MODULO 04 - EJERCICIO 04-B

ALEXIS YURI M.

Analiza un conjunto de operaciones CRUD y determina cuáles están correctamente diseñadas para Cassandra. Identifica mejoras en rendimiento según claves de partición y uso de índices.

1: Análisis de la consulta y la tabla

Se elige el Caso 1 que es una operación de inserción (INSERT) que registra que un usuario vio un episodio. La tabla se llama reproducciones y tiene una clave primaria compuesta PRIMARY KEY ((id\_usuario), timestamp).

La consulta es la siguiente:

INSERT INTO reproducciones (

id\_usuario, id\_episodio, timestamp, dispositivo

)

VALUES (

'u001', 'e321', '2025-07-05 19:30:00', 'Smart TV'

);

Y el esquema de la tabla es el siguiente:

CREATE TABLE reproducciones (

id\_usuario UUID,

id\_episodio UUID,

timestamp TIMESTAMP,

dispositivo TEXT,

PRIMARY KEY ((id\_usuario), timestamp)

);

2. Identificación de la eficiencia y posibles.

El esquema de la tabla utiliza una clave de partición correcta (id\_usuario). Esto es fundamental en Cassandra porque asegura que todas las reproducciones de un mismo usuario se almacenen en una sola partición de datos.

La columna timestamp es la clave de agrupación (clustering key), lo que permite que las reproducciones dentro de una partición se ordenen cronológicamente.

La operación de inserción (INSERT) es altamente eficiente en Cassandra. Las inserciones en Cassandra son escrituras rápidas que se escriben primero en un log de commit y en la memoria (memtable) del nodo, sin una lectura previa, lo que las hace ideales para operaciones de alta concurrencia.

En resumen, la consulta de inserción es eficiente y está correctamente diseñada para Cassandra. No presenta problemas de rendimiento significativos por las siguientes razones:

- Alineación con la clave de partición: La inserción proporciona el id\_usuario, que es la clave de partición. Esto permite a Cassandra saber exactamente en qué nodo o conjunto de nodos debe escribir los datos, lo que evita operaciones de dispersión y reduce la latencia.

- Naturaleza de las escrituras en Cassandra: Cassandra está optimizada para escrituras masivas y rápidas. El modelo de datos, donde las inserciones son adiciones de datos sin necesidad de una búsqueda previa (a diferencia de las bases de datos relacionales), es inherentemente eficiente para este tipo de operación.

3. Propuestas de mejora.

Dado que la consulta de inserción ya es eficiente, las mejoras no se centrarían en la sintaxis de la consulta en sí, sino en la correcta modelación de datos para futuras consultas o la optimización de escrituras masivas.

- Optimización para lecturas: Si bien la consulta de inserción es eficiente, el modelo actual solo es óptimo para obtener el historial de un usuario. Para responder a preguntas como "¿cuántas veces se ha reproducido el episodio 'e321'?" o "¿cuáles son los episodios más vistos?", se necesitarían tablas materializadas (denormalizadas).

- Optimización para escrituras masivas: Para un gran volumen de inserciones, se podrían usar herramientas de carga masiva como COPY o un cliente de lenguaje de programación que envíe inserciones asíncronas para maximizar el rendimiento.

4. Conclusiones.

La operación de inserción es un claro ejemplo de una operación correctamente diseñada para Cassandra. El uso de la clave de partición (id\_usuario) en la inserción es la razón principal de su eficiencia, ya que dirige la escritura a la partición correcta sin una búsqueda costosa. El modelo de datos propuesto en el ejercicio, con la clave de agrupación (timestamp), también permite lecturas rápidas de series de tiempo para un usuario específico, lo cual es ideal para un historial de reproducción. Para casos de uso de agregación de datos (lecturas que no sean por usuario), se deberían crear tablas adicionales optimizadas para esas consultas.