Examen Bases de Datos II

Isaac Araya Solano 2018151703

Pregunta 1 (60 pts)

Aproximadamente para el año 23651 de nuestra era y durante el apogeo del imperio galáctico, el matemático Hari Seldon ha desarrollado su teoría llamada Psicohistoria, mediante esta, ha podido predecir con un grado de confianza bastante alto la caída de la civilización seguida de un periodo de barbarie, con el fin de reducir este periodo de barbarie, este ha desarrollado un plan y como parte de este, se encuentra la conformación de la Enciclopedia Galáctica, la cual de acuerdo con el divulgador científico Carl Sagan es un sugerente proyecto del saber colectivo de las civilizaciones avanzadas del universo.

Usted ha sido escogido como líder técnico del equipo que se encargará de implementar la base de datos que mantendrá esta información con alta disponibilidad y con un mecanismo adecuado para navegar los datos y realizar búsquedas. Es importante mencionar:

- **La tecnología en bases de datos SQL y NoSQL no han cambiado desde el año 2023.**
- **No existe restricción en cuanto a dinero que se puede invertir en el proyecto.**
- **Los proveedores de Cloud siguen existiendo y ahora han expandido sus ubicaciones en prácticamente todo el universo conocido.**
- **Los productos ofrecidos en los proveedores de Cloud para el 2023 siguen siendo ofrecidos para el año 23651.**
- **Se tiene que permitir full text search sobre la información en la Enciclopedia Galáctica.**
- **Se tienen que establecer relaciones entre los diferentes elementos de información de forma tal que permita descubrir relaciones entre la**

información. Un excelente ejemplo de cómo funcionara la navegación es el sitio de Wikipedia.

- **La Enciclopedia Galáctica presenta un alto número de lecturas contra un bajo número de escrituras (prácticamente 0).**
- **Para el año 23651, se han escrito:**
  - **4 billones de libros con una media de 200 páginas.**
  - **1 billón de artículos científicos con una media de 10 páginas.**
  - **20 billones de sitios web con una media de 10 páginas cada uno.**

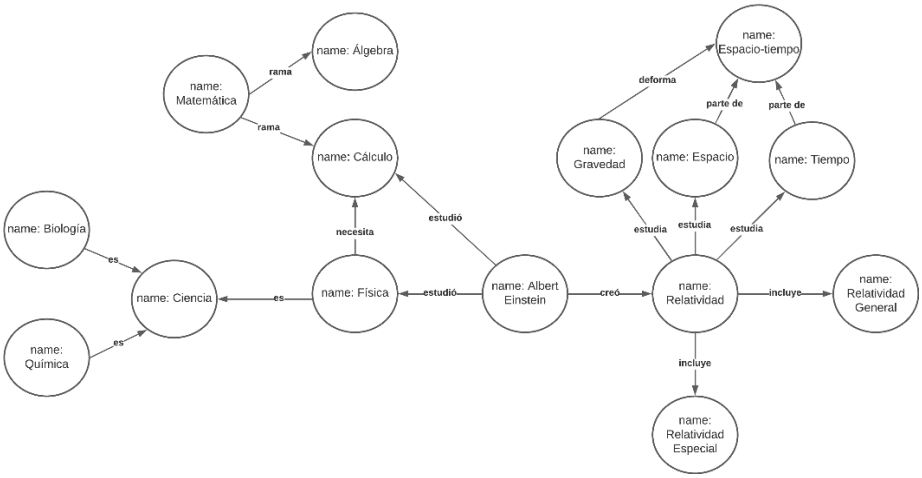
En su calidad de líder técnico, usted debe presentar una propuesta para dar respuesta a las siguientes preguntas:

1. ¿Qué motor de base

1. ¿Qué motor de base de datos utilizaría para implementar la navegación entre distintos elementos de información? ¿Es necesario que este motor de base de datos contenga todo el elemento de información o solo palabras clave que permitan establecer relaciones? Justifique su respuesta mediante la elaboración de un pequeño modelo de datos y las relaciones que establecería entre los diferentes elementos de información, lo más importante es garantizar una navegación y que permita descubrir relaciones. (20 pts)

R/ Para la implementación de la navegación entre datos utilizaría una base de datos basada en grafos como puede ser Neo4J. Esto porque las bases de datos de grafos permiten mucha más flexibilidad a la hora de crear relaciones entre información y al ser una plataforma pensada en Wikipedia que tiene múltiples relaciones, puede permitir conectar fácilmente muchos tipos de información.

Considero que el motor solo debería contener palabras clave que permitan establecer relaciones. Aquí hay un ejemplo de como se vería un modelo de datos para información almacenada en este sistema:



2. ¿Qué motor de base de datos utilizaría para almacenar los elementos de información y garantizar full text search? Justifique su respuesta comentando: (20 pts)

a.Capacidad del motor para implementar full text search.

b. Particionamiento o sharding de datos.

c. Representación de elementos de información en la base de datos (tablas, documentos, collections, etc.)

R/ Para almacenar la información y garantizar full search utilizaría como motor MongoDB. Esto principalmente porque MongoDB tiene la capacidad de implementar búsqueda full search utilizando tecnologías como Apache Lucene en implementaciones como la de MongoDB Atlas Full-text Search.

Además, ya que es una cantidad importante de información y datos los que se desean almacenar se debe tener en cuenta el particionamiento de datos. Para esto, MongoDB tiene lo que se conocen como colecciones que sería lo que se utilizaría para realizar la partición de los datos. Las colecciones son la estructura de datos que vendría a ser como las “tablas” de una base de datos relacional. Dentro de las colecciones tenemos

los documentos que almacenan la información y así podemos relacionar la información agrupando los documentos dentro de las colecciones. Por otro lado, cada documento tiene lo que se conoce como “campo” que permite almacenar las diferentes características, información y/o datos que se quieran guardar sobre ese documento en específico.

Para terminar, ya que la base de datos MongoDB es NoSQL, nos permite una estructura flexible de almacenamiento de información al usar JSON, por ende, cada documento podrá tener la información en los campos que considere necesarios y podría variar dependiendo de cada colección.

3. Describa la forma en la cual combinaría los dos motores anteriores (navegación y full text search) para crear un sistema simple de búsqueda y navegación de información similar al que tiene el sitio Wikipedia donde se busca un elemento de información y nos podemos mover entre términos. (5 pts)

Primero crearía un api, este api tendría un endpoint que buscaría dentro de MongoDB utilizando la búsqueda full text para encontrar el documento solicitado. Luego el api llamaría a la base Neo4J para encontrar las relaciones posibles dentro del documento y a partir de ahí se devolvería la palabra clave y por medio de esa palabra clave se generarían las palabras a subrayar dentro del documento al mostrarlo en la pagina del sistema. Luego habría otro endpoint que reciba la palabra relacionada y el documento actual y devolvería desde mongo el documento relacionado para mostrar.

4. ¿De qué forma garantizaría alta disponibilidad de las bases de datos? (5 pts)

Haría muchas replicas de lectura y unas cuantas de escritura para generar un sistema multimaster. El modelo de multimaster es lento al realizar escrituras por tener el principio de siempre mantener la consistencia, pero al recibir un flujo tan bajo de escrituras entonces no habría ningún problema. Además, para asegurarme de que los usuarios no puedan percibir ninguna caída de un sistema haría que todas las conexiones a la base de los usuarios pasen primero por un proxy. Este proxy se encargaría de recibir el flujo de consultas y solicitudes y distribuiría estas a través de

las distintas replicas de la base. Esto evitaría que los usuarios puedan darse cuenta de si una replica se cae o tiene algún problema.

5. ¿Cómo podría garantizar que las búsquedas siempre tengan un tiempo de respuesta constante? (5 pts)

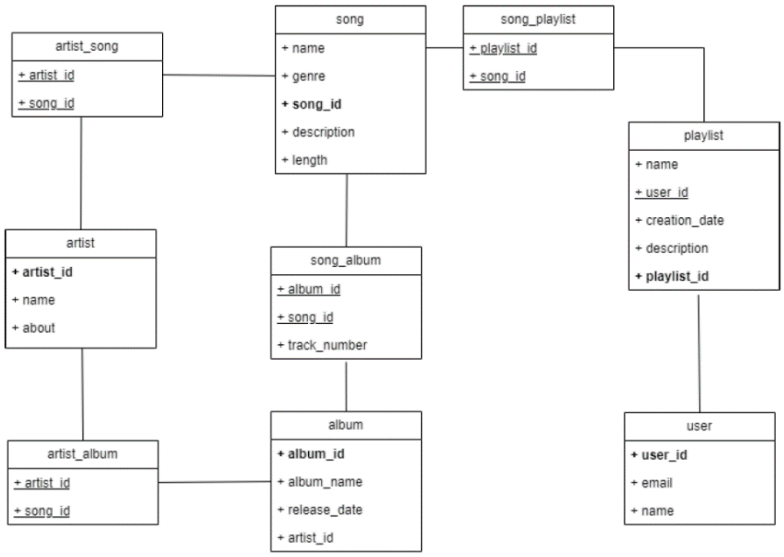
Pondría distintas Availability Zones, asegurándome que exista una distribución pareja a través de la galaxia para que las respuestas que reciban los usuarios tengan un tiempo de respuesta equivalente, sin importar en que región de la galaxia se encuentren. Cada availability zone tendría su réplica de escritura y distintas réplicas de lectura en función de la cantidad de queries recibidas para asegurarme que no haya colas de espera para la respuesta de las lecturas.

6. ¿Cómo el uso de caches y localidad podría mejorar el rendimiento del sistema? (5 pts)

A la hora de mejorar el rendimiento usando caches y localidad lo que podemos hacer es, analizando la base de grafos de Neo4J, podemos darnos cuenta de la información que está más cercanamente relacionada a la que se está mostrando, entonces se puede cargar al cache la información más relacionada, haciendo uso de la localidad. Esto permitiría que si un usuario quiere acceder a esta información relacionada ya tengamos esa información pre-cargada en el caché, por lo que se ahorraría todo el tiempo de ir a traer esa información a disco y reduciría los tiempos de carga y haría ágil la navegación entre temas relacionados.

Pregunta 2 (10 pts)

El siguiente diagrama representa una versión simplificada de un sistema de reproducción de música que utiliza una base de datos relacional:



Este sistema tiene varios vicios o problemas de normalización, así como el grave problema de que no tiene definidos índices, en conjunto esto ha causado que se esté experimentando muchos timeouts y la solución convencional de agregar más hardware se ha vuelto insostenible. Luego de un estudio del workload de la base de datos, se llegó a las siguientes conclusiones:

- **Es necesario definir algunos índices fuera de los que son definidos automáticamente mediante llaves primarias y foráneas.**
- **Un motor de base de datos relacional no parece ser el más adecuado para el problema.**
- **El patrón de uso es muchas lecturas contra pocas escrituras.**

Mediante los logs de acceso y los logs de slow queries, se ha encontrado que los siguientes queries son los más usuales y problemáticos en tiempo que tardan en ejecutarse:

SELECT name FROM artist WHERE name like '%{text}%'

SELECT name FROM album WHERE name like '%{text}%'

SELECT a.name as artist_name, al.name as album_name, s.name, s.genre, sa.track_number, s.length, s.description FROM artist a INNER JOIN artist_song as ON a.artist_id = as.artist_id INNER JOIN song s ON as.song_id = s.song_id INNER JOIN song_album sa ON s_song_id = sa.song_id INNER JOIN album al ON sa.album_id = al_album_id WHERE a.name = '%{name}%' AND al.name = '%{name}%' and s.name = '%{name}%'

Como administrador o administradora de bases de datos, elabore respuestas a las siguientes preguntas:

- **¿Qué índices definiría para aumentar la velocidad de todo el sistema? Tome en cuenta todos los tipos de índices estudiados en el curso. (3 pts)**

  Definiría un índice non-clustered sobre la columna artist.name, uno sobre album.name y uno sobre song.name. Estos 3 índices acelerarían en general las 3 consultas más problemáticas hasta el momento y al ser non-clustered, estarían cargados en memoria principal, acelerando la carga y búsqueda de esta información.

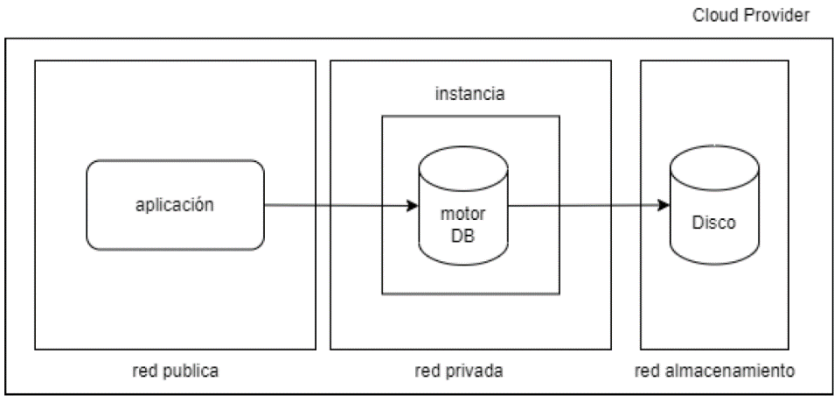
- **¿Qué base de datos SQL o NoSQL recomendaría para reemplazar la base de datos actual? Justifique su respuesta. (3 pts).**

  Yo utilizaría MongoDB para reemplazar esta base de datos, esto porque esta base permite la creación fácil de índices muy eficientes, permitiendo consultas muy variadas y rápidas. Además, MongoDB maneja fácilmente cantidades de datos grandes, haciendo que la escalabilidad sea fácil, y dado que la cantidad de canciones presentes es alta y la música no para de crecer, hace que sea un factor importante para tener en cuenta. Finalmente, al tener problemas con la normalización de la base, el modelo de datos flexible que tiene MongoDB hace que no se requiera un esquema fijo, permitiendo que se defina libremente como se quiere almacenar la información.

- **¿Existirá alguna otra forma de mejorar el rendimiento de la base de datos relacional en especial para la tercera consulta? Comente. (4 pts)**
  Podría crearse una vista de la tercera consulta, pero sin las condiciones Where y almacenarlo. Esta vista nos serviría porque a la hora de realizar la consulta, nos ahorraríamos todo el tiempo que se tomaría hacer cada inner join y podríamos simplemente filtrar sobre esa vista con las condiciones where del final.

Pregunta 3 (20 pts)

En los últimos 15 años, la forma en la cual se mantienen e instalan servidores de bases de datos ha cambiado considerablemente, la aparición del Cloud ha proporcionado muchas ventajas para la instalación y mantenimiento, pero ha inducido nuevos problemas de seguridad y nuevas soluciones, en el siguiente diagrama se muestra una arquitectura típica de una base de datos en un Cloud Provider:



Tomando como referencia el diagrama anterior, ¿Cuáles son las buenas prácticas en términos de seguridad que se deben seguir cuando se instala un motor de base de datos en el Cloud? Fundamente su respuesta hablando de la seguridad de cada uno de los componentes que se exponen en el diagrama.

Primero hay que tener en cuenta la seguridad para la aplicación en la red pública. En este componente lo principal es colocar un firewall, esto lo que permite es que se pueda limitar los puertos necesarios que estarán abiertos. Además, se debe agregar

autenticación de usuario para validar las personas que pueden utilizar la aplicación. Para los usuarios es importante implementar el Principio de Mínimo privilegio, que es básicamente darle los permisos justos y limitados a los usuarios para que hagan las funciones que tienen que hacer, pero sin tener acceso a nada que no deberían. Importante tener en cuenta que no se deben tener bases de datos en la red publica nunca, siempre en la red privada.

Luego, para el motor de la base de datos en la red privada, también debe de establecerse un firewall que limite las conexiones de la aplicación a la base, brindando puertos específicos a los que pueda conectarse. Además, debe limitarse que la única ip que pueda conectarse a la base sea la de la aplicación.

Para la red de almacenamiento, la seguridad sería primero establecer un firewall para la conexión entre la red privada y el almacenamiento, igualmente definiendo puertos específicos para la transferencia de información y finalmente encriptar la información almacenada en los discos para que, aunque se acceda a ellos, la información se mantenga protegida.

Finalmente, algunas recomendaciones para todos los componentes es utilizar herramientas de observabilidad para monitorear la actividad de los usuarios y así detectar en cualquier caso cuando un usuario esta haciendo actividades inusuales como acceder a horas que nunca accedería, desde ubicaciones que nunca ha accedido o haciendo acciones que nunca ha hecho anteriormente. Esto ayudará a encontrar cualquier tipo de infiltración al sistema y a detectar intrusos. También, por otro lado, se recomienda deshabilitar componentes que no estén en uso, esto con el fin de que no puedan ser utilizados para vulnerar el sistema.

Pregunta 4 (10 pts)

La Observabilidad es una gran herramienta que nos permite tener una visión en el tiempo de la forma en la cual se comportan sistemas computacionales, estos sistemas hacen uso extensivo de bases de datos de series de tiempo, una de las más utilizadas es Prometheus, pero existen soluciones que utilizan otras bases de datos o motores de búsqueda como Elasticsearch u OpenSearch. Como

ingeniera o ingeniero a cargo de los sistemas de Observabilidad de una empresa, se le ha solicitado dar respuesta a las siguientes preguntas, con el fin de determinar la estrategia que seguirá la empresa en términos de Observabilidad en los siguientes años.

- **¿Por qué las bases de datos de series de tiempo son tan utilizadas en soluciones de Observabilidad? Realice un análisis desde el punto de vista de la naturaleza de los datos que se recolectan. (2 pts)**
  Las bases de datos timeseries están enfocadas en almacenar información relacionada a una marca de tiempo, esta información se ordena secuencialmente en función de estas marcas de tiempo y suele ser muy útil para análisis de información en tiempo real. Esto es especialmente útil para la observabilidad, ya que de esta manera es fácil encontrar patrones de uso en función del tiempo, permite observar periodos de tiempo con un uso más alto, ya sea periodos de horas o temporadas de meses y por su parte, los periodos de uso más bajos también. Además, debido a que las bases timeseries permiten el análisis en tiempo real, hace que se puedan detectar problemas mucho más rápido y, por ende, se logren solucionar estos problemas en tiempo real y se puedan tomar decisiones en tiempo real.
- **¿Es posible utilizar BigTable como una base de datos de series de tiempo que se pueda utilizar como parte de una solución de Observabilidad? Justifique su respuesta desde el punto de vista de la naturaleza de la base de datos. (2 pts)**
  A pesar de que BigTable no está originalmente pensado para ser una base de datos timeseries, si puede utilizarse como una para una solución de observabilidad. Esto es posible debido a que las celdas de bigtable poseen la capacidad de ser indexadas a través de timestamps. Bigtable almacena las versiones mas actuales de primero, lo cual también puede brindar una forma de usar esta base como timeseries para observabilidad.
- **Suponiendo que tenemos una solución de Observabilidad que utiliza Elasticsearch, ¿Cómo podemos ahorrar dinero con información histórica? (2 pts)**

Se puede ahorrar dinero distribuyendo la información en distintos datatiers dependiendo de la antigüedad. Por lo general, la información más antigua suele utilizarse muy poco o prácticamente nada, por ende, esta puede almacenarse en un frozen tier que esta diseñado para data accesada prácticamente nunca y que no se actualiza. Este tier suele ser el más barato de todos. De ahí en adelante la información puede ir siendo distribuida en los distintos tiers, teniendo después del frozen el cold, luego el warm y luego el hot, cada uno diseñado para soportar cada vez más actividad y para información que necesita más actualización o es más accesada. De esta forma, se ahorra progresivamente más dinero moviendo la información a distintos tiers para que la información menos usada cueste cada vez menos almacenarla.

- **Comente las ventajas y las desventajas de utilizar un servicio de Observabilidad on-premise (por ejemplo, Prometheus y Grafana) vs un Managed Service (como Datadog), justifique**

  Los servicios de observabilidad on-premise tienen distintas ventajas y desventajas. Al hablar de ventajas podemos enfocarnos primero en mayor control, esto se debe a que tenemos el control total del diseño de la solución de observabilidad y su infraestructura, que puede ser bueno cuando se tienen requisitos muy específicos. Además, se tiene independencia de cualquier Cloud Provider por lo que las actualizaciones y en general cualquier momento en el que se necesite detener el sistema es total decisión propia y no se depende de nadie más.
  Por otro lado, las principales desventajas es que se requiere infraestructura propia, lo que hace que se generen costos y gasto de tiempo adicionales en la logística de la contratación de personas que se encarguen del trabajo, la compra de equipo y el manejo de recursos económicos para subsidiar el sistema. Además, hay que realizar todo el proceso de configuración del sistema y gestionar el mantenimiento de este. También, limita lo escalable que sea el sistema y hace que si quisiéramos escalarlo tengamos que invertir en hardware adicional o realizar cambios al existente.

Por todo esto considero que es preferible utilizar un Managed Service ya que provee una facilidad de implementación y gestión increíble, ahorra mucho tiempo y dinero en cuestiones de mantenimiento y actualizaciones, proveen soporte y escalabilidad fácil y es cuestión de pagar un monto que generalmente suele ser más económico que la inversión de infraestructura que debe hacerse en un servicio on-premise.