MySQL vs. MongoDB in an E-commerce website

Phase 1- Data Modelling and Querying

Helton Mendonça Nº 56870| Riccardo Frattarelli N°64492

Sebastião Cancela Nº 58282 | Mateusz Kleszcz Nº64327

Contributions

Each student contributed about the same amount to every part of the project.

Comparative analysis

Relative to the last phase, a few changes were made.
The first major difference is the volume of data used. The original dataset had 1500 observations at maximum, corresponding to the Order Details file. This is an extremely low amount of data, and we were seeing almost no significant differences between the execution of simple and complex queries, in both databases. To gain a more robust insight on our work, we decided to create randomly, roughly 25k lines for both the Order Details and Order List variables. Alongside that, we also wanted a greater scope of analysis trough time, since the original data only had examples from March 2018 to April 2019. So, we increased the number of months, starting in April 2014 until April 2019, equivalent to 60 months.
The final biggest difference, comes in the reformed structure of MongoDB. Previously, we created a collection for every original CSV file, meaning 3. But this time, after creating more data, we found that MongoDB was having trouble executing the complex queries in a reasonable time, so we decided to embed Orders List into Order Details, as they shared the most information. More on that in the next section.
It's to note that due to the reformed structure of the NoSQL database, we also had to slightly change all queries, since they all used information from Order Details collection.
We also considered slightly modifying the structure of the rational database, especially regarding the last table of Sales Target, due to the complex Primary Key. However, since no major differences in time were found after the generation of more data, we concluded we should not try to fix what's not broken and overcomplicate things.
Finally, we also implemented indexing on the variables we were using in the queries so we could access the differences in executing time.

Optimization

Regarding optimization, we believe the restructuring of MongoDB and Indexing, were the ones that contributed the most to a more optimized version of our project.
As said previously, MongoDB was having trouble with executing the complex queries. To put in perspective, the same hard queries in SQL were taking hundreds of milliseconds, while in NoSQL were taking tens of thousands of milliseconds. In other words, the difference was between almost instant execution and taking minutes to run. So, we looked into the queries, and we noticed the amount of $lookups and $unwids we were using in the complex queries. This is a way in MongoDB to “join” collections, but has a downside of being computationally expensive, especially when we are dealing with a lot of data. So we settled on embedding Orders List data onto Orders Details data, into one single collection named Orders. The difference was crystal clear, going from minutes to execute, to hundreds of milliseconds, just like in SQL. In fact, even without indexing, the complex queries in MongoDB were taking less time than MySQL, in average (we ran multiple times).
We also performed indexing similarly in both databases. They were done in variables used in the queries like Profit, Amount, City, State or Order Date. The difference was that in MySQL we also used an index on CategoryID, unlike MongoDB, where the index was implemented directly on the Category documents.

Discussion

This stage showcased significant enhancements in query performance for both MySQL and MongoDB. Expanding the dataset to 25,000 rows (up from 1,500) and broadening the time span to 60 months enabled more significant comparisons.
The primary optimization was reconfiguring MongoDB by integrating the Orders List within Order Details to form a cohesive Orders collection. This removed costly $lookup and $unwind processes, leading to a substantial decrease in execution times for intricate queries from minutes to hundreds of milliseconds, often surpassing MySQL on average. However, across various tests, simple queries consistently executed faster in MySQL.
Indexing was utilized for essential variables such as Profit, Amount, and Order Date in both databases, leading to slight decreases in query times overall. Although beneficial, these enhancements weren't as significant as the reorganization.
While we looked into expanding to 100,000 rows, the time required to create this data was too much. Nonetheless, the patterns seen with 25,000 rows validated that our optimization endeavors were effective, demonstrating significant enhancements in complex query performance for MongoDB.

Conclusion

This project offered a thorough comparison of MySQL and MongoDB with an E-commerce dataset, emphasizing the unique advantages and drawbacks of both database systems. During the initial phase, our main emphasis was on verifying that the queries functioned

properly and applying normalization in the SQL database. This method enabled us to carry out different queries efficiently, demonstrating the flexibility and organization of MySQL for handling relational data. In MongoDB, we opted for simplicity and utilized the same number of collections as the datasets we possessed.

During the second phase, we redirected our attention to data generation, optimization, and denormalization within the NoSQL database. By increasing the dataset to 25,000 rows and storing documents in MongoDB, we minimized dependence on resource-heavy processes such as $lookup. When coupled with indexing, these modifications greatly enhanced query execution speeds, with MongoDB surpassing MySQL in managing intricate queries. In the meantime, MySQL continued to hold its edge in basic queries, showcasing its effectiveness in simple relational tasks.

Even though expanding the dataset to 100,000 rows was impractical due to time limitations, the detected trends confirmed the effectiveness of our method. This project successfully connected theory with practice, offering valuable insights into database design and performance enhancement evaluation.