Examen

Bases de Datos II

Luis Diego Delgado Muñoz

Prof. Nereo Campos

Fecha de Entrega: 01/06/2024

Pregunta 1

1. Motor de base de datos para la navegación entre distintos elementos de información

Para implementar la navegación entre los distintos elementos de información y descubrir relaciones, recomendaría el uso de un motor de base de datos de grafos, específicamente **Neo4j**. Las bases de datos de grafos son ideales para gestionar relaciones complejas entre grandes conjuntos de datos, permitiendo consultas eficientes y navegación a través de nodos y aristas que representan entidades y sus relaciones.

Justificación:

- **Relaciones entre elementos:** Las bases de datos de grafos como Neo4j están diseñadas para manejar relaciones complejas y de multiples conexiones. Los nodos pueden representar libros, artículos, y sitios web, mientras que las aristas pueden representar las relaciones entre ellos, tales como referencias, citas, y vínculos temáticos.
- **Eficiencia en la navegación:** Neo4j permite ejecutar consultas en tiempo real sobre grafos de gran tamaño, lo cual es crucial para la navegación interactiva y la exploración de datos similar a Wikipedia.
- **Modelo de datos:** Un pequeño modelo de datos para la Enciclopedia Galáctica podría ser el siguiente:

  ```
  (Libro)<--->[CITA]--->(Artículo)
  (Libro)<--->[REFERENCIA]--->(SitioWeb)
  (Artículo)<--->[CITA]--->(Libro)
  (Artículo)<--->[REFERENCIA]--->(SitioWeb)
  (SitioWeb)<--->[ENLACE]--->(Libro)
  (SitioWeb)<--->[ENLACE]--->(Artículo)
  ```

  Aquí, cada entidad (Libro, Artículo, SitioWeb) es un nodo, y las relaciones entre ellos son aristas con etiquetas como `CITA`, `REFERENCIA`, y `ENLACE`.

- **Palabras clave:** El motor de base de datos de grafos no necesita almacenar todo el contenido de los elementos de información, sino los metadatos y palabras clave que permitan establecer y navegar las relaciones. El contenido completo puede almacenarse en otro sistema optimizado para búsquedas de texto completo.

2. Motor de base de datos para almacenar los elementos de información y garantizar full text search

Para almacenar los elementos de información y garantizar full text search, recomendaría el uso de **Elasticsearch**. Elasticsearch es un motor de búsqueda altamente escalable y distribuido, diseñado para búsquedas de texto completo, análisis y almacenamiento de grandes volúmenes de datos.

Justificación:

- **Capacidad para full text search:**
  - Elasticsearch proporciona capacidades avanzadas de búsqueda de texto completo, incluyendo análisis de texto, tokenización, y entre otros.
  - Permite búsquedas rápidas y eficientes en grandes volúmenes de datos, lo cual es esencial para una enciclopedia galáctica con billones de documentos.
- **Particionamiento o sharding de datos:**
  - Elasticsearch soporta sharding nativamente, distribuyendo los datos en múltiples nodos para balancear la carga y aumentar la eficiencia.
  - Los datos se pueden replicar en múltiples shards y réplicas para mejorar la disponibilidad y tolerancia a fallos.
- **Representación de elementos de información:**
  - Los elementos de información se pueden representar como documentos JSON dentro de índices.
  - Un posible esquema JSON para un documento podría ser:

```
{
  "titulo": "Título del Libro",
  "autor": "Autor del Libro",
  "contenido": "Texto completo del libro...",
  "palabras_clave": ["palabra1", "palabra2", "palabra3"],
  "tipo": "libro"
}
```

3. Combinación de motores para búsqueda y navegación

Para crear un sistema que combine búsqueda y navegación, se puede seguir este enfoque:

- **Integración de Neo4j y Elasticsearch:**
  - **Almacenamiento y búsqueda:** Elasticsearch almacena y proporciona capacidades de búsqueda de texto completo.
  - **Navegación y relaciones:** Neo4j almacena las relaciones y permite la navegación eficiente entre elementos.
- **Proceso de búsqueda y navegación:**
  - **Búsqueda inicial:** Cuando un ususario realiza una búsqueda, Elasticsearch proporciona una lista de resultados relevantes.
  - **Navegación de relaciones:** Al seleccionar un resultado, el sistema consulta Neo4j para obtener las relaciones del documento seleccionado, permitiendo al usuario explorar documentos relacionados.

4. Alta disponibilidad de las bases de datos

Para garantizar alta disponibilidad:

- **Elasticsearch:**
  - Desplegar un clúster con múltiples nodos, utilizando sharding y réplicas para garantizar que los datos estén siempre disponibles, incluso si uno o más nodos fallan.
  - Implementar balanceadores de carga y políticas de failover para redirigir el tráfico en caso de fallos.
- **Neo4j:**
  - Desplegar en un clúster causal, donde múltiples instancias de Neo4j funcionan en conjunto para proporcionar alta disponibilidad y consistencia.
  - Utilizar réplicas y backups regulares para proteger los datos.

5. Garantizar tiempo de respuesta constante

- **Optimización de consultas:** Optimizar las consultas y los índices en Elasticsearch para mejorar la velocidad de búsqueda.
- **Cache:** Utilizar cache en capas de aplicación y bases de datos para almacenar resultados de consultas frecuentes.
- **Recursos escalables:** Desplegar recursos escalables en la nube que se pueden ajustar automáticamente según la carga de trabajo.

6. Uso de caches y localidad para mejorar el rendimiento

- **Caches en múltiples niveles:**
  - **Cliente:** Caches en el lado del cliente (navegador) para almacenar temporalmente los resultados de búsquedas y navegación.
  - **Servidor:** Caches en el lado del servidor para almacenar resultados de consultas frecuentes y reducir la carga en los motores de base de datos.
- **Localidad de datos:**
  - **Proximidad geográfica:** Desplegar nodos de Elasticsearch y Neo4j en ubicaciones geográficamente distribuidas para reducir la latencia.
  - **Red de entrega de contenido (CDN):** Utilizar CDNs para almacenar en caché contenido estático y mejorar la velocidad de entrega a los usuarios en diferentes regiones.

Pregunta 2

1. Definición de Índices

Para mejorar la velocidad de las consultas, es crucial definir índices adicionales como por ejemplo:

Tabla `artist`:

- **Índice de texto completo:** Para mejorar la búsqueda de artistas por nombre.

Tabla `album`:

- **Índice de texto completo:** Para mejorar la búsqueda de álbumes por nombre.

Tabla `song`:

- **Índice compuesto:** Para optimizar la búsqueda de canciones por nombre y género.

Tabla `artist_song`:

- **Índice compuesto:** Para optimizar las consultas de relación entre artistas y canciones.

Tabla `song_album`:

- **Índice compuesto:** Para optimizar las consultas de relación entre canciones y álbumes.

2. Recomendación de Base de Datos

Dado el patrón de uso de muchas lecturas y pocas escrituras, y las necesidades de búsquedas de texto completo, lo mejor sería el uso de **Elasticsearch** como base de datos principal para reemplazar la actual base de datos relacional. Elasticsearch está diseñado para manejar búsquedas de texto completo de manera eficiente, además de estar optimizado para cargas de trabajo con muchas lecturas. También, permite la creación de índices que pueden acelerar significativamente las consultas.

3. Mejoras en la Base de Datos Relacional

Para mejorar el rendimiento de la base de datos relacional, especialmente para la tercera consulta, se pueden considerar las siguientes estrategias:

Desnormalización:

- Almacenar información redundante en la tabla `song` que incluya nombres de artistas y álbumes puede reducir la necesidad de múltiples `JOINs`.
- Por ejemplo, agregar columnas `artist_name` y `album_name` a la tabla `song`.

Índices compuestos:

- Definir índices compuestos que incluyan las columnas utilizadas en las `JOINs` y en los filtros de la consulta.

Materialized Views:

- Crear vistas materializadas para las consultas más comunes, que pueden ser actualizadas periódicamente.

Pregunta 3

Para garantizar la seguridad al instalar y mantener un motor de base de datos en la nube, es esencial seguir una serie de buenas prácticas para cada uno de los componentes que se exponen en el diagrama (red pública, red privada y red de almacenamiento):

1. Red Pública (Aplicación)

Buenas Prácticas:

- **Cifrado del Tráfico (HTTPS/TLS):** Todo el tráfico entre la aplicación y el motor de base de datos debe estar cifrado utilizando HTTPS o TLS. Esto asegura que los datos en tránsito estén protegidos contra intercepciones y ataques de intermediario (Man-in-the-Middle).
- **Autenticación y Autorización Robusta:** Implementar mecanismos de autenticación y autorización robustos, como OAuth, JWT, o MFA (autenticación multifactor), para

asegurar que solo usuarios y servicios autorizados puedan acceder a la aplicación.

- **Firewall y Control de Acceso:** Utilizar firewalls para restringir el acceso a la aplicación a direcciones IP y puertos específicos. Aplicar políticas de lista blanca para limitar el acceso a la aplicación solo a usuarios y servicios autorizados.

Estas medidas protegen los datos en tránsito y aseguran que solo usuarios legítimos puedan interactuar con la aplicación, reduciendo así la superficie de ataque.

2. Red Privada (Instancia y Motor DB)

Buenas Prácticas:

- **Aislamiento de Red:** Configurar la base de datos en una red privada virtual (VPN) para aislar el tráfico de la base de datos del tráfico de red general. Esto añade una capa adicional de seguridad al restringir el acceso solo a través de conexiones seguras y autenticadas.
- **Segmentación de Red:** Dividir la red en subredes para separar la base de datos de otros servicios y limitar la superficie de ataque. Utilizar subredes privadas que no sean accesibles directamente desde la Internet pública.
- **Controles de Acceso Estrictos:** Implementar controles de acceso rigurosos a la instancia de la base de datos mediante políticas de IAM (Identity and Access Management). Asegurarse de que solo usuarios y servicios específicos tengan acceso, siguiendo el principio de mínimo privilegio.
- **Cifrado de Datos en Reposo:** Utilizar cifrado para los datos almacenados en la base de datos. Los proveedores de servicios en la nube, como AWS, Google Cloud y Azure, ofrecen cifrado automático de datos en reposo.

Estas prácticas garantizan que solo las conexiones seguras y autenticadas puedan acceder a la base de datos, y protegen los datos tanto en tránsito como en reposo.

3. Red de Almacenamiento (Disco)

Buenas Prácticas:

- **Cifrado de Datos en Reposo:** Asegurarse de que todos los datos almacenados en disco estén cifrados. La mayoría de los proveedores de servicios en la nube ofrecen cifrado automático para datos en discos de almacenamiento.
- **Backups y Restauración:** Configurar copias de seguridad regulares y pruebas de restauración para garantizar que los datos puedan ser recuperados en caso de pérdida o corrupción. Utilizar cifrado también para los backups.
- **Gestión de Acceso y Permisos:** Limitar el acceso al almacenamiento solo a aquellos servicios y usuarios que realmente lo necesitan, y auditar regularmente los permisos y accesos.

El cifrado de datos en reposo y en backups asegura que los datos estén protegidos contra accesos no autorizados. La gestión estricta de acceso y permisos reduce la probabilidad de fugas de datos.

Pregunta 4

1. Bases de Datos de Series de Tiempo en Soluciones de Observabilidad

Naturaleza de los Datos Recolectados: Las soluciones de observabilidad recolectan datos que cambian con el tiempo, como métricas de rendimiento, logs de eventos y trazas de solicitudes. Estos datos suelen ser recolectados a alta frecuencia y en grandes volúmenes, lo cual requiere una capacidad de almacenamiento y procesamiento eficiente. Las consultas más comunes sobre estos datos son de tipo temporal, como agregaciones sobre intervalos de tiempo, detección de tendencias y análisis de patrones.

Estas bases de datos de tiempo brindan grandes ventajas ya que están diseñadas para manejar eficientemente datos que son principalmente de naturaleza temporal. Implementan técnicas de compresión que reducen el espacio de almacenamiento, dado que los datos temporales suelen tener redundancia. Finalmente, proporcionan mecanismos optimizados para realizar consultas sobre intervalos de tiempo, permitiendo agregaciones y análisis rápidos.

2. Uso de BigTable como Base de Datos de Series de Tiempo

Naturaleza de los datos de BigTable:

- **Escalabilidad:** Google BigTable es altamente escalable y puede manejar grandes volúmenes de datos distribuidos, lo cual es ideal para datos de series de tiempo que se generan en grandes cantidades.
- **Modelo de Datos:** BigTable utiliza un modelo de datos basado en columnas, que es adecuado para almacenar y acceder a datos temporales con claves compuestas (e.g., combinación de una clave de fila con un timestamp).
- **Compatibilidad:** Puede integrarse con herramientas de observabilidad mediante APIs y conectores disponibles.

Tomando todo esto en cuenta BigTable es una buena opción para bases de datos de series de tiempo debido a su escalabilidad, modelo de datos flexible y capacidad de manejar grandes volúmenes de datos temporales de manera eficiente.

3. Ahorro de Dinero con Información Histórica en Elasticsearch

Estrategias:

- **Rollover y Index Lifecycle Management (ILM):** Utilizar políticas de ciclo de vida de índices para gestionar la retención y eliminación de datos antiguos.
- **Compresión y Reducción de Réplicas:** Configurar índices más antiguos con mayor compresión y menos réplicas para reducir costos de almacenamiento.
- **Cold Storage:** Mover datos históricos a almacenamiento más económico, como el almacenamiento en frío o archivado.

4. Comparación entre Soluciones On-Premise y Managed Services

Soluciones On-Premise:

Ventajas:

- **Control Total:** Mayor control sobre la configuración, personalización y seguridad de los sistemas.
- **Costo:** Puede ser más económico a largo plazo si se dispone de infraestructura propia.

Desventajas:

- **Mantenimiento:** Requiere recursos internos para mantener y gestionar la infraestructura.
- **Escalabilidad:** Escalar la infraestructura puede ser más complejo y costoso.

Managed Services:

Ventajas:

- **Facilidad de Uso:** Configuración y despliegue rápidos sin necesidad de gestionar ni comprar la infraestructura.
- **Escalabilidad:** Escalabilidad automática para manejar grandes volúmenes de datos sin intervención manual.
- **Soporte:** Soporte técnico especializado y actualizaciones automáticas.

Desventajas:

- **Costo:** Puede ser más costoso a largo plazo, especialmente con el crecimiento de los datos.
- **Menor Control:** Menor control sobre la personalización y la seguridad.

Durante la TC 1, se experimentó la implementación y gestión de varias herramientas de observabilidad (Prometheus, Grafana, etc.) utilizando Helm Charts. Según lo que pudimos notar, aunque el control y la personalización eran ventajosos, el conocimiento y capacidad de equipo requerido para mantener y escalar la infraestructura on-premise es más significativo. Por otro lado, un Managed Service como Datadog ofrece simplicidad y escalabilidad automática, aunque a un costo más alto.

Referencias bibliográficas

- [1] I. Neo4j, “Neo4j graph database & analytics – the leader in graph databases,” Graph Database & Analytics, https://neo4j.com/ (accessed Jun. 1, 2024).
- [2] B. V. Elasticsearch, “Elasticsearch: The Official Distributed Search & Analytics engine,” Elasticsearch, https://www.elastic.co/elasticsearch (accessed Jun. 1, 2024).
- [3] M. S. Learn, CREATE MATERIALIZED VIEW AS SELECT (Transact-SQL), https://learn.microsoft.com/en-us/sql/t-sql/statements/create-materialized-view-as-select-transact-sql?view=azure-sqldw-latest (accessed Jun. 1, 2024).
- [4] G. Cloud, “Bigtable documentation | bigtable documentation | google cloud,” Google, https://cloud.google.com/bigtable/docs (accessed Jun. 1, 2024).
- [5] B. V. Elasticsearch, “ILM: Manage the index lifecycle: Elasticsearch Guide [8.13],” Elastic, https://www.elastic.co/guide/en/elasticsearch/reference/current/index-lifecycle-management.html (accessed Jun. 1, 2024).