Tarea Patrones de diseño

Carlos Noguez Juárez

April 2024

Las bases de datos SQL (Structured Query Language) y NoSQL (No Solo SQL) son dos enfoques diferentes para el almacenamiento y recuperación de datos. Veamos las diferencias principales:

Bases de Datos SQL:

- Las bases de datos SQL son relacionales y se basan en un modelo de datos tabular.
- Utilizan el lenguaje SQL para consultar y manipular los datos.
- Los datos se organizan en tablas con filas y columnas, y se establecen relaciones entre las tablas mediante claves primarias y foráneas.
- Ofrecen integridad de datos y transacciones ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad).
- Son adecuadas para aplicaciones que requieren una estructura de datos fija y consistente.
- Ejemplos: MySQL, PostgreSQL, Oracle, SQL Server.

Bases de Datos NoSQL:

- Las bases de datos NoSQL son no relacionales y no se basan en un modelo de datos tabular.
- Cada base de datos NoSQL tiene su propio enfoque para almacenar y recuperar datos, como clave-valor, documentos, columnas, grafos, etc.
- Son flexibles y escalables horizontalmente.
- Pueden manejar grandes volúmenes de datos no estructurados o semiestructurados.
- No garantizan las propiedades ACID, pero ofrecen un modelo de consistencia eventual.
- Son adecuadas para aplicaciones que requieren alta disponibilidad, escalabilidad y flexibilidad en la estructura de datos.
- Ejemplos: MongoDB (documentos), Redis (clave-valor), Cassandra (columnas), Neo4j (grafos).

1 Implementación en Node.js

1.1 SQL

1. Instalar el módulo MySQL:

```
npm install mysql
```

2. Crear una conexión:

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

const connection = mysql.createConnection({
  host: 'localhost',
  user: 'tu_usuario',
  password: 'tu_contrase a',
  database: 'nombre_de_tu_base_de_datos',
});
```

Listing 1: JavaScript

3. Manejar errores de conexión:

```
connection.connect((err) => {
  if (err) {
    console.error('Error al conectar a la base de datos
        :', err);
    return;
  }
  console.log('Conexi n exitosa a la base de datos.');
});
```

Listing 2: JavaScript

4. Ejecutar consultas SQL:

```
connection.query('SELECT * FROM mi_tabla', (err, results
    ) => {
  if (err) {
    console.error('Error al ejecutar la consulta:', err)
        ;
    return;
  }
  console.log('Resultados:', results);
});
```

Listing 3: JavaScript

5. Cerrar la conexión:

```
connection.end((err) => {
  if (err) {
    console.error('Error al cerrar la conexi n:', err);
    return;
  }
  console.log('Conexi n cerrada correctamente.');
});
```

Listing 4: JavaScript

1.2 NoSQL

1. Instalar el módulo de MongoDB:

```
npm install mongodb
```

2. Configurar la conexión:

```
const { MongoClient } = require('mongodb');
const url = 'tu_mongo_url';
const client = new MongoClient(url);
```

Listing 5: JavaScript

3. Definir modelos y esquemas:

```
npm install mongoose

const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const miEsquema = new Schema({
  nombre: String,
  // otros campos...
});

const MiModelo = mongoose.model('MiModelo', miEsquema);
```

Listing 6: JavaScript

4. Realizar operaciones CRUD:

   - Crear:

```
MiModelo.create({ nombre: 'Ejemplo' }, function(err,
    doc) {
  // manejar error o trabajar con el documento
```

```
});
```

Listing 7: JavaScript

- Leer:

```
MiModelo.find({ nombre: 'Ejemplo' }, function(err, docs) {
  // manejar error o trabajar con los documentos
});
```

Listing 8: JavaScript

- Actualizar:

```
MiModelo.updateOne({ nombre: 'Ejemplo' }, { nombre: 'Nuevo Nombre' }, function(err, res) {
  // manejar error o trabajar con la respuesta
});
```

Listing 9: JavaScript

- Eliminar:

```
MiModelo.deleteOne({ nombre: 'Ejemplo' }, function(err) {
  // manejar error
});
```

Listing 10: JavaScript

Requerimiento de servicio de notificaciones: Para un servicio de notificaciones que maneja datos simples (como destinatarios, mensajes, canales de entrega), una base de datos NoSQL podría ser una opción adecuada debido a las siguientes razones:

1. **Escalabilidad:** Las bases de datos NoSQL están diseñadas para escalar horizontalmente y manejar grandes volúmenes de datos, lo cual es crucial para un servicio de notificaciones que podría tener un alto tráfico.

2. **Flexibilidad de esquema:** Como los datos de notificaciones pueden variar en estructura (diferentes campos para diferentes canales), una base de datos NoSQL como MongoDB (basada en documentos) sería una buena opción, ya que permite una estructura flexible de datos.

3. **Rendimiento de escritura:** Las bases de datos NoSQL suelen tener un buen rendimiento para operaciones de escritura intensiva, lo cual es importante para un servicio de notificaciones que recibe y procesa grandes cantidades de solicitudes.

Sin embargo, si el servicio de notificaciones requiere operaciones complejas de consulta, integridad de datos estricta o soporte para transacciones ACID, una base de datos SQL podría ser una mejor opción. En ese caso, se puede optar por una base de datos SQL y utilizar estrategias de particionamiento y escalamiento horizontal para manejar el alto volumen de datos.