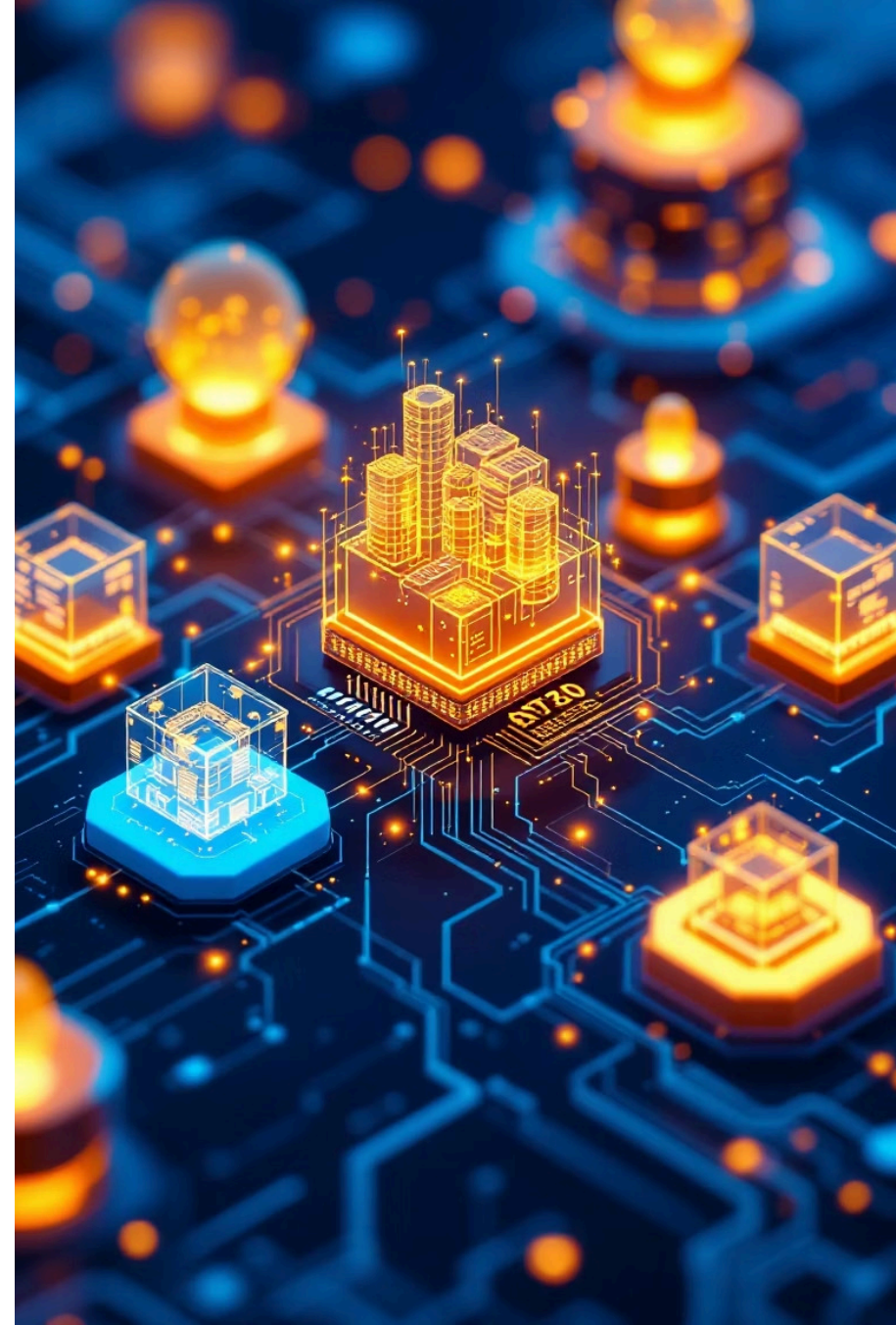


Base de datos Cassandra

 por Kibernetum Capacitación S.A.



Introducción a Cassandra

Apache Cassandra es una base de datos NoSQL distribuida y orientada a columnas. Fue creada por Facebook para manejar grandes volúmenes de datos con alta disponibilidad y tolerancia a fallos, y hoy en día es usada por empresas como Netflix, Instagram, Apple, entre muchas otras.

A diferencia de las bases de datos relacionales tradicionales, Cassandra está diseñada para trabajar en entornos altamente escalables y con datos distribuidos geográficamente.



Características de Cassandra



Distribuida

No hay un nodo maestro. Todos los nodos son iguales (arquitectura peer-to-peer).



Escalabilidad horizontal

Puedes agregar nodos al clúster sin reiniciar ni afectar el rendimiento.



Alta disponibilidad

Si un nodo falla, los otros siguen funcionando.



Modelo orientado a columnas

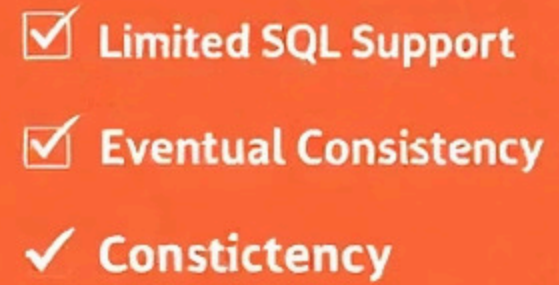
Los datos se organizan por column families, no por filas.



Consistencia ajustable

Puedes decidir si priorizar rendimiento o consistencia según el caso.

- 
- ✓ Scalability
 - ✓ Fault Tolerance
 - ✓ Complexity

- 
- ✓ Limited SQL Support
 - ✓ Eventual Consistency
 - ✓ Constistency

Ventajas y Desventajas

Ventajas

- Tolerancia a fallos: Si un nodo cae, el clúster sigue funcionando sin interrupciones
- Alta escalabilidad: Se pueden agregar nodos de forma sencilla y automática
- Rendimiento en escrituras: Muy optimizada para escrituras a gran velocidad
- Configuración flexible: Ajustes de replicación, consistencia, compresión

Desventajas

- Curva de aprendizaje: Más compleja que bases relacionales
- No soporta JOINS ni subconsultas: Las relaciones entre tablas se modelan de otra forma
- Redundancia de datos: En muchos casos se duplican datos para optimizar lectura

¿Qué es una base de datos orientada a columnas?

En Cassandra, los datos se organizan en familias de columnas, en lugar de filas tradicionales.

Cada fila no tiene que tener las mismas columnas. Este modelo es útil para grandes volúmenes de datos analíticos, por ejemplo, sensores, logs, monitoreo, etc.

Ejemplo conceptual:

Fila 1:

`usuario_id = 1`

`nombre = "Laura"`

`edad = 29`

Fila 2:

`usuario_id = 2`

`nombre = "Pedro"`

`ciudad = "Santiago"`

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

Arquitectura de Cassandra



Clúster

Conjunto de nodos que colaboran entre sí



Data Center

Grupo lógico de nodos en una ubicación física



Nodo

Instancia de Cassandra que almacena parte de los datos



Snitch

Informan sobre topología y dividen datos por clave

Cassandra usa algoritmos de particionado (como Murmur3) para distribuir los datos entre nodos.



Particiones

Dividen los datos por clave para distribuir la carga.

Keyspaces

¿Qué es un keyspace?

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

Crear un keyspace

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

- 'SimpleStrategy': Estrategia básica de replicación.
- 'replication_factor': 3: Cada dato se almacena en 3 nodos.

Usar un keyspace

```
USE academia;
```

Manipulación de Datos en Cassandra

Operaciones en tabla:



Modificar tabla

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

Cassandra **requiere una clave primaria** que defina cómo se particiona la tabla.



Modificar tabla

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


CRUD en Cassandra



Insertar datos

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



Consultar datos

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

¡Importante! Cassandra solo permite filtrar por clave primaria o por columnas indexadas.



Actualizar datos

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



Eliminar datos

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

Cassandra Query Language (CQL)

CQL es el lenguaje utilizado para interactuar con Cassandra. Es similar a SQL, pero con sus propias reglas y limitaciones

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	✓ Pero solo con índices o claves

Recomendaciones de modelado



Modela según consultas

No como en una base relacional.



Desnormaliza cuando sea necesario

Es una práctica común en Cassandra.



Repite datos si mejora el rendimiento

Es común y aceptado.



Evita operaciones JOIN

Cassandra no las soporta.



Cada tabla debe tener una clave primaria clara

Bien pensada para el acceso a datos.

Ejemplo completo

```
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';
```

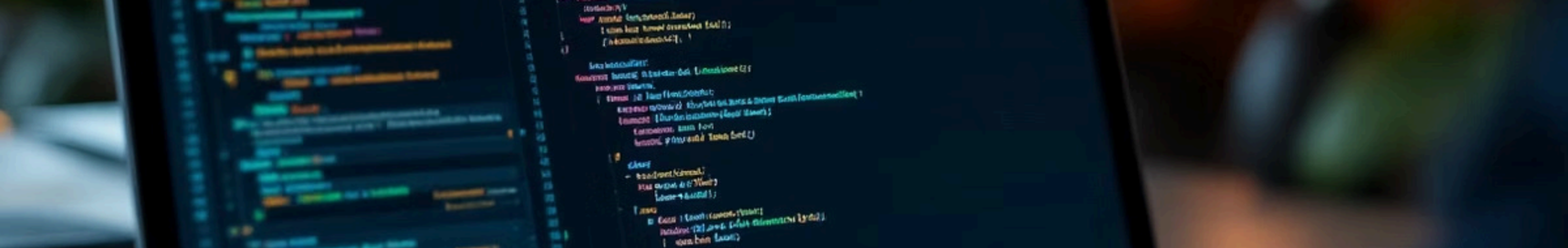
Tabla resumen

Comando	Acción
CREATE KEYSPACE	Crea un espacio lógico (como BD)
USE nombre	Selecciona un keyspace
CREATE TABLE	Crea una tabla
INSERT INTO	Inserta un registro
SELECT	Consulta (con restricciones)
UPDATE	Actualiza un valor
DELETE	Elimina un registro

Conclusión

Apache Cassandra es una solución poderosa para manejar **grandes volúmenes de datos distribuidos**, especialmente cuando necesitas escalabilidad, velocidad de escritura y tolerancia a fallos.

Su modelo columnar y su arquitectura distribuida la hacen ideal para entornos modernos, pero también implica un **cambio de paradigma respecto a bases relacionales**.



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

Seguiremos un proceso estructurado para implementar un sistema de monitoreo de temperatura utilizando Cassandra.

Pasos de la Actividad Práctica

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
};
```



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):



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
```



```
USE monitoreo_temperatura;

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



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';
```

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';
```



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