# C.4. Database Comparison

## C.4.1. The Decision

We have selected Cassandra for MonashBnB, considering various aspects of both MongoDB and Cassandra. Cassandra is designed in a way that can help in handling high volumes of data and most importantly, provide high availability without a single point of failure. It is also highly scalable and consistent. The availability is due to Cassandra’s peer-to-peer approach which allows easy access to data as the read and write operations can be performed on all the nodes in the cluster.

## C.4.2. Merging Process

Merging process for importing JSON data in Cassandra:

There are multiple processes for importing JSON data in Cassandra or exporting MongoDB data to Cassandra. Two of these methods are:

1)     Importing JSON data by using the “JSON Support” provided by Cassandra. In this approach, we can either use the “INSERT JSON” or the “fromJson()” function.

2)     Exporting the MongoDB data through JSON or CSV export of that data.

The first process is a good one but, in our thought, the greatest limitation here is efficiency and processing speed due to executing ‘n’ number of queries. Therefore, we have selected the second method for the merging process.

Steps to merge the data:

1. Start the mongo shell on the terminal.
2. Write the mongo export command ("mongoDB - Documentation," 2019):

> ‘mongoexport --collection <collection\_name> --db <db\_name> --out <file\_name.csv>’

where:

·       --db is used to specify the database name

·       --collection is used to specify the collection name

·       --out  is used to specify the path of the output file (Batra, 2018)

1. Once the csv file is generated, create a KEYSPACE in Cassandra.
2. Create a table with all the fields similar to the fields present in csv file.
3. Copy the csv file data in the table, using the copy command:

> COPY <keyspace\_name.table\_name (field1, field2, field3)> FROM '<path\_name/file\_name.csv>' WITH HEADER = TRUE;

1. Check whether all the data has been stored correctly.

## 

*Fig. 1 Flowchart for Merging Process*

## C.4.3. Strengths and Weaknesses

|  |  |  |
| --- | --- | --- |
|  | **Strengths** | **Weaknesses** |
| **MongoDB** | * Supports secondary indexes ("ScaleGrid," 2016) * Is an expressive object model and is very ‘object-oriented’ * Supports auto-sharding * Good for unstructured data * Easy to use | * Is a ‘single-master’ model with just 1 primary server * Low write scalability (low write-throughput) * No ‘JOIN’ support, ‘$lookup’ function is used instead * High data consumption |
| **Cassandra** | * Is a ‘multiple-master’ model with no single point-of-failure * Is highly available with 100% uptime * High write scalability with the acceptance of writes on all the servers in the cluster (high write-throughput) (Al-Saeedi, 2016) * In comparison, high in performance * High query language support (CQL is similar to SQL) * Easy to use | * No ‘JOIN’ support, No ‘OR’ support * No ad-hoc query support * No subquery support * No ‘regex’ support * Cannot sort by a partition key column * A query needs to be run across all the declared partitions * Data modelling is tough |

## C.4.4. Comparison Report

MonashBnB is a residential service, offering the students and staff of Monash University a short time lodging experience. This field of service usually faces high amount of traffic in terms of users. Various new users and their data is going to be frequently added and removed in the MonashBnB’s database. The users here are the hosts, with their property listings, and the customers with their hunt of properties where they can find a stay. In such a case, continuous updating is required in the database. High traffic of users accessing a system and its database may lead to problems like congestion.

Acting towards the suggestion, MonashBnB’s database system was built and handled using MongoDB and Cassandra technologies. Both the technologies were involved in controlling the entire data of the company. The data was mainly divided into 3 parts, namely, hosts, listings and reviews. The hosts and listings parts were managed in MongoDB, whereas, the reviews were entirely worked on Cassandra. Specifically, managing the data included a new creation of the MonashBnB database and also updating the same. Further, few queries were resolved on the existing database provided by the company, and some new were designed as well.

As mentioned, we have decided to cruise further with Cassandra as the technology for MonashBnB’s database system. Working on both the technologies helped in identifying the various pros and cons of MongoDB and Cassandra. These steered us in choosing the appropriate technology for the given case study. Cassandra, as a technology is more robust when compared with MongoDB. Cassandra is known to handle huge amounts of data efficiently. Implied, the high number of users will be handled by Cassandra effectively. Also, Cassandra supports a peer-to-peer approach which makes the available data easily accessible, allowing the respective users perform the read and write operations effectively. The data managed on Cassandra is spread across all the assigned servers with equal operation rights. This makes the users experience a seamless performance while operating the MonashBnB system. The access to multiple servers avoids a total breakdown of the system, making it highly available for the end users. Additionally, CQL (Cassandra Query Language) has a high query language support. CQL syntax is quite similar to SQL (Structured Query Language), which is one of the most popular and commonly used query languages (Conrad, Misenar, & Feldman, 2016). This makes Cassandra a much easy-to-learn and easy-to-use query language. This feature can be used by the various database analysts and database administrators which might have a prior experience of working with SQL.

The data provided and generated by MonashBnB is leaned towards being more structured and has a clear understanding of the data with respect to their fields. There are a negligible number of null attributes, but high anomalies in terms of the data types of the information. The data storage can be classified into flexible and wide-column. Considering these factors, the decision of choosing Cassandra over MongoDB is justified even more.

To summarise, the information stored in the MonashBnB database system needs to be available 24 x 7 for the end users. The data should also be highly scalable, easily accessible and readily operable. The high number of users and huge amount of data should not affect the efficiency and effectiveness of the database system. To maintain this performance throughout, with least interruptions and difficulties for the audience, Cassandra is a clear winner of choice in comparison with MongoDB for this particular case study.