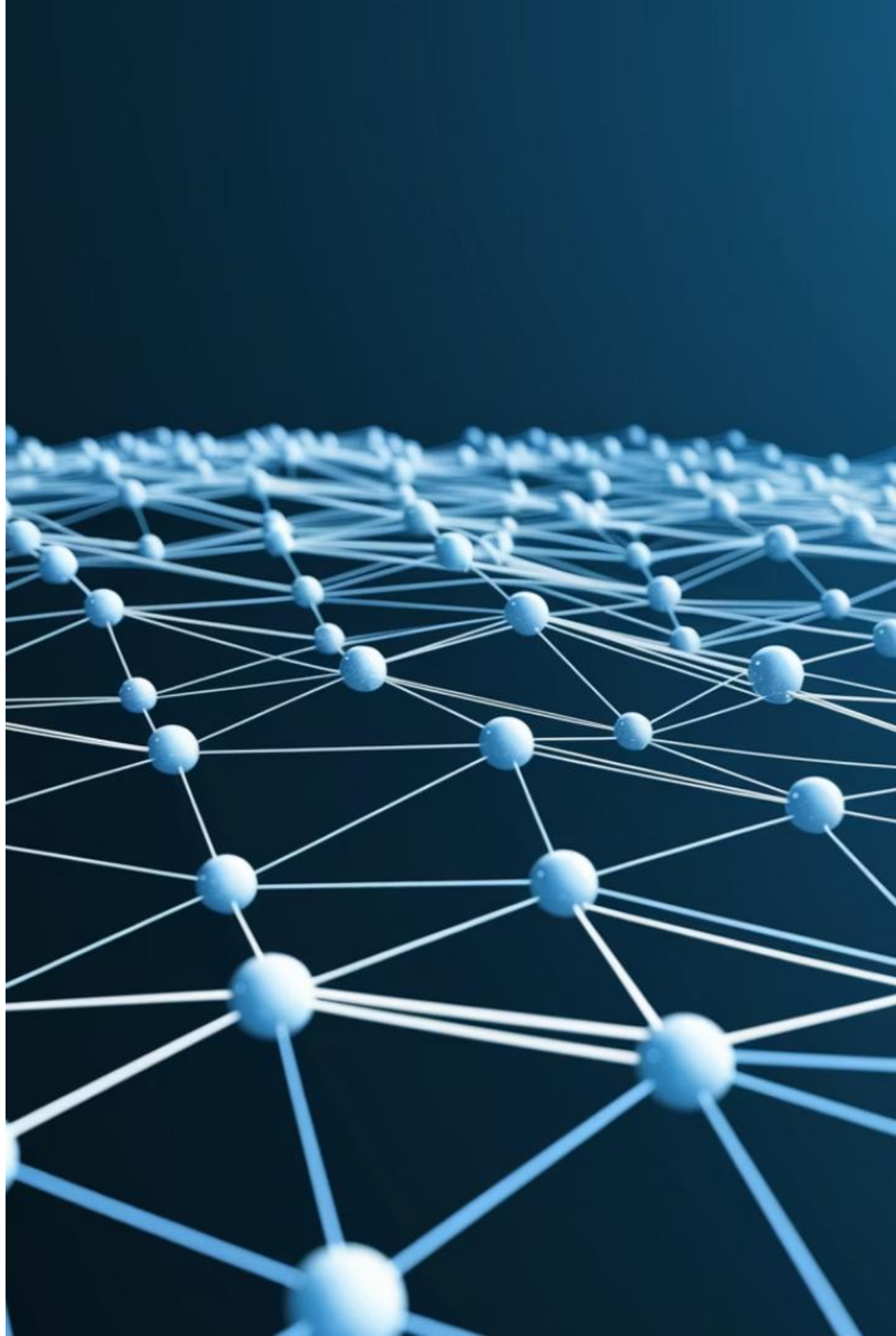
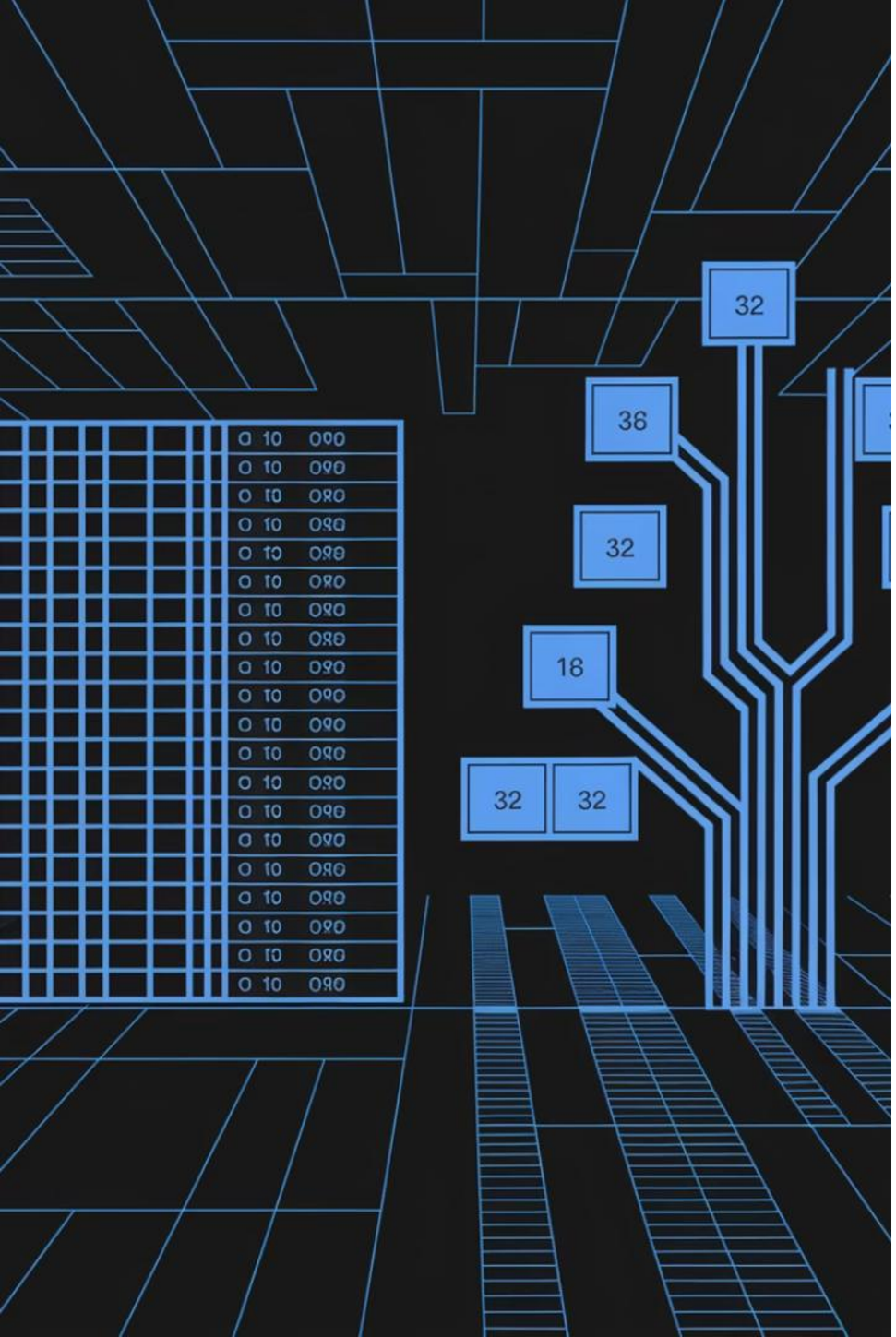
Base de Datos Apache Cassandra

Apache Cassandra es una base de datos NoSQL distribuida y orientada a columnas, creada originalmente por Facebook para manejar grandes volúmenes de datos con alta disponibilidad y tolerancia a fallos. Actualmente es utilizada por empresas como Netflix, Instagram y Apple.

A diferencia de las bases de datos relacionales tradicionales, Cassandra está diseñada específicamente para trabajar en entornos altamente escalables y con datos distribuidos geográficamente, ofreciendo una arquitectura sin puntos únicos de fallo.

por Kibernum Capacitación S.A.





Preguntas de Activación

¿Cuál es la principal diferencia entre una base de datos relacional y una base de datos NoSQL?

¿Conoces algún sistema donde se necesiten altas tasas de escritura y escalabilidad horizontal?

¿Qué dificultades crees que podrían surgir al migrar desde un modelo relacional a uno NoSQL orientado a columnas?

Características Principales

Distribuida

No hay un nodo maestro. Todos los nodos son iguales (arquitectura peer-to-peer), lo que elimina puntos únicos de fallo.

Escalabilidad horizontal

Puedes agregar nodos al clúster sin reiniciar ni afectar el rendimiento, permitiendo un crecimiento orgánico.

Alta disponibilidad

Si un nodo falla, los otros siguen funcionando sin interrupciones, garantizando la continuidad del servicio.

Modelo orientado a columnas

Los datos se organizan por column families, no por filas, optimizando el almacenamiento y acceso.

Ventajas de Cassandra

Tolerancia a fallos

Si un nodo cae, el clúster sigue funcionando sin interrupciones, manteniendo la disponibilidad de los datos en todo momento.

Alta escalabilidad

Se pueden agregar nodos de forma sencilla y automática, permitiendo que el sistema crezca según las necesidades sin tiempo de inactividad.

Rendimiento en escrituras

Muy optimizada para escrituras a gran velocidad, ideal para aplicaciones que generan grandes volúmenes de datos continuamente.

Configuración flexible

Ajustes de replicación, consistencia y compresión que pueden adaptarse a diferentes casos de uso y requisitos.

Desventajas a Considerar

Curva de aprendizaje

Más compleja que las bases de datos relacionales tradicionales, requiriendo un cambio de mentalidad en el diseño y modelado de datos.

No soporta JOINs ni subconsultas

Las relaciones entre tablas se modelan de otra forma, generalmente mediante la desnormalización y duplicación estratégica de datos.

Redundancia de datos

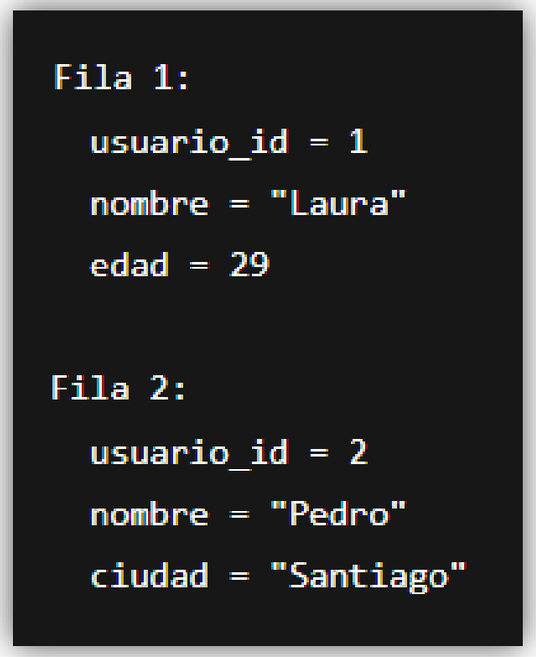
En muchos casos se duplican datos para optimizar la lectura, lo que puede aumentar el espacio de almacenamiento necesario.

Base de Datos Orientada a Columnas

Concepto Fundamental

En Cassandra, los datos se organizan en familias de columnas, en lugar de filas tradicionales. Cada fila no tiene que tener las mismas columnas, lo que proporciona gran flexibilidad.

Este modelo es especialmente útil para grandes volúmenes de datos analíticos, como sensores, logs, sistemas de monitoreo y otras aplicaciones que generan datos a gran escala.



Cada fila puede tener columnas distintas. Esto le da **flexibilidad y eficiencia en lectura de columnas específicas**.

Arquitectura de Cassandra

Nodo

Instancia de Cassandra que almacena parte de los datos. Es la unidad básica de la arquitectura.

Particiones

Dividen los datos por clave para distribuir la carga entre los nodos del clúster.

Clúster

Conjunto de nodos que colaboran entre sí para distribuir y replicar los datos.

Data Center

Grupo lógico de nodos en una ubicación física, permitiendo la replicación geográfica.

Snitch

Informa a Cassandra sobre la topología del clúster para optimizar la comunicación.

Keyspaces en Cassandra

Definición y Uso

Un keyspace es el equivalente a una base de datos en Cassandra. Dentro de un keyspace creamos tablas (column families) que contendrán nuestros datos.

Al crear un keyspace, definimos la estrategia de replicación y el factor de replicación, que determina cuántas copias de cada dato se almacenarán en el clúster.

```
CREATE KEYSPACE academia
WITH replication = {
  'class': 'SimpleStrategy',
  'replication_factor': 3
};
```

Ejemplo de creación de un keyspace con SimpleStrategy y factor de replicación 3, lo que significa que cada dato se almacenará en tres nodos diferentes del clúster.

Usar un keyspace:

```
USE academia;
```

Operaciones en Tablas

Crear una tabla

Cassandra requiere una clave primaria que defina cómo se particiona la tabla. Esta clave determina la distribución de los datos en el clúster.

Modificar tabla

Podemos alterar la estructura de una tabla existente, añadiendo o modificando columnas según evolucionen nuestras necesidades.

Insertar datos

La sintaxis es similar a SQL, pero con consideraciones específicas sobre la clave primaria y la partición de los datos.

Consultar datos

Cassandra solo permite filtrar por clave primaria o por columnas indexadas, lo que influye directamente en el diseño de tablas.

Operaciones en Tablas

Crear una tabla:

```
CREATE TABLE estudiantes (
  rut TEXT PRIMARY KEY,
  nombre TEXT,
  correo TEXT,
  carrera TEXT
);
```

Modificar tabla:

```
ALTER TABLE estudiantes ADD edad INT;
ALTER TABLE estudiantes DROP correo;
```

CRUD en Cassandra

Insertar datos (Create)

INSERT INTO tabla (columna1, columna2) VALUES (valor1, valor2);

Consultar datos (Read)

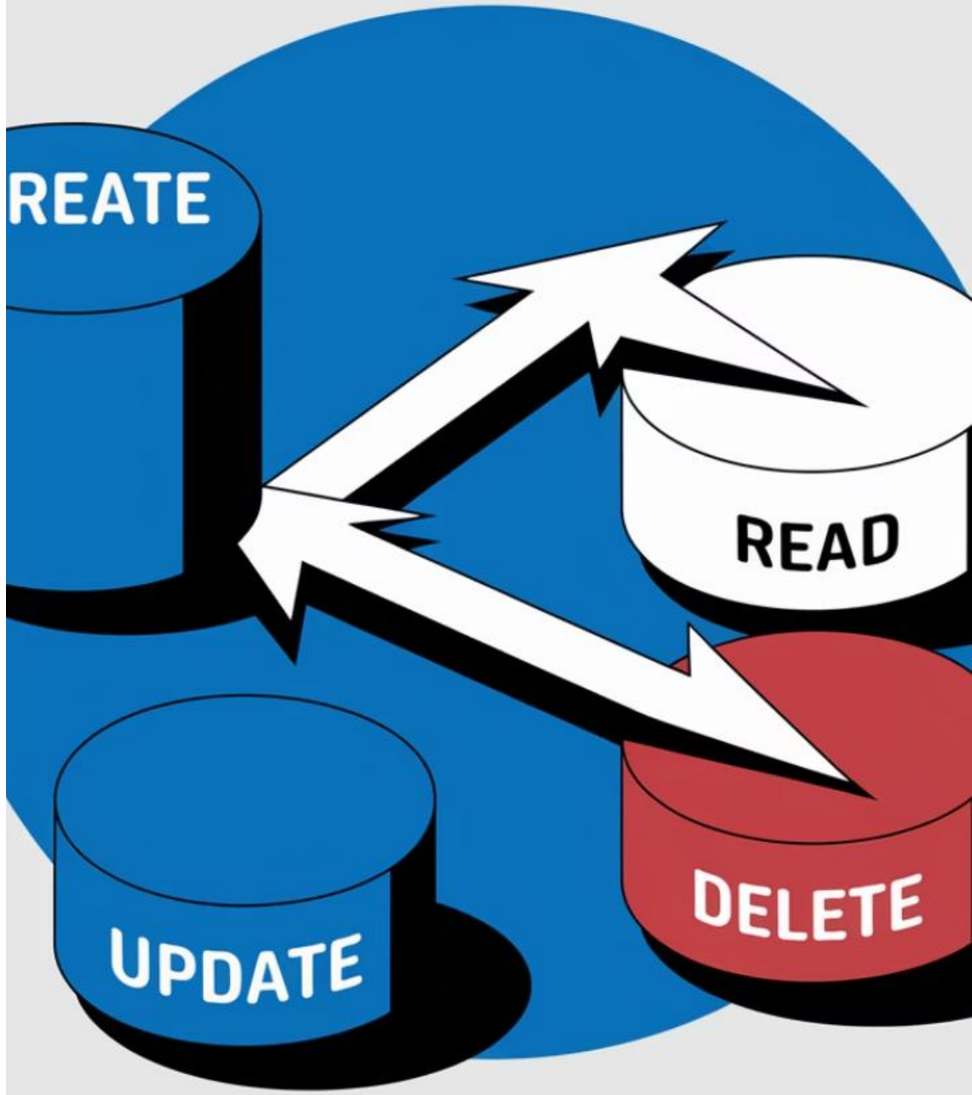
SELECT columna1, columna2 FROM tabla WHERE clave_primaria = valor;

Actualizar datos (Update)

UPDATE tabla SET columna1 = nuevo_valor WHERE clave_primaria = valor;

Eliminar datos (Delete)

DELETE FROM tabla WHERE clave_primaria = valor;



Operaciones en Tablas

Insertar Datos

```sql
INSERT INTO estudiantes (rut, nombre, carrera, edad)
VALUES ('12345678-9', 'Miguel Ramos', 'Ingeniería en Informática', 35);
```

Actualizar Datos

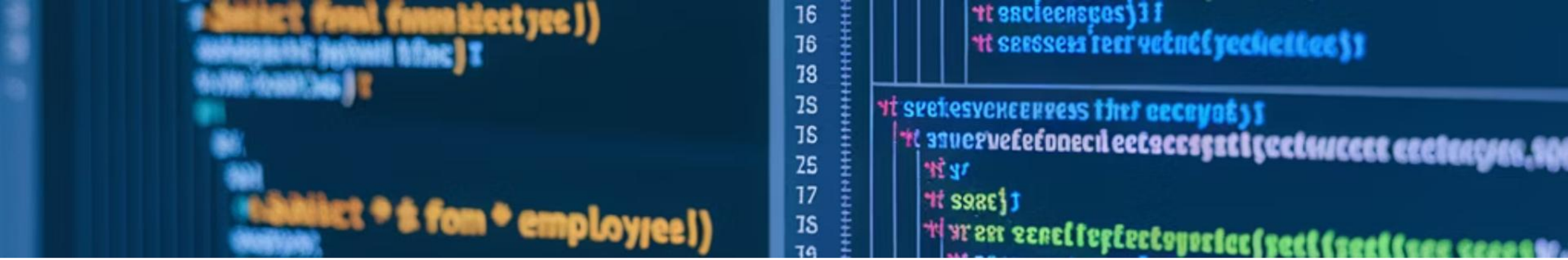
```sql
UPDATE estudiantes
SET edad = 36
WHERE rut = '12345678-9';
```

Consultar Datos

```sql
SELECT * FROM estudiantes WHERE rut = '12345678-9';
```

Eliminar Datos

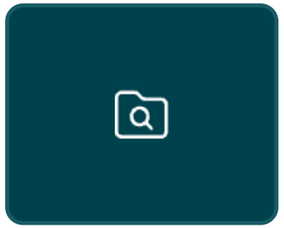
```sql
DELETE FROM estudiantes WHERE rut = '12345678-9';
```



Cassandra Query Language (CQL)

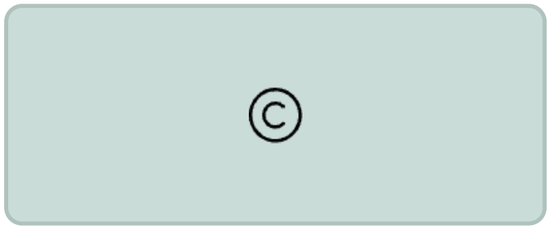
SQL clásico	CQL (Cassandra)
CREATE DATABASE	CREATE KEYSPACE
CREATE TABLE	CREATE TABLE
SELECT * FROM ...	SELECT ... FROM ...
JOIN	No soportado
GROUP BY	No soportado
WHERE con filtros	Solo con índices o claves

Recomendaciones de Modelado



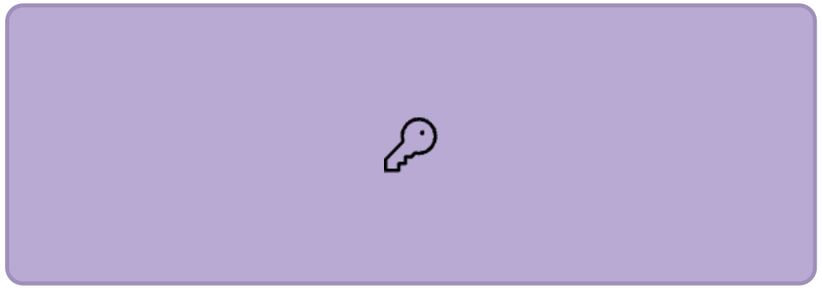
Modela según consultas

Diseña tus tablas pensando en cómo serán consultadas, no como en una base relacional



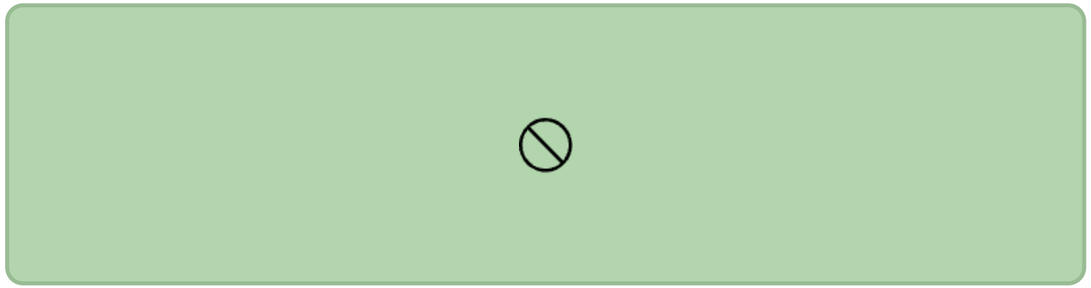
Desnormaliza cuando sea necesario

Repite datos si mejora el rendimiento (es común y aceptado)



Define claves primarias claras

Cada tabla debe tener una clave primaria bien pensada que optimice las consultas



Evita operaciones JOIN

Cassandra no soporta JOINs, adapta tu modelo de datos a esta limitación

Ejemplo Completo de Uso

Crear un keyspace para monitoreo de temperatura

CREATE KEYSPACE monitoreo_temperatura WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 3};

Definir una tabla para lecturas de sensores

CREATE TABLE lecturas (sensor_id text, fecha_hora timestamp, ciudad text, temperatura float, PRIMARY KEY (sensor_id, fecha_hora));

Insertar registros de prueba

INSERT INTO lecturas (sensor_id, fecha_hora, ciudad, temperatura) VALUES ('sensor_A1', '2023-10-15 10:30:00', 'Madrid', 22.5);

Consultar y manipular datos

SELECT * FROM lecturas WHERE sensor_id = 'sensor_A1';

Ejemplo Completo de Uso

```
CREATE KEYSPACE universidad
WITH replication = {
  'class': 'SimpleStrategy',
  'replication_factor': 2
};

USE universidad;

CREATE TABLE cursos (
  codigo TEXT PRIMARY KEY,
  nombre TEXT,
  profesor TEXT,
  semestre TEXT
);

INSERT INTO cursos (codigo, nombre, profesor, semestre)
VALUES ('INF101', 'Bases de Datos', 'Sofía', '2024-2');

SELECT * FROM cursos;

UPDATE cursos SET profesor = 'Miguel' WHERE codigo = 'INF101';

DELETE FROM cursos WHERE codigo = 'INF101';
```

Actividad Práctica Guiada: Diseño y Operaciones CRUD en Apache Cassandra

Objetivo

Aplicar los conocimientos adquiridos sobre Cassandra para crear un keyspace, diseñar una tabla orientada a columnas y realizar operaciones básicas de inserción, consulta, actualización y eliminación utilizando **CQL (Cassandra Query Language)**.

Paso a Paso Detallado

1 Diseña un keyspace para un sistema de monitoreo de temperatura

Imagina que estás desarrollando un sistema para almacenar datos de sensores de temperatura en distintas ciudades.

Tarea: Crea un keyspace llamado monitoreo_temperatura con una estrategia simple de replicación:

```
CREATE KEYSPACE monitoreo_temperatura
WITH replication = {
  'class': 'SimpleStrategy',
  'replication_factor': 1
};
```

Actividad Práctica Guiada: Diseño y Operaciones CRUD en Apache Cassandra

2 Define una tabla para registrar las lecturas de sensores

La tabla debe permitir almacenar:

•ID del sensor (sensor_id)
•Ciudad (ciudad)
•Fecha y hora (fecha_hora)
•Temperatura (temperatura)

Tarea: Crea la tabla lecturas con clave primaria compuesta (sensor_id, fecha_hora):

```
USE monitoreo_temperatura;

CREATE TABLE lecturas (
  sensor_id text,
  ciudad text,
  fecha_hora timestamp,
  temperatura float,
  PRIMARY KEY (sensor_id, fecha_hora)
);
```

Actividad Práctica Guiada: Diseño y Operaciones CRUD en Apache Cassandra

3 Inserta registros de prueba

Tarea: Inserta al menos **tres lecturas distintas** para un mismo sensor:

```
INSERT INTO lecturas (sensor_id, ciudad, fecha_hora, temperatura)
VALUES ('sensor_A1', 'Santiago', toTimestamp(now()), 24.7);

-- Repetir con al menos dos valores más para el mismo sensor
```

Actividad Práctica Guiada: Diseño y Operaciones CRUD en Apache Cassandra

4 Realiza una consulta para verificar los datos

Tarea: Ejecuta una consulta para visualizar todas las lecturas del sensor 'sensor_A1':

```
SELECT * FROM lecturas WHERE sensor_id = 'sensor_A1';
```

Actividad Práctica Guiada: Diseño y Operaciones CRUD en Apache Cassandra

5 Actualiza un valor de temperatura

Tarea: Actualiza la temperatura de una de las lecturas (usa una fecha exacta ya insertada):

```
UPDATE lecturas
SET temperatura = 25.3
WHERE sensor_id = 'sensor_A1' AND fecha_hora = '2024-04-09 10:00:00';
```

Actividad Práctica Guiada: Diseño y Operaciones CRUD en Apache Cassandra

6 Elimina una lectura específica

Tarea: Borra uno de los registros:

```
DELETE FROM lecturas
WHERE sensor_id = 'sensor_A1' AND fecha_hora = '2024-04-09 10:00:00';
```

Resultado Esperado

- Creación exitosa de un keyspace y tabla en Cassandra.
- Realización correcta de operaciones CRUD básicas.
- Familiarización con la sintaxis de CQL y la lógica de claves primarias compuestas.

Recursos Complementarios

📺 **Video explicativo:**
https://www.youtube.com/watch?v=7bM5e2xa6Ic
Este video ofrece una introducción clara a Cassandra, su arquitectura distribuida, sus ventajas y cómo se diferencia de otras bases de datos.

📘 **Documentación oficial:**
🔗 Documentación oficial de Apache Cassandra

Preguntas de Reflexión Final

1) ¿Qué ventajas ofrece Cassandra frente a otras bases NoSQL cuando hablamos de tolerancia a fallos y disponibilidad?
2) ¿Por qué es importante pensar en las consultas antes de diseñar la estructura de tablas en Cassandra?
3) ¿Qué recomendaciones clave recordarías al modelar datos en Cassandra utilizando CQL?