

Simulación de diseño y operación de un clúster Cassandra

Contexto: 🧑‍🤝‍🧑

Comprender cómo se estructura Cassandra y cómo se define un keyspace es clave para operar correctamente en entornos distribuidos. Esta actividad busca aplicar conceptos como replicación, escalabilidad y organización de datos.

Consigna: 📌

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

Tiempo: 30 minutos

Paso a paso:

1. Definan el número de centros de datos y nodos por región (mínimo 2 regiones).
2. Seleccionen la estrategia de replicación adecuada ([NetworkTopologyStrategy](#)).
3. Propongan un keyspace y una tabla que registre visualizaciones de contenido (usuario, episodio, timestamp).
4. Escriban al menos dos consultas en CQL optimizadas para ese modelo.
5. Expongan brevemente su diseño y justificación al grupo.

Ejercicio 1: Simulación de diseño y operación de un clúster Cassandra

1. Estructura de centros de datos y nodos

- Región 1: Sudamérica (SCL-DC) – 3 nodos
- Región 2: Norteamérica (US-DC) – 3 nodos

2. Estrategia de replicación

Se utiliza 'NetworkTopologyStrategy' para definir el número de réplicas por data center.

- SCL-DC: 2 réplicas
- US-DC: 2 réplicas
- US-DC: 2 réplicas

3. Keyspace y tabla

```
CREATE KEYSPACE streaming_app
WITH REPLICATION = {
  'class': 'NetworkTopologyStrategy',
  'SCL-DC': 2,
  'US-DC': 2
};
```

```
USE streaming_app;
CREATE TABLE visualizaciones (
  user_id UUID,
  episodio_id UUID,
```

```
timestamp TIMESTAMP,  
dispositivo TEXT,  
duracion_segundos INT,  
PRIMARY KEY ((user_id), timestamp)  
);
```

user_id como clave de partición permite agrupar las visualizaciones por usuario. *timestamp* como clustering key permite ordenar por fecha.

4. Consultas CQL optimizadas

-- a) Visualizaciones de un usuario

```
SELECT * FROM visualizaciones  
WHERE user_id = aaaa-bbbb-cccc-dddd;
```

-- b) Visualizaciones de un usuario en un día

```
SELECT * FROM visualizaciones  
WHERE user_id = aaaa-bbbb-cccc-dddd  
AND timestamp >= '2025-07-01' AND timestamp < '2025-07-02';
```

-- c) Agregación del lado del cliente

Se obtienen los registros y se suman en la aplicación

Análisis de rendimiento de operaciones CRUD en Cassandra

Contexto: 🧑🏫

Las operaciones CRUD en Cassandra tienen particularidades por su arquitectura basada en columnas y sistema distribuido. Esta actividad ayuda a explorar sus efectos en el rendimiento.

Consigna: 📝

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.

Tiempo: ⌚ 30 minutos

Paso a paso: 🛠️

1. El docente compartirá 4 consultas: una de inserción, una de lectura, una de actualización y una de borrado.
2. En parejas, identifiquen si usan claves de partición adecuadas, si son eficientes y si afectan negativamente al rendimiento.
3. Propone mejoras para cada caso (reorganización de tabla, uso o eliminación de índices, ajuste en clave primaria, etc.).
4. Comparte tus conclusiones con otra dupla.

Ejercicio 2: Análisis de rendimiento de operaciones CRUD en Cassandra

1. Análisis de consultas

| Operación | Consulta | Evaluación |
|---------------|--|--|
| Inserción | INSERT INTO tabla (...) VALUES (...) | Eficiente, Cassandra optimiza escritura |
| Lectura | SELECT * FROM tabla WHERE campo_secundario = 'x' | Ineficiente si no se usa la clave de partición |
| Actualización | UPDATE tabla SET campo = valor WHERE id = ... | Eficiente si <i>id</i> es clave primaria |
| Borrado | DELETE FROM tabla WHERE campo_secundario = 'x' | Ineficiente si no usa la clave de partición |

2. Mejoras sugeridas

- Lectura y borrado ineficientes: rediseñar clave primaria para incluir campo_secundario o agregar índice secundario si es lectura frecuente.
- Reorganización de tabla: considerar desnormalización para adaptarse a consultas frecuentes.
- Ajuste de clave primaria: crear nuevas tablas si las consultas requieren otro enfoque (modelo orientado a consultas).
- Evitar múltiples índices: solo usarlos cuando la cardinalidad lo justifique.