Universidad Autónoma de Querétaro

Facultad de informática

Patrones de Diseño

Alumno: Diego Octavio Nieves Terrazas

307047

Fecha de entrega: 4 de mayo de 2024

Bases de Datos NoSQL vs SQL

Bases de Datos SQL (Relacionales):

Utilizan el lenguaje SQL (Structured Query Language) para consultar y manipular datos.

Datos se organizan en tablas con filas y columnas

Mantienen relaciones entre tablas mediante claves primarias y foráneas.

Son ideales para aplicaciones que necesitan transacciones complejas y consistentes, como sistemas bancarios o de reservas.

Bases de Datos NoSQL (No Relacionales):

Diseñadas para manejar grandes volúmenes de datos no estructurados o semiestructurados.

No requieren un esquema fijo, lo que permite una mayor flexibilidad.

Pueden ser de diferentes tipos, como documentos, columnares, clave-valor o gráficas.

Son útiles para aplicaciones que necesitan escalabilidad horizontal, como redes sociales, IoT (Internet de las cosas) y análisis de big data.

Casos de Uso Comunes:

SQL:

- Aplicaciones financieras
- Sistemas de gestión de inventario
- Sistemas de reservas y reservas
- Sistemas de gestión de contenido

NoSQL:

- Redes sociales y aplicaciones web
- Análisis de big data y procesamiento en tiempo real
- Aplicaciones IoT (Internet de las cosas)
- Almacenamiento de datos JSON o XML

Implementación en Node.js

Bases de Datos SQL:

Para trabajar con bases de datos SQL en Node.js, puedes usar bibliotecas como mysql, pg (para PostgreSQL), o sequelize (un ORM compatible con múltiples bases de datos SQL).

Estas bibliotecas te permiten conectarte a la base de datos, ejecutar consultas SQL y manejar los resultados de manera asincrónica.

Bases de Datos NoSQL:

Para bases de datos NoSQL como MongoDB, puedes usar el controlador oficial de MongoDB para Node.js llamado mongodb.

Este controlador te permite conectarte a una base de datos MongoDB, realizar consultas y manipular documentos JSON de manera eficiente.

También hay bibliotecas que proporcionan abstracciones más altas sobre MongoDB, como mongoose, que es un ODM (Object Document Mapper) que simplifica la interacción con la base de datos.

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

// Conexión a la base de datos
mongoose.connect('mongodb://localhost:27017/mi_base_de_datos', { useNewUrlParser: true, useUnifiedTopology: true });

// Definición de un esquema
const Schema = mongoose.Schema;
const userSchema = new Schema({
  name: String,
  age: Number
});

// Definición de un modelo
const User = mongoose.model('User', userSchema);

// Crear un nuevo usuario
const newUser = new User({ name: 'John', age: 30 });
newUser.save((err, user) => {
  if (err) return console.error(err);
  console.log('Usuario creado:', user);
});

// Consultar usuarios
User.find({}, (err, users) => {
  if (err) return console.error(err);
  console.log('Usuarios encontrados:', users);
});
```