MODULO 04 - EJERCICIO 04-A

ALEXIS YURI M.

Diseñe un esquema básico de clúster Cassandra para una app de streaming global. Incluye keyspaces, nodos, estrategia de replicación y tipos de consultas frecuentes.

1. Definición del número de centros de datos y nodos por región.

Para una aplicación de streaming global, la arquitectura del clúster de Cassandra debe ser distribuida geográficamente para reducir la latencia y aumentar la disponibilidad.

En este ejemplo se definirán dos centros de datos (regiones). Un ejemplo sería:

Centro de datos 1 (DC1): América del Norte (en California, EEUU).

Centro de datos 2 (DC2): América del Sur (en Santiago, Chile).

Para asegurar la tolerancia a fallos, cada centro de datos debería tener un mínimo de 3 nodos. Así, el clúster total constará de al menos 6 nodos, distribuidos 3 en América del Norte y 3 en América del Sur.

2. Estrategia de replicación.

La estrategia de replicación sugerida para este escenario es NetworkTopologyStrategy. Esta estrategia es fundamental para clústeres que se extienden por múltiples centros de datos, ya que permite especificar cuántas réplicas se deben mantener en cada uno de ellos. Esto es clave para garantizar la alta disponibilidad y la baja latencia para los usuarios de cada región.

Se establecerá un factor de replicación de 3 en cada centro de datos. Esto significa que cada pieza de datos tendrá 3 copias dentro de su respectivo centro de datos, lo que permite que el sistema siga funcionando incluso si dos nodos en esa región fallan.

3. Propuesta de un Keyspace y de una tabla de registro de visualizaciones.

Se propone un keyspace llamado *streaming\_analytics* y una tabla para registrar las visualizaciones.

- Keyspace:

CREATE KEYSPACE streaming\_analytics

WITH replication = {'class': 'NetworkTopologyStrategy', 'DC1' : 3, 'DC2' : 3};

- Tabla user\_views:

Esta tabla está optimizada para la consulta principal de obtener el historial de visualizaciones de un usuario.

CREATE TABLE streaming\_analytics.user\_views (

user\_id text,

episode\_id text,

view\_timestamp timestamp,

view\_duration int,

PRIMARY KEY (user\_id, view\_timestamp)

) WITH CLUSTERING ORDER BY (view\_timestamp DESC);

La clave primaria está compuesta por "user\_id" y "view\_timestamp".

user\_id es la clave de partición, lo que asegura que todas las visualizaciones de un mismo usuario se almacenen en el mismo nodo o en el mismo conjunto de nodos. Esto optimiza las consultas de historial de visualización.

view\_timestamp es la clave de agrupación, y el comando WITH CLUSTERING ORDER BY (view\_timestamp DESC) asegura que los resultados de una consulta para un usuario se devuelvan ordenados por fecha de visualización, de la más reciente a la más antigua, de manera predeterminada.

4. Creación de consultas optimizadas en CQL.

A continuación se muestran 2 consultas optimizadas para este modelo.

4.1.Consulta para obtener el historial de visualización de un usuario:

SELECT episode\_id, view\_timestamp

FROM streaming\_analytics.user\_views

WHERE user\_id = 'usuario\_ejemplo'

LIMIT 20;

Esta consulta es extremadamente rápida porque utiliza la clave de partición (user\_id) para ir directamente al nodo que contiene los datos del usuario. Como los datos ya están agrupados y ordenados por fecha, la consulta puede devolver rápidamente las 20 visualizaciones más recientes sin tener que escanear una gran cantidad de datos.

4.2. Consulta para obtener la duración de visualización de un episodio específico para un usuario:

SELECT view\_duration

FROM streaming\_analytics.user\_views

WHERE user\_id = 'usuario\_ejemplo' AND episode\_id = 'episodio\_001';

Aunque episode\_id no es parte de la clave de agrupación, esta consulta aún es eficiente. La base de datos puede ir directamente a la partición del usuario y luego escanear de manera eficiente la pequeña cantidad de filas dentro de esa partición para encontrar el episode\_id deseado.

5. Resumen.

El diseño de este clúster de Cassandra está optimizado para las cargas de trabajo de una aplicación de streaming global. Sus principales características son las siguientes:

- Escalabilidad horizontal: El modelo fragmentado de Cassandra y el uso de NetworkTopologyStrategy garantizan que el sistema pueda crecer indefinidamente al añadir más nodos y centros de datos, sin comprometer el rendimiento.

- Baja latencia: Al replicar los datos en múltiples regiones, los usuarios de cada parte del mundo pueden acceder a sus datos localmente, reduciendo la latencia y mejorando la experiencia de usuario.

- Consultas optimizadas: El diseño de la tabla user\_views con una clave de partición basada en el usuario y una clave de agrupación basada en el tiempo de visualización permite que las consultas más frecuentes y críticas sean extremadamente rápidas.