SQL VS NoSQL Database for Okainos

# SQL (MySQL)

SQL (Structured Query Language) is a language that allows you to write database queries (requests for data from a database table). With SQL you can write commands for fetching data, inserting data, combining tables, etc. This makes SQL very powerful in manipulating a database.

SQL databases works with multiple tables (cards table, score table, etc). These tables have a clear schema (logical structure of data) of which data can go into table. This structure is defined by fields. This means that when you insert a new record into a table, the record must contain the data for each field of the table. So, if a table has the fields card id and card name, every record will have data for each of the fields, whether they are null or not. This also means that a record cannot have an additional card type data without adding it as a field.

SQL databases are relational meaning that we can partition the database into multiple tables and create relational links between them with the tables’ id field, as shown in the diagram below:

Graphical user interface, diagram

Description automatically generated

This allows you to get data from every field in the data base from a single record even though it is partitioned into different tables due to relationships between the tables.

This is one of the main advantages of SQL allowing you to normalise your data to distribute them into different tables which you can merge with SQL queries.

# NoSQL (MongoDB)

There are many no SQL solutions, but mongo is the most popular one. It’s called mongo because of its capability and efficiency of storing humongous amounts of data, which SQL can struggle with.

Instead of tables like in SQL database, in mongo databases we have something called collections. And each collection has ‘documents’ which is like the rows of data in a table. What makes these documents so special is that they don’t need to use the same schema as the rows in SQL tables; you can have multiple documents in a collection that have different fields. This means that there is no strict schema applied to the structure of your data. So, if there was a cards collection, you can have a document with card id, card name and card type and another document with card id, card name but no card type. The disadvantage of being able to have completely different documents in a collection is that specific formats are not restricted on the documents, but the advantage is that the structure is super flexible.

In a non-SQL environment, there is no relations between separate collections like the normalised tables in SQL databases. This means that even if there is a collection storing detail data on something like items, other collections which have data on a specific item cost can’t link to it in the item collection, but instead have it manually entered. This leads to data duplication, which is a drawback to this system for the cost of fetching queries efficiently for specific document in a specific collection rather than a specific record of all the fields of the database, including ones that are not needed.

Diagram

Description automatically generated

# SQL VS NoSQL

|  |  |
| --- | --- |
| **SQL** | **NoSQL** |
| Data uses schemas, enforcing fields, data validation and predictability, but limits flexibility of the data | Does not use schemas, no strict enforcements providing flexibility of data in records, but making them less reliable in having common fields since they are not being enforced |
| Relationship links between tables, allowing data to be manually updated in only one located to be updated across all other connected tables automatically, but it can lead to worse performance due to the related data gets merged in a single record for each query | No relations, which is great for reading data a lot, but ignores data duplication and performance is affected when the same data is being manually updated across all collections |
| Data is partitioned into multiple tables | Data is typically merge/nested in a few collections to keep data that is being queried a lot in a single collection to prevent querying multiple collections manually |
| Horizontal scaling (adding more servers to work with the same database while maintaining performance) is difficult/impossible since the data can’t be split into multiple servers | Both horizontal and vertical scaling (increasing the capacity of servers to maintain performance) is possible but vertical scaling would have capacity limits of course |
| Limitations for lots of (thousands) read and write queries per second | Great performance for mass (simple) read and writes requests (unless a lot of collections are being updated at the same time regularly) |

# Conclusion

You’d usually run into the issues when handling large amounts of data.

**Reasons for choosing SQL:** You want to have a clear schema, you want a lot of relations, you work with data that changes frequently, you want it used on different parts of your application a lot, and you want to use complex commands in queries.

**Reasons for choosing NoSQL:** You want to read and write a lot, you don’t want to update the same data in multiple places (tables) at the same time in each write, you want to display your data as quick as possible, and you want to use simple commands in queries.