Universidad Nacional Autónoma de México

Facultad de Ingeniería

Bases de Datos

Profesor(a): *Ing. Fernando Arreola Franco*
Semestre 2025-2

Tarea 1

Nombre de la Tarea
Investigación modelo orientado a objetos y y modelos NoSQL

Grupo: 1
Alumno:
Nava Benítez David Emilio
No. de Cuenta:
320291599

Cd. Universitaria a 8 de febrero de 2025

Modelo orientado a objetos

La orientación a objetos representa el mundo real y resuelve problemas a través de objetos, ya sean tangibles o digitales. Este paradigma tecnológico considera un sistema como una entidad dinámica formada de componentes. Un sistema sólo se define por sus componentes y la manera en que éstos interactúan.

Los manejadores de bases de datos orientados a objetos deben considerar lo siguiente:

- Definir sus propios tipos de datos.
- Gestionar grandes cantidades de datos.
- Manejar transacciones largas.
- Recuperar rápidamente objetos complejos.
- Utilizar lenguajes de consulta de objetos como OQL.
- Implementar mecanismos de seguridad basados en objetos.
- Definir reglas deductivas.

Características:

1.- Cada objeto tiene un nombre, atributos y operaciones

2.- Es una tecnología para producir modelos que reflejen un dominio de negocio y utiliza la terminología propia de tal dominio

3.- Cuenta con cinco conceptos subyacentes: objeto, mensajes, clases, herencia y polimorfismo

4.- Un objeto tiene un estado, un comportamiento y una identidad

5.- Los mensajes brindan una comunicación ente objetos

Persistencia en el modelo orientado a objetos

La persistencia es una característica necesaria de los datos en un sistema de bases de datos. En el caso de los sistemas de gestión de base de datos orientada a objetos (OODBMS por sus siglas en inglés), la persistencia implica almacenar los valores de atributos de un objeto con la transparencia necesaria para que el desarrollador de aplicaciones no tenga que implementar ningún mecanismo distinto al mismo lenguaje de programación orientado a objetos.

Modelos NoSQL

Cave-Valor:

El modelo de base de datos NoSQL de clave-valor es uno de los más simples dentro de la familia NoSQL. Se basa en almacenar datos como un par clave-valor, donde cada clave es única y apunta a un valor correspondiente. Este tipo de bases de

datos está optimizado para operaciones rápidas de lectura y escritura y se emplea comúnmente en aplicaciones que requieren alto rendimiento y escalabilidad.

El almacenamiento de datos en clave-valor puede implementarse en diversas estructuras, como almacenamiento en memoria, bases de datos distribuidas o incluso en archivos simples. Las claves suelen ser cadenas de texto o identificadores únicos, mientras que los valores pueden ser cualquier tipo de datos, desde cadenas simples hasta objetos binarios complejos.

Ejemplos de bases de datos clave-valor incluyen Redis, Amazon DynamoDB, Riak y Apache Cassandra (en ciertos modos de operación).

Ventajas

1. Alto Rendimiento: Debido a su estructura simple, las operaciones de lectura y escritura son extremadamente rápidas, especialmente si la base de datos está en memoria (como Redis).
2. Escalabilidad Horizontal: Se puede distribuir fácilmente en múltiples servidores, lo que permite manejar grandes volúmenes de datos y alto tráfico.
3. Simplicidad: Su modelo de almacenamiento es fácil de comprender e implementar, lo que lo hace ideal para aplicaciones que no requieren estructuras de datos complejas.
4. Flexibilidad: Puede almacenar cualquier tipo de datos como cadenas, objetos JSON, datos binarios, etc.
5. Tolerancia a fallos: Algunas implementaciones cuentan con replicación y distribución de datos para evitar la pérdida de información en caso de fallas del sistema.

Desventajas

1. Falta de Relaciones: No admite consultas complejas ni relaciones entre los datos como en bases de datos relacionales (SQL).
2. Búsqueda limitada: Sin un sistema de indexación avanzado, la búsqueda dentro de los valores puede ser ineficiente.
3. Pérdida de Consistencia: En algunas implementaciones distribuidas, la consistencia puede sacrificarse para obtener mayor disponibilidad y particionamiento.
4. Uso Ineficiente del Espacio: Almacenamientos en memoria como Redis pueden consumir grandes cantidades de RAM si no se gestionan correctamente los datos.

5. Falta de soporte para consultas avanzadas: No se pueden realizar operaciones como agregaciones, joins o filtros avanzados sin diseños específicos adicionales.

Casos de Uso

1. Caché de Datos: Servicios como Redis se utilizan para almacenar temporalmente datos frecuentemente accedidos, reduciendo la carga en bases de datos principales y mejorando el tiempo de respuesta.
2. Sesiones de Usuario: Muchas aplicaciones web utilizan bases de datos clave-valor para manejar sesiones de usuario debido a su rápido acceso y almacenamiento volátil.
3. Colas de Mensajes: Algunos sistemas de mensajería y procesamiento en tiempo real utilizan bases de datos clave-valor para manejar colas de tareas.
4. Configuraciones y Preferencias: Servicios en la nube y microservicios almacenan configuraciones en bases de datos clave-valor debido a su acceso rápido y facilidad de actualización.
5. Procesamiento de Eventos en Tiempo Real: Aplicaciones como motores de recomendación y monitoreo de sistemas utilizan bases de datos clave-valor para manejar eventos en tiempo real y analizar patrones rápidamente.

Documentales:

El modelo de base de datos NoSQL documental almacena información en documentos estructurados, generalmente en formatos como JSON, BSON o XML. Cada documento es una unidad autónoma que contiene datos organizados en pares clave-valor, listas o estructuras anidadas, lo que permite una representación flexible y jerárquica de la información.

Este modelo es ampliamente utilizado en aplicaciones que requieren almacenamiento de datos semi-estructurados y acceso rápido a información específica sin necesidad de unir múltiples tablas, como en las bases de datos relacionales.

Ejemplos de bases de datos documentales incluyen MongoDB, CouchDB y Firebase Firestore.

Ventajas

1. Flexibilidad del Esquema: No requiere una estructura fija, lo que facilita cambios en los datos sin necesidad de alterar esquemas predefinidos.
2. Escalabilidad Horizontal: Se pueden distribuir los datos en múltiples servidores para manejar grandes volúmenes de información.

3. Alta Velocidad de Lectura y Escritura: Almacena datos en un solo documento, evitando uniones complejas, lo que mejora el rendimiento.
4. Fácil Integración con Aplicaciones Web y Móviles: Su estructura basada en JSON/BSON se ajusta naturalmente a APIs REST y entornos de desarrollo modernos.
5. Consulta Eficiente: Permite búsquedas indexadas y filtrado avanzado dentro de documentos sin necesidad de estructuras de datos externas.

Desventajas

1. Redundancia de Datos: La falta de normalización puede llevar a la duplicación de información y mayor consumo de almacenamiento.
2. Mayor Consumo de Memoria: Al almacenar datos con estructuras flexibles, los documentos pueden ocupar más espacio en comparación con bases de datos relacionales optimizadas.
3. Falta de Soporte para Relaciones Complejas: No maneja eficientemente relaciones entre diferentes documentos como lo hacen las bases de datos relacionales con el modelo de clave foránea.
4. Curva de Aprendizaje: Aunque es más flexible, los desarrolladores acostumbrados a SQL pueden necesitar tiempo para adaptarse a las consultas en bases de datos documentales.
5. Consistencia Eventual en Sistemas Distribuidos: En entornos distribuidos, algunas implementaciones sacrifican la consistencia fuerte para mejorar la disponibilidad y la tolerancia a fallos.

Casos de Uso

1. Aplicaciones Web y Móviles: Ideal para almacenar datos de usuario, configuraciones y contenido dinámico en aplicaciones de redes sociales y comercio electrónico.
2. Gestión de Contenidos: Plataformas de blogs, CMS y sistemas de almacenamiento de artículos pueden beneficiarse de la flexibilidad documental.

Grafos:

El modelo de base de datos NoSQL de grafos está diseñado para representar y gestionar relaciones complejas entre datos de manera eficiente. Utiliza nodos (entidades), aristas (relaciones) y propiedades para almacenar información de manera estructurada. A diferencia de las bases de datos relacionales, donde las relaciones requieren uniones costosas, las bases de datos de grafos permiten consultas rápidas mediante la navegación directa entre nodos conectados.

Ejemplos de bases de datos de grafos incluyen Neo4j, ArangoDB y Amazon Neptune.

Ventajas

1. Optimización para Relaciones Complejas: Maneja eficientemente consultas sobre datos interconectados sin necesidad de uniones complicadas.
2. Escalabilidad: Permite el crecimiento de la base de datos sin degradar el rendimiento en consultas de relaciones.
3. Alta Velocidad de Consultas: Gracias a la indexación basada en relaciones, las búsquedas y recorridos en grafos son más rápidas en comparación con modelos relacionales.
4. Modelado Natural de Datos Conectados: Representa datos en estructuras intuitivas que reflejan conexiones del mundo real, como redes sociales o sistemas de recomendación.
5. Flexibilidad: No requiere un esquema fijo, lo que permite adaptarse a cambios en los datos sin restricciones estructurales.

Desventajas

1. Curva de Aprendizaje: Requiere un cambio de paradigma en comparación con bases de datos relacionales tradicionales.
2. Menos Madurez en Herramientas: Aunque han crecido en popularidad, las bases de datos de grafos aún tienen menos herramientas y soporte que los modelos relacionales o documentales.
3. Almacenamiento y Consumo de Memoria: Puede requerir más espacio en disco y memoria RAM para optimizar el acceso a los datos.
4. Complejidad en Consultas de Datos No Relacionales: No es ideal para datos tabulares con pocas interconexiones, donde otros modelos pueden ser más eficientes.
5. Escalabilidad Limitada en Sistemas Distribuidos: Aunque algunas soluciones permiten la distribución de grafos, la fragmentación eficiente sigue siendo un desafío.

Casos de Uso

1. Redes Sociales: Representación de usuarios, conexiones, interacciones y recomendaciones basadas en relaciones.
2. Gestión de Redes: Modelado de infraestructuras de TI y telecomunicaciones, identificando puntos de fallo y optimizando rutas.

Bibliografía:

[1] E. Redmond y J. R. Wilson, *Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement*, 2nd ed. Raleigh, NC, USA: Pragmatic Bookshelf, 2017.

[2] P. Atzeni, L. Rossi, y P. A. Bernstein, "A Comparative Analysis of NoSQL Database Systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 2, pp. 271-288, Feb. 2018.

[3] “Modelo Orientado a Objetos”. Accedido el 5 de febrero de 2025. [Online]. Available: https://repositorio-uapa.cuaieed.unam.mx/repositorio/moodle/pluginfile.php/2731/mod_resource/content/1/UAPA-Modelo-Orientado-Objetos/index.html

Cd. Universitaria a 8 de febrero de 2025