Memoria

Álvaro Felipe Pérez

Miguel Gómez Prieto

Memoria de la Base de Datos Cassandra para el Videojuego

Este documento describe el diseño, implementación y consideraciones realizadas para la migración del modelo relacional a una base de datos NoSQL (Cassandra) destinada a la gestión de estadísticas y registros en un videojuego.

Quick Start

1. **Iniciar el Clúster:**
    - Levantar la base de datos con 3 nodos:

        ```
        docker compose up -d
        ```

2. **Preparar Archivos:**
    - Copiar el archivo *creacion.cql* y la carpeta *resultados* en la carpeta *tmp/cassandra1*:

        ```
        cp creacion.cql tmp/cassandra1/
        ```

        ```
        cp -r Query_sql/resultados tmp/cassandra1/
        ```

3. **Cargar los Datos en Cassandra:**
    - Ingresar al contenedor:

        ```
        docker exec -it cassandra1 cqlsh
        ```

    - Ejecutar en cqlsh:

        ```
        SOURCE '/var/lib/cassandra/creacion.cql';
        ```

4. **Ejecución de Consultas:**
    - Las queries se pueden ejecutar desde cqlsh o utilizando funciones en Python.

Diseño de la Base de Datos

Uso del Email como Identificador

- Hemos considerado usar el campo Email como equivalente al UserID en la base de datos relacional.

Tabla Hall_of_fame

- Se ha utilizado como clave de partición los valores de País y mazmorra_id para almacenar de forma conjunta las estadísticas sobre cada mazmorra.
- En la tabla Hall_of_fame se ha empleado el Email como clave de clustering key para diferenciar casos en los que dos usuarios del mismo país completen la misma mazmorra con el mismo tiempo.
- El nombre de la mazmorra es un campo estático, porque cada partición compuesta por una mazmorra id corresponde siempre al mismo nombre de mazmorra. Además, los tiempos se ordenan de forma ascendente para posteriormente seleccionar los 5 mejores tiempos, por eso la columna del tiempo forma parte de la clave de clustering.

Tabla Statistic

- En la tabla Statistics se ha utilizado como clave de partición los valores de Email y mazmorra_id para almacenar de forma conjunta las estadísticas de cada jugador sobre cada mazmorra.
- Se ha utilizado tiempo como clustering key para ordenar de menor a mayor los tiempos de finalización de cada mazmorra junto con la fecha para evitar que si un jugador completa la misma mazmorra dos veces con el mismo tiempo, se pierda información sobre sus estadísticas.

Tabla Top_horde

- En la tabla top_horde se ha utilizado como clave de partición evento_id junto con el País, ya que el mismo evento se repite en distintos países.
- Se utiliza como clave de clustering N_killed (número de enemigos derrotados) en orden descendente, priorizando a los jugadores con mayor número de enemigos derrotados.
  - Hemos considerado no implementar N_killed como clave de clustering para facilitar las actualizaciones de los datos, pero como la prioridad es la velocidad en lectura, que estén ordenados los datos acelera mucho la velocidad de lectura, ralentizando un poco las escrituras al no poder modificar un dato que compone la clave primaria. Para actualizar el valor, se debe realizar un delete/insert, operación más costosa en escritura al tener que borrar e insertar los enemigos asesinados cuando se complete una

horda. Con el objetivo de disminuir las probabilidades de que se lea entre el borrado y la inserción, se realizan ambas operaciones juntas en un batch.

Tabla Usuarios

- Se ha creado la tabla Usuarios para facilitar las escrituras, ya que algunas solicitudes de inserción requieren datos adicionales como el nombre de usuario, que no siempre se proporcionan en la consulta.

Hemos considerado usar el campo Email como equivalente al UserID en la base de datos relacional.

En la tabla Hall_of_fame se ha utilizado Email como clave de clustering key para diferenciar casos en los que dos usuarios del mismo país completen la misma mazmorra con el mismo tiempo. El nombre de la mazmorra es un campo estático, porque cada partición compuesta por una mazmorra id corresponde siempre al mismo nombre de mazmorra. Además, los tiempos se ordenan de forma ascendente para posteriormente seleccionar los 5 mejores tiempos.

En la tabla Statistics se ha utilizado como clave de partición los valores de Email y mazmorra_id para almacenar de forma conjunta las estadísticas de cada jugador sobre cada mazmorra. Se ha utilizado tiempo como clustering key para ordenar de menor a mayor los tiempos de finalización de cada mazmorra junto con la fecha para evitar que si un jugador completa la misma mazmorra dos veces con el mismo tiempo, se pierda información sobre sus estadísticas.

En la tabla top_horde se ha utilizado como clave de partición evento_id junto con el País, ya que el mismo evento se repite en distintos países. Se utiliza como clave de clustering N_killed (número de enemigos derrotados) en orden descendente, priorizando a los jugadores con mayor número de enemigos derrotados. Hemos considerado no implementar N_killed como clave de clustering para facilitar las actualizaciones de los datos, pero como la prioridad es la velocidad en lectura, que estén ordenados los datos acelera mucho la velocidad de lectura, ralentizando un poco las escrituras al no poder modificar un dato que compone la clave primaria.

Para actualizar el valor, se debe realizar un delete/insert, operación más costosa en escritura al tener que borrar e insertar los enemigos asesinados cuando se complete una horda. Con el objetivo de disminuir las probabilidades de que se lea entre el borrado y la inserción, se realizan ambas operaciones juntas en un batch.

Se ha creado la tabla Usuarios para facilitar las escrituras, ya que algunas solicitudes de inserción requieren datos adicionales como el nombre de usuario, que no siempre se proporcionan en la consulta.

Creación del servicio

Creación de la base de datos

- En el archivo *creacion.cql* se implementa el diseño de las 3 tablas diseñadas dentro del keyspace videojuego en cql.
- Los datos se han importado desde archivos CSV generados a partir del modelo relacional. El archivo *query_sql.ipynb* crea las tablas csv desde el modelo relacional y posteriormente se copian los datos en la base de datos Cassandra. Además, se han eliminado las filas con tiempos iguales a 0 para solventar errores ya que en la tabla top_horde hemos guardado únicamente 5 usuarios por cada mazmorra de cada país.

Creación del cluster

- Para levantar el cluster local de 3 nodos hemos usado un Docker-compose. Nuestra base de datos solo tiene un rack y un centro de datos por lo que no es estrictamente necesario usar NetworkTopologyStrategy, de hecho, SimpleStrategy es suficiente, ya que está diseñado para clústeres con un solo centro de datos según lo que solicita el enunciado, pero como teóricamente estamos implementado una NoSQL para una gran empresa de videojuegos, consideramos que tendrán más de un centro de datos, por eso usamos NetworkTopologyStrategy.

Querys

Hemos implementado diversas querys de lectura y escritura.

Lecturas

- Las querys *hall of fame* y *user statistics* requieren de mayor consistencia, sin priorizar la velocidad, por tanto, en ellas hemos cambiado el nivel de consistencia a All.
- Para la query *top horde*, no se da tanta importancia a la consistencia y se prioriza la velocidad de lectura, por tanto, hemos usado un nivel de

consistencia de Local_One que puede suponer algún error en los datos, pero maximiza la rapidez de las consultas.

Escrituras

- Sobre las escrituras hemos usado un nivel de consistencia Any para que la escritura se considere exitosa tan pronto un nodo confirme la operación. Al completar una horda se activa una función que borra y reescribe el numero de monstruos eliminados en la horda.

- Para simular de manera más realista como un videojuego haría las peticiones a la base de datos mediante triggers o llamadas a funciones se han implementado las querys sobre los cuadernillos *escritura.ipynb* y *lectura.iypnb*. Estos cuadernos jupyter funcionan una vez la base de datos ha sido creada.