Report: Comparison of SQL and NoSQL Databases for Pokemon Database Design

Siri Sharvani A

May 9, 2024

1 Introduction

During the assignment to design a database for managing Pokemon data, we explored both SQL and NoSQL database solutions to evaluate their suitability for the task. This report outlines our experience, observations, and comparison between the two approaches.

2 Database Design

In the SQL approach, we designed tables for Pokemon, Types, Moves, and a junction table for the many-to-many relationship between Pokemon and Moves. Each entity was clearly defined with primary and foreign keys establishing relationships between tables.

Contrarily, the NoSQL approach involved creating collections for Pokemon and Moves. This schema was simpler and more flexible, allowing for variable structures within documents, which suited the semi-structured nature of Pokemon data.

3 Data Population

Populating the SQL database required multiple insert statements, especially for mapping relationships in the junction table. This process was straightforward but involved writing more SQL queries.

Conversely, populating the NoSQL collections was simpler as we could directly insert JSON documents representing Pokemon and Moves. This approach offered more flexibility and agility, especially when dealing with semi-structured data.

4 Querying

Writing queries in SQL involved using join operations to retrieve data from related tables. While SQL queries provided powerful capabilities for filtering and aggregating data, complex queries could be verbose and require a good understanding of SQL syntax.

In the NoSQL approach, queries were simpler and more intuitive, usually involving basic CRUD operations on documents within collections. However, complex queries requiring joins or aggregations were challenging and often required additional processing on the client-side.

5 Performance

SQL databases are optimized for handling structured data and complex queries efficiently. With proper indexing and query optimization, SQL databases can provide high-performance even with large datasets and complex queries.

NoSQL databases excel in horizontal scalability and can handle large volumes of data with ease. However, performance might degrade with complex queries or when dealing with deeply nested documents.

6 Conclusion

Both SQL and NoSQL databases offer unique advantages and trade-offs in the context of managing Pokemon data. SQL databases provide robustness, ACID compliance, and powerful querying capabilities, making them suitable for structured data and complex relationships.

On the other hand, NoSQL databases offer flexibility, scalability, and agility, making them suitable for semi-structured or rapidly evolving data schemas. However, they might lack the sophisticated querying capabilities of SQL databases, especially for complex relational queries.

Ultimately, the choice between SQL and NoSQL databases depends on the specific requirements of the application, including data structure, scalability needs, query complexity, and development agility. For the Pokemon database scenario, both SQL and NoSQL solutions can be viable depending on the specific use case and requirements.

7 Recommendation

Based on our experience and evaluation, we recommend considering the following factors when choosing between SQL and NoSQL databases for managing Pokemon data:

- If the data structure is well-defined and requires complex relational queries, SQL databases like MySQL or PostgreSQL are recommended.

- If the data schema is semi-structured and requires flexibility and scalability, NoSQL databases like MongoDB or Firestore are recommended.
- Consider the development agility and ease of scaling when making the decision, as these factors can significantly impact the long-term maintainability and performance of the application.

Overall, both SQL and NoSQL databases offer valuable solutions for managing Pokemon data, and the choice should be made based on the specific requirements and constraints of the project.