQL

Google Cloud SQL Google Cloud SQL has the advantage of being on the cloud, which comes with a wide array of benefits. Cloud based database hosting offers better scalability, location independence, and lighter administrative burden [6]. Data is stored in the cloud with Google Cloud SQL, and thus it reaps these benefits. Moreover, Google Cloud SQL offers high performance, and provides the peace of mind of being fully managed by Google. Google Cloud SQL boasts 99.95% availability, up to 10TB of storage capacity, 25,000 input/output operations per second, and 208GB of RAM per instance. Google pricing ranges from $0.0150 - $4.0240 per hour from 600MB to 208GB. [7]

3.3.2 Microsoft Azure

Microsoft Azure, similar to Google Cloud, offers cloud hosting. Cloud hosting with Microsoft comes with all the same benefits that come with Google Cloud. Furthermore, Microsoft Azure offers built-in high availability with no additional cost, predictable performance, scaling on the fly within seconds, automatic backups, and enterprise-grade security and compliance that is backed by Microsoft. Microsoft Azure pricing is currently set at $0.021/hour for 50 compute units.[8]

3.3.3 Oregon State Hosting

As Oregon State students, every student is provided with one MySQL database. This database is hosted by the school, and has the great advantage of being free to use, as well as being locally hosted. Oregon State hosting is not cloud based, and thus does not boast many of the impressive gains in scalability and uptime that are provided by Microsoft Azure and Google Cloud SQL.

3.4 Discussion

When selecting a host for our database, the things we must consider as valid concerns and metrics differ significantly from considerations in an enterprise setting. Scalability and uptime/availability are primary concerns for a company whose database directly affects their bottom line, and thus cloud hosting is much more appealing. What is valuable for the scope of our application, however, is pricing, as well as response time, while top-tier scalability is mostly negligible, as well as many of the other listed benefits that would come with outside cloud hosting.

3.5 Conclusion

For the reasons listed, we will be using Oregon State hosting for the development of our application due in large to the appeal of the local hosting and lack of cost. Microsoft Azure and Google Cloud SQL would also both be suitable for this app, but the features that we would be paying for would not be effectively utilized enough to justify including them as an expense.

References

[1] ACID, Wikipedia, 26-Nov-2017. [Online]. Available: https://en.wikipedia.org/wiki/ACID. [Accessed: 28-Nov-2017].

[2] NoSQL, Wikipedia, 16-Nov-2017. [Online]. Available: https://en.wikipedia.org/wiki/NoSQL. [Accessed: 28-Nov-2017].

[3] BM Knowledge Center. [Online]. Available: https://www.ibm.com/support/knowledgecenter/zosbasics/com.ibm.zos.zmiddbmg/zmiddle_46.htm. [Accessed: 28-Nov-2017].

[4] . Solutions, A Comparison between MySQL vs. MS SQL Server Mindfire Solutions Medium, Medium, 10-Apr-2017. [Online]. Available: https://medium.com/@mindfiresolutions.usa/a-comparison-between-mysql-vs-ms-sql-server-58b537e474be. [Accessed: 28-Nov-2017].

[5] Key differences between MySQL vs PostgreSQL, GoDaddy. [Online]. Available: https://in.godaddy.com/help/key-differences-between-mysql-vs-postgresql-12392. [Accessed: 28-Nov-2017].

[6] . by R. Recchia, What Are The Advantages And Disadvantages Of Having Your Database In The Cloud?, Incendia, 26-Jan-2017. [Online]. Available: http://www.incendia.com/what-are-the-advantages-and-disadvantages-of-having-your-database-in-the-cloud/. [Accessed: 28-Nov-2017].

[7] Cloud SQL - MySQL PostgreSQL Relational Database Service — Google Cloud Platform, Google. [Online]. Available: https://cloud.google.com/sql/. [Accessed: 28-Nov-2017].

[8] Azure Database for MySQL pricing Preview, Pricing - Azure Database for MySQL — Microsoft Azure. [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/mysql/. [Accessed: 28-Nov-2017].

[9] System Properties Comparison Microsoft SQL Server vs. MySQL vs. PostgreSQL, Microsoft SQL Server vs. MySQL vs. PostgreSQL Comparison. [Online]. Available: https://db-engines.com/en/system/Microsoft SQL Server%3BMySQL%3BPostgreSQL. [Accessed: 28-Nov-2017].