1. Explique el concepto de Write Concern en MongoDB. (10 pts)

Write concern describes the level of acknowledgment requested from MongoDB for write operations to a standalone `mongod` or to replica sets or to sharded clusters. In sharded clusters, `mongos` instances will pass the write concern on to the shards. Write concern can include the following fields:

`{ w: <value>, j: <boolean>, wtimeout: <number> }`

- the `w` option to request acknowledgment that the write operation has propagated to a specified number of `mongod` instances or to `mongod` instances with specified tags.
- the `j` option to request acknowledgment that the write operation has been written to the on-disk journal, and
- the `wtimeout` option to specify a time limit to prevent write operations from blocking indefinitely.

w Option

The `w` option requests acknowledgment that the write operation has propagated to a specified number of `mongod` instances or to `mongod` instances with specified tags.

j Option

The `j` option requests acknowledgment from MongoDB that the write operation has been written to the on-disk journal.

wtimeout

This option specifies a time limit, in milliseconds, for the write concern. `wtimeout` is only applicable for `w` values greater than `1`. `wtimeout` causes write operations to return with an error after the specified limit, even if the required write concern will eventually succeed. When these write operations return, MongoDB does not undo successful data modifications performed before the write concern exceeded the `wtimeout` time limit.

2. Explique diferencias entre bases de datos NoSQL y SQL, tome como ejemplo las bases de datos estudiadas en clase y utilizadas en proyectos y tareas: (40 pts)

1. SQL databases are relational, NoSQL databases are non-relational.
2. SQL databases use structured query language and have a predefined schema. NoSQL databases have dynamic schemas for unstructured data.
3. SQL databases are vertically scalable, while NoSQL databases are horizontally scalable.
4. SQL databases are table-based, while NoSQL databases are document, key-value, graph, or wide-column stores.

5. SQL databases are better for multi-row transactions, while NoSQL is better for unstructured data like documents or JSON.

Database Architecture

At the most basic level, the biggest difference between these two technologies is that SQL databases are relational, while NoSQL databases are non-relational.

Database Schemas and Query Languages

SQL databases use structured query language and have a pre-defined schema for defining and manipulating data. SQL is one of the most versatile and widely used query languages available, making it a safe choice for many use cases. It’s perfect for complex queries. However, SQL can be too restrictive. You have to use predefined schemas to determine your data structure before you can work with it. All of your data must follow the same structure. This process requires significant upfront preparation. If you ever wanted to change your data structure, it would be difficult and disruptive to your whole system.

NoSQL databases have dynamic schemas for unstructured data, and the data is stored in many ways. You can use column-oriented, document-oriented, graph-based, or KeyValue store for your data. This flexibility means that:

- You can create documents without having to first define their structure
- Each document can have its own unique structure
- The syntax can vary from database to database
- You can add fields as you go

Database Scaling

SQL databases are vertically scalable in most situations. You’re able to increase the load on a single server by adding more CPU, RAM, or SSD capacity. NoSQL databases are horizontally scalable. You’re able to handle higher traffic by sharding, which adds more servers to your NoSQL database. Horizontal scaling has a greater overall capacity than vertical scaling, making NoSQL databases the preferred choice for large and frequently changing data sets.

Data Structure

SQL databases are table-based, while NoSQL databases are document, key-value, graph, or wide-column stores. Some examples of SQL databases include MySQL, Oracle, PostgreSQL, and Microsoft SQL Server. NoSQL database examples include MongoDB, BigTable, Redis, RavenDB Cassandra, HBase, Neo4j, and CouchDB.

- MongoDB
- Elasticsearch
- SQL Server
- MySQL
- PostgreSQL

3. Desde un punto de vista de una base de datos de series de tiempo, ¿Porqué la localidad de datos es relevante para la

escogencia del hardware a utilizar?, puede justificar su respuesta utilizando los data tiers de algún motor de bases de datos como Elasticsearch. (40 pts)

4. Explique el concepto de Federated Queries y el impacto que tienen estas en el rendimiento de bases de datos. (10 pts)

Another piece of data.world-specific functionality is the ability to join tables which are in different datasets and which might not even have the same owner. Normally querying in SQL is database or dataset-specific. However as more and more datasets come on-line, it’s increasingly important to be able to pull information from combinations of them to present a more complete picture of the data.

Queries that join tables from different datasets are called federated queries. Their syntax is very similar to that of other join queries with additional location information specified.

In the CRM dataset we have been using, we have a table called accounts which has a small bit of information about our client accounts. We have recently learned that one of the regional offices has a dataset with the id “crm-account-stats” which has the table “external_account” which contains other information about the accounts, but doesn’t have some of the data we have. If we combined these tables, we could run a federated query like the following against them:

```
SELECT accounts.account,
       accounts2.sector,
       accounts2.year_established,
       accounts2.subsidiary_of,
       accounts.revenue
  FROM siyeh.`crm-account-stats`.external_account AS accounts2, accounts
 WHERE accounts2.account = accounts.account
```