Guía Bases de datos NoSQL – Programación de Software
Actividad 1: Transitando por SQL y NoSQL

1

Términos SQL:

1. SQL: Lenguaje para gestionar bases de datos relacionales, permitiendo realizar consultas y modificar datos.

2. Base de Datos Relacional: Organiza datos en tablas con filas y columnas, manteniendo relaciones entre los datos.

3. Transacción: Conjunto de operaciones que se ejecutan como una unidad, garantizando integridad en la base de datos.

4. DDL: Lenguaje SQL para definir estructuras de bases de datos, como tablas, índices y esquemas.

5. DML: Subconjunto de SQL para manipular datos, incluyendo operaciones como `INSERT`, `UPDATE`, `DELETE`.

6. Normalización: Proceso de estructuración para minimizar redundancias y asegurar la integridad de los datos.

7. Índice: Estructura que optimiza la velocidad de consultas al organizar los datos para acceso más rápido.

Términos NoSQL:

1. NoSQL: Bases de datos no relacionales que permiten manejar grandes volúmenes de datos no estructurados.

2. Base de Datos de Documentos: Almacena datos en documentos JSON, permitiendo estructuras flexibles y variadas.

3. Base de Datos de Grafos: Maneja relaciones complejas entre datos utilizando nodos y aristas, útil en redes sociales.

4. Base de Datos de Clave-Valor: Almacena datos en pares clave-valor, proporcionando acceso rápido y eficiente a la información.

5. Desnormalización: Técnica que mejora el rendimiento al duplicar datos, reduciendo la necesidad de uniones complejas.

6. Eventual Consistency: Modelo en bases NoSQL donde los datos se sincronizan eventualmente en todos los nodos.

7. Escalabilidad Horizontal: Capacidad de aumentar el rendimiento añadiendo más servidores, clave en NoSQL.

2

Ventajas de SQL:

Consistencia y Estandarización: SQL es un lenguaje universal y estandarizado, ideal para gestionar datos estructurados con transacciones ACID.

Consultas Complejas: Permite realizar consultas avanzadas y relaciones entre múltiples tablas con integridad de datos asegurada.

Desventajas de SQL:

Escalabilidad Limitada: Escala mejor en vertical que en horizontal, lo que puede ser problemático con grandes volúmenes de datos.

Rigidez en Esquemas: Los esquemas fijos requieren predefinir la estructura de datos, dificultando la adaptación a cambios.

Ventajas de NoSQL:

Flexibilidad y Escalabilidad: Maneja datos no estructurados con facilidad y permite escalar en horizontal añadiendo servidores.

Rendimiento para Altas Cargas: Ideal para aplicaciones que requieren alta disponibilidad y manejo de grandes volúmenes de datos.

Desventajas de NoSQL:

Consistencia Eventual: Puede sacrificar la consistencia inmediata para mejorar la disponibilidad.

Falta de Estandarización: No hay un estándar universal como en SQL, lo que puede generar variabilidad entre implementaciones.

3

SQL	NoSQL

Requiere un esquema predefinido para organizar los datos de manera tabular y relacional, utilizando tablas con columnas y filas.	Permite esquemas flexibles y dinámicos para datos no estructurados, sin mucha necesidad de estructurar u organizar los datos antes de colocarlos.
Requiere un esquema rígido y definido antes de almacenar los datos, organizándolos en tablas.	Ofrece modelos de consistencia más flexibles, como consistencia eventual, lo que puede significar que los datos se sincronicen en diferentes momentos.
Opta por bases de datos relacionales que facilitan la integridad referencial mediante el uso de claves foráneas.	Utiliza esquemas dinámicos que permiten agregar, modificar o eliminar campos sin afectar la estructura general.
Es ideal para aplicaciones empresariales tradicionales que requieren transacciones seguras y consistentes.	Emplea modelos de datos flexibles como clave-valor, documentos, y grafos, que se adaptan a distintos tipos de datos.
Las consultas se realizan con SQL, un lenguaje estándar que es potente y universalmente reconocido.	Se adapta mejor a aplicaciones modernas que manejan grandes volúmenes de datos no estructurados, como redes sociales o análisis de big data.
Las transacciones son atómicas y garantizan que todas las operaciones se completen correctamente o no se ejecute ninguna.	Utiliza lenguajes de consulta específicos según el tipo de base de datos, como CQL para Cassandra o MapReduce para MongoDB.
Escala de manera vertical, lo que implica actualizar el hardware existente para mejorar el rendimiento.	Escala de manera horizontal, lo que facilita la adición de más nodos o servidores para gestionar el crecimiento.
Soporta operaciones complejas de agregación y unión que permiten obtener información detallada a partir de múltiples tablas.	Ejemplos de bases de datos NoSQL incluyen Couchbase, HBase, y DynamoDB.
Las bases de datos relacionales son maduras y cuentan con un soporte robusto, incluyendo herramientas de gestión avanzadas.	Las bases de datos NoSQL son más nuevas y todavía están en proceso de consolidación en cuanto a soporte y herramientas.
Ejemplos de bases de datos SQL incluyen SQL Server, MySQL, y PostgreSQL.	Se prefiere en sistemas donde la flexibilidad y la velocidad son más

	importantes que la estricta consistencia de los

4

Contextos de Uso de SQL

Bases de Datos Relacionales: Estructura en tablas con filas y columnas, ideal para datos bien definidos.

Ejemplos: MySQL, PostgreSQL.

Aplicaciones con Estructura Fija: Sistemas con datos estructurados y relaciones entre ellos, como sistemas financieros o CRM.

Ejemplos: Sistemas de gestión empresarial.

Consultas Complejas y Transacciones: Manejo de transacciones que requieren alta consistencia y consultas complejas.

Ejemplos: Aplicaciones financieras, comercio electrónico.

Integridad y Seguridad de Datos: Mecanismos robustos para asegurar la integridad y seguridad de los datos.

Ejemplos: Aplicaciones gubernamentales, sistemas de salud.

Contextos de Uso de NoSQL

Escalabilidad Horizontal: Manejo de grandes volúmenes de datos distribuidos en múltiples servidores.

Ejemplos: Redes sociales, big data.

Datos No Estructurados: Ideal para datos que no se ajustan a un esquema tabular, como documentos JSON.

Ejemplos: Blogs, foros.

Desarrollo Ágil: Flexibilidad en el esquema, facilitando cambios rápidos sin migraciones complejas.

Ejemplos: Startups y empresas tecnológicas.

Casos de Uso Específicos: Modelos para necesidades particulares como grafos o almacenamiento en clave-valor.

Ejemplos: Neo4j para redes sociales, Redis para cachés.

5

Claro, aquí tienes un resumen de características y tipos de licencia de cinco sistemas de bases de datos SQL y cinco NoSQL:

Sistemas de Bases de Datos SQL

5

1. MySQL

- Características: Popular, soporte para transacciones ACID, replicación.

- Licencia: GPL con versiones comerciales (MySQL Enterprise).

2. PostgreSQL

- Características: Extensible, soporte para JSON, cumplimiento de estándares SQL.

- Licencia: Licencia PostgreSQL (permisiva, similar a MIT).

3. Microsoft SQL Server

- Características: Herramientas de BI, soporte para transacciones ACID, integración con Microsoft.

- Licencia: Comercial, con ediciones gratuitas y de pago.

4. Oracle Database

- Características: Potente, soporte para transacciones ACID, herramientas avanzadas de BI.

- Licencia: Comercial; versión gratuita (Oracle Database XE).

5. MariaDB

- Características: Fork de MySQL, mejoras en rendimiento, soporte para transacciones ACID.

- Licencia: GPL.

Sistemas de Bases de Datos NoSQL

1. MongoDB

- Características: Almacena documentos JSON, escalabilidad horizontal, consultas flexibles.

- Licencia: Server Side Public License (SSPL), con versión comercial disponible.

2. Cassandra

- Características: Alta disponibilidad, escalabilidad horizontal, grandes volúmenes de datos.

- Licencia: Apache License 2.0.

3. Redis

- Características: Almacén en memoria, estructuras de datos avanzadas, alto rendimiento.

- Licencia: MIT.

4. Neo4j

- Características: Base de datos orientada a grafos, consulta con Cypher.

- Licencia: GPL (comunitaria), licencia comercial (Enterprise).

5. Couchbase

- Características: Documentos con caché en memoria, consultas SQL-like, alta disponibilidad.

- Licencia: Apache License 2.0 (comunitaria), versiones comerciales disponibles.

6

Instrucciones SQL

1. Selección de Datos

- Comando: `SELECT`

- Descripción: Recupera datos de una o más tablas.

2. Inserción de Datos

- Comando: `INSERT INTO`

- Descripción: Inserta nuevas filas en una tabla.

3. Actualización de Datos

- Comando: `UPDATE`

- Descripción: Modifica datos existentes en una tabla.

4. Eliminación de Datos

- Comando: `DELETE`

- Descripción: Elimina filas de una tabla.

5. Creación de Tablas

- Comando: `CREATE TABLE`

- Descripción: Define una nueva tabla en la base de datos.

Instrucciones NoSQL

1. Selección de Datos

- Comando: Depende del tipo de base de datos (e.g., `find` en MongoDB).

- Descripción: Recupera documentos de una colección o registros clave-valor.

2. Inserción de Datos

- Comando: Depende del tipo de base de datos (e.g., `insertOne` en MongoDB).

- Descripción: Inserta un nuevo documento o par clave-valor.

3. Actualización de Datos

- Comando: Depende del tipo de base de datos (e.g., `updateOne` en MongoDB).

- Descripción: Modifica documentos existentes o valores.

4. Eliminación de Datos

- Comando: Depende del tipo de base de datos (e.g., `deleteOne` en MongoDB).

- Descripción: Elimina documentos o pares clave-valor.

5. Creación de Colecciones

- Comando: Depende del tipo de base de datos (e.g., `createCollection` en MongoDB).

- Descripción: Define una nueva colección de documentos.

7

Para un proyecto pequeño como el mío, MariaDB es una excelente opción. Ofrece un rendimiento eficiente y compatibilidad con MySQL, es de código abierto con bajos costos operativos, y proporciona características avanzadas como motores de almacenamiento optimizados. Además, su capacidad de escalabilidad y replicación, junto con el soporte de una comunidad activa, la convierte en una solución robusta y flexible para mi biblioteca virtual.

8