Hotel Booking Platform

Cloud-based Data Portability Across Multiple Providers

# Table of Contents

[Table of Contents 2](#_Toc134891991)

[1. Introduction 3](#_Toc134891992)

[2. Literature Review 4](#_Toc134891993)

[3. System Overview 5](#_Toc134891994)

[4. System Model 6](#_Toc134891995)

[5. Implementation 6](#_Toc134891996)

[6. Results 6](#_Toc134891997)

[7. References 7](#_Toc134891998)

# Introduction

The hotel booking platform project aims to provide a solution for companies who face challenges when using different cloud providers that have varying database architectures. In today's world, has become an essential part of the business world. However, many businesses face the issue of being locked into a specific cloud platform, which makes data portability between different cloud providers a significant challenge. This project focuses on addressing this problem by providing a solution that allows data portability between multiple cloud providers, reducing the risk of being locked into a specific cloud platform.

The project utilizes a Python GUI to provide an easy-to-use interface for hotel bookings. The Java server is designed to connect to multiple cloud databases, depending on the requests received from the Python client. This provides businesses with the flexibility to choose between different cloud providers and architectures based on their specific requirements. The server also includes a feature that allows for the migration of data from one database to another.

By using multiple cloud providers simultaneously, businesses can benefit from the unique strengths of each provider while avoiding vendor lock-in. This project's architecture allows for seamless switching between providers while maintaining data consistency and minimizing downtime.

The report will discuss the design and implementation of the project, including the challenges faced during development and the solutions that were implemented. It will also discuss the benefits of the project, including data portability, fault tolerance, and scalability, as well as the data migration feature. Finally, the report will provide recommendations for future improvements to the project, including additional features and optimizations.

# Literature Review

The emergence of cloud computing has led to a vast array of cloud databases that offer unique features and benefits. However, each cloud database has its own advantages and disadvantages, making it challenging to choose the most appropriate database for a particular application. Moreover, being locked into a specific cloud platform can be a common problem when developing software for the cloud, making it difficult to switch providers or use multiple clouds simultaneously.

SQL databases have been around for several decades and are well known for their strong consistency and reliability. However, they may not be suitable for applications that require horizontal scaling due to their rigid schema design. On the other hand, NoSQL databases are designed for horizontal scaling and can handle large amounts of unstructured data. MongoDB is a popular NoSQL database that offers high performance for basic operations such as insertion and retrieval. However, it does not support aggregate functions natively, which can result in performance issues for applications that require such functions.

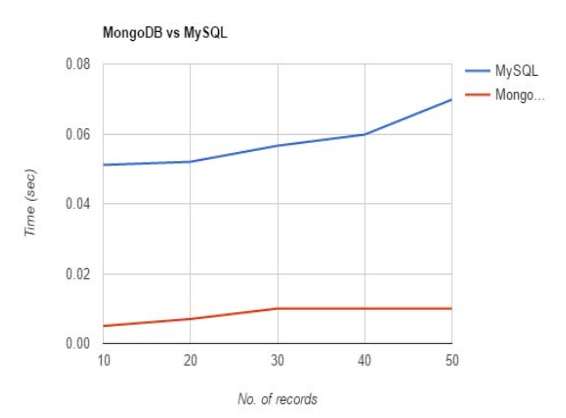
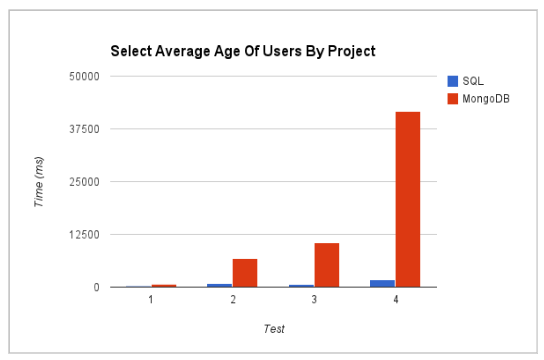


Figure 1: MySQL and MongoDB performance for two different cases [1][2]

The conflicting results from the plots in Figure 1, highlight the importance of considering the specific use case and workload when selecting a cloud database. As shown in the right plot, MongoDB outperforms MySQL. However, as shown in the left plot, MySQL shows better performance for the aggregate function. This is because each database has its own strengths and weaknesses and may be optimized for different types of data and workloads. Therefore, a thorough analysis of the specific requirements of the project and the available cloud database options is necessary to make an informed decision. Additionally, the use of multiple cloud databases can be a viable solution to leverage the strengths of each database and mitigate their weaknesses.

To overcome the limitations of using a single cloud provider, many organizations are moving towards multi-cloud architectures. A multi-cloud architecture involves using multiple cloud providers to distribute the workload and minimize the risk of vendor lock-in. By leveraging the strengths of each cloud provider, a multi-cloud architecture can improve performance, scalability, and availability.

Data portability is a critical aspect of a multi-cloud architecture. Moving data between cloud providers can be challenging due to differences in data formats and storage architectures. However, some cloud databases offer built-in features for data migration. For example, MongoDB provides the mongodump and mongorestore utilities that allow users to export and import data from one database to another.

In conclusion, choosing the right cloud database is a critical decision that can have a significant impact on the performance and scalability of an application. By utilizing multiple cloud databases, organizations can leverage the strengths of each provider and improve overall performance. Furthermore, a multi-cloud architecture can provide additional benefits such as increased availability and reduced vendor lock-in. However, data portability remains a challenge, and organizations must carefully consider the trade-offs when designing a multi-cloud architecture.

# System Model

A screenshot of a computer

Description automatically generated with medium confidence

# System Model

# Implementation

# Results

# References

|  |  |
| --- | --- |
| [1] | Parker, Zachary, et al. “Comparing NoSQL Mongodb to an SQL DB.” Proceedings of the 51st ACM Southeast Conference, 2013, https://doi.org/10.1145/2498328.2500047. |
| [2] | M. M. Patil, A. Hanni, C. H. Tejeshwar, and P. Patil, “A qualitative analysis of the performance of MongoDB vs MySQL database based on insertion and retriewal operations using a web/android application to explore load balancing — Sharding in MongoDB and its advantages,” 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Feb. 2017, doi: [https://doi.org/10.1109/i](https://doi.org/10.1109/i-)smac.2017.8058365. |
| [3] | Mahdi Negahi Shirazi, Ho Chin Kuan, and Hossein Rezaei Dolatabadi, “Design Patterns to Enable Data Portability between Clouds’ Databases,” International Conference on Computational Science and Its Applications, Jun. 2012, doi: <https://doi.org/10.1109/iccsa.2012.29>. |
| [4] | F. Ravat, J. Song, O. Teste, and C. Trojahn, “Efficient querying of multidimensional RDF data with aggregates: Comparing NoSQL, RDF and relational data stores,” International Journal of Information Management, p. 102089, Jun. 2020, doi: <https://doi.org/10.1016/j.ijinfomgt.2020.102089>. |
| [5] | B. Jose and S. Abraham, “Exploring the merits of nosql: A study based on mongodb,” IEEE Xplore, Jul. 01, 2017. https://ieeexplore.ieee.org/document/8076778. |
| [6] | M. Diogo, B. Cabral, and J. Bernardino, “Consistency Models of NoSQL Databases,” Future Internet, vol. 11, no. 2, p. 43, Feb. 2019, doi: <https://doi.org/10.3390/fi11020043>. |
| [7] | R. Gunawan, A. Rahmatulloh, and I. Darmawan, “Performance Evaluation of Query Response Time in The Document Stored NoSQL Database,” IEEE Xplore, Jul. 01, 2019. https://ieeexplore.ieee.org/document/8898035. |
| [8] | S. Bradshaw, K. Chodorow, and Eoin Brazil, MongoDB : the definitive guide : powerful and scalable data storage. Sebastopol, Ca: O’reilly, 2019. |
| [9] | A. Davoudian, L. Chen, and M. Liu, “A Survey on NoSQL Stores,” ACM Computing Surveys, vol. 51, no. 2, pp. 1–43, Apr. 2018, doi: <https://doi.org/10.1145/3158661>. |
| [10] | M. Ha and Y. Shichkina, "The Query Translation from MySQL to MongoDB Taking into Account the Structure of the Database," 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), St. Petersburg, Moscow, Russia, 2021, pp. 383-386, doi: 10.1109/ElConRus51938.2021.9396591. |
| [11] | I. Mearaj, P. Maheshwari and M. J. Kaur, "Data Conversion from Traditional Relational Database to MongoDB using XAMPP and NoSQL," 2018 Fifth HCT Information Technology Trends (ITT), Dubai, United Arab Emirates, 2018, pp. 94-98, doi: 10.1109/CTIT.2018.8649513. |
| [12] | A. Kanade, A. Gopal and S. Kanade, "A study of normalization and embedding in MongoDB," 2014 IEEE International Advance Computing Conference (IACC), Gurgaon, India, 2014, pp. 416-421, doi: 10.1109/IAdCC.2014.6779360. |