Bases de Datos SQL vs NoSQL

Bases de Datos SQL:

Las bases de datos SQL se basan en un modelo relacional, lo que significa que organizan los datos en tablas relacionadas entre sí mediante el uso de llaves primarias y llaves foráneas. Utilizan un lenguaje de consulta estructurado (SQL) para realizar operaciones como inserción, actualización, eliminación y consulta de datos entre otros. Algunos ejemplos de bases de datos SQL son:

- MySQL.
- PostgreSQL.
- Oracle.
- SQL Server.

Estas bases de datos son ideales para aplicaciones que requieren transacciones complejas y relaciones entre datos bien definidas. Son altamente estructuradas y garantizan la integridad de los datos mediante la aplicación de restricciones de integridad y la realización de operaciones ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad). Por lo tanto, son una excelente opción para sistemas de gestión de contenidos, sistemas de gestión de relaciones con clientes (CRM), aplicaciones financieras y sistemas de gestión de inventario, entre otros.

Bases de Datos NoSQL:

Por otro lado, las bases de datos NoSQL adoptan un enfoque más flexible y escalable para almacenar datos. Están diseñadas para manejar grandes volúmenes de datos no estructurados o semiestructurados, y a menudo se utilizan en entornos distribuidos. Las bases de datos NoSQL pueden adoptar varias formas, incluidas las bases de datos de documentos, de clave-valor, de columnas y de grafos. Ejemplos populares son:

- MongoDB.
- Cassandra
- Redis.
- Neo4j.

Las bases de datos NoSQL son adecuadas para casos de uso donde la escalabilidad horizontal es crucial, como en aplicaciones web de alta carga, análisis de big data, IoT (Internet de las cosas) y sistemas de gestión de contenido multimedia. Al eliminar la estructura rígida de las bases de datos relacionales, las bases de datos NoSQL permiten una mayor flexibilidad en la organización y manipulación de datos.

Selección de la Base de Datos Adecuada:

Al elegir entre una base de datos SQL y NoSQL, es importante considerar diversos factores. Si la aplicación requiere una estructura de datos clara y transacciones complejas, una base de datos SQL podría ser la mejor opción. Por otro lado, si la escalabilidad y la flexibilidad son más importante, una base de datos NoSQL podría ser más adecuada.

Análisis de Requerimiento.

Servicio que únicamente se dedica a enviar notificaciones por distintos canales (SMS, PUSH, EMAIL).

En este caso lo mejor seria utilizar una base de datos NoSQL. Al existir distintos canales cada tipo de notificación será diferente así que la forma más sencilla de almacenar los diversos datos requeridos es aprovechar la capacidad que tienen las BD NoSQL de no tener una estructura fija, lo que facilita la adaptación a las todas las notificaciones.

Además, al ser enviadas en gran volumen y esperarse que se envíen en tiempo real se necesitara un rendimiento superior, que las bases de datos NoSQL ofrecen al enfrentarse fácilmente a escenarios de alto volumen y alta concurrencia.

Conectarse a una BD SQL y NoSQL.

Node.js puede conectarse tanto a BD SQL y NoSQL utilizando bibliotecas especificas aquí un ejemplo de cada una

Base de Datos SQL (MySQL).

Para conectarnos a MySQL con ayuda de Node.js es necesario contar con la librería ‘mysql’ que se instala con el siguiente código.

npm install mysql.

Posteriormente podemos conectarnos de la siguiente forma:

```
const mysql = require('mysql');

// Configuración de la conexión

const connection = mysql.createConnection({

  host: 'localhost',

  user: 'usuario',

  password: 'contraseña',
```

```
  database: 'nombre_base_de_datos'
});

// Establecer la conexión
connection.connect((err) => {
  if (err) {
    console.error('Error al conectar a MySQL: ' + err.stack);
    return;
  }
  console.log('Conexión exitosa a MySQL con el ID ' + connection.threadId);
});

/* Una vez que hayamos terminado de usar la conexión, no debemos olvidar cerrarla de no hacerlo podemos saturar las conexiones disponibles a la base de datos. */
connection.end();
```

Base de Datos NoSQL (MongoDB)

De igual forma para conectarnos a MongoDB en Node.js es necesario contar con su librería ‘mongodb’ que se instala con el siguiente código.

```
npm install mysql
```

Posteriormente podemos conectarnos de la siguiente forma:

```
const { MongoClient } = require('mongodb');

// URL de conexión a la base de datos MongoDB
const url = 'mongodb://localhost:27017';

// Nombre de la base de datos
const dbName = 'nombre_base_de_datos';
```

```
// Crear un nuevo cliente de MongoDB
const client = new MongoClient(url);

// Establecer la conexión
client.connect((err) => {
  if (err) {
    console.error('Error al conectar a MongoDB: ', err);
    return;
  }
  console.log('Conexión exitosa a MongoDB');

  // Trabajar con la base de datos...
});

/* Una vez que hayamos terminado de usar la conexión, no debemos olvidar cerrarla de no hacerlo podemos saturar las conexiones disponibles a la base de datos. */
client.close();
```