Modelado de Base de Datos NoSQL - Column-Family

PRESENTADO POR:

- DANNY CHRISTIAN LOPEZ PAJSI
- ANDERSON BRIAN FLORES SUAÑA
- HARRISON CAPIA TINTAYA

TIPOS PRINCIPALES DE BASES DE DATOS NOSQL

1. Bases de datos Key-Value (Clave-Valor)

Almacenan datos como pares clave-valor, donde la clave es única y el valor puede ser cualquier tipo de dato. No tienen estructura fija, lo que las hace muy rápidas y flexibles para accesos directos.

Usos comunes: Cachés, sesiones de usuario, configuraciones rápidas, contadores.

2. Bases de datos Documentales

Guardan datos en documentos estructurados en formatos como JSON o BSON. Los documentos pueden variar en estructura, permitiendo flexibilidad para datos semi-estructurados y consultas complejas.

Usos comunes: Aplicaciones web, gestión de contenido, catálogos de productos.

3. Bases de datos Column-Family (Columnas)

Organizan datos en filas identificadas por una clave única, con columnas agrupadas en familias que pueden variar por fila. Son ideales para datos distribuidos y escalables con alta velocidad de lectura/escritura.

Usos comunes: Análisis de logs, sistemas de recomendación, big data.

4. Bases de datos de Grafos

Modelan datos como nodos y relaciones con propiedades, facilitando la exploración de conexiones complejas entre entidades.

Usos comunes: Redes sociales, motores de recomendación, detección de fraude.

¿Qué es Column-Family?

Definición: Las bases de datos Column-Family son un tipo de base de datos NoSQL que organizan los datos en familias de columnas relacionadas, donde cada fila se identifica por una clave única y puede contener diferentes conjuntos de columnas.

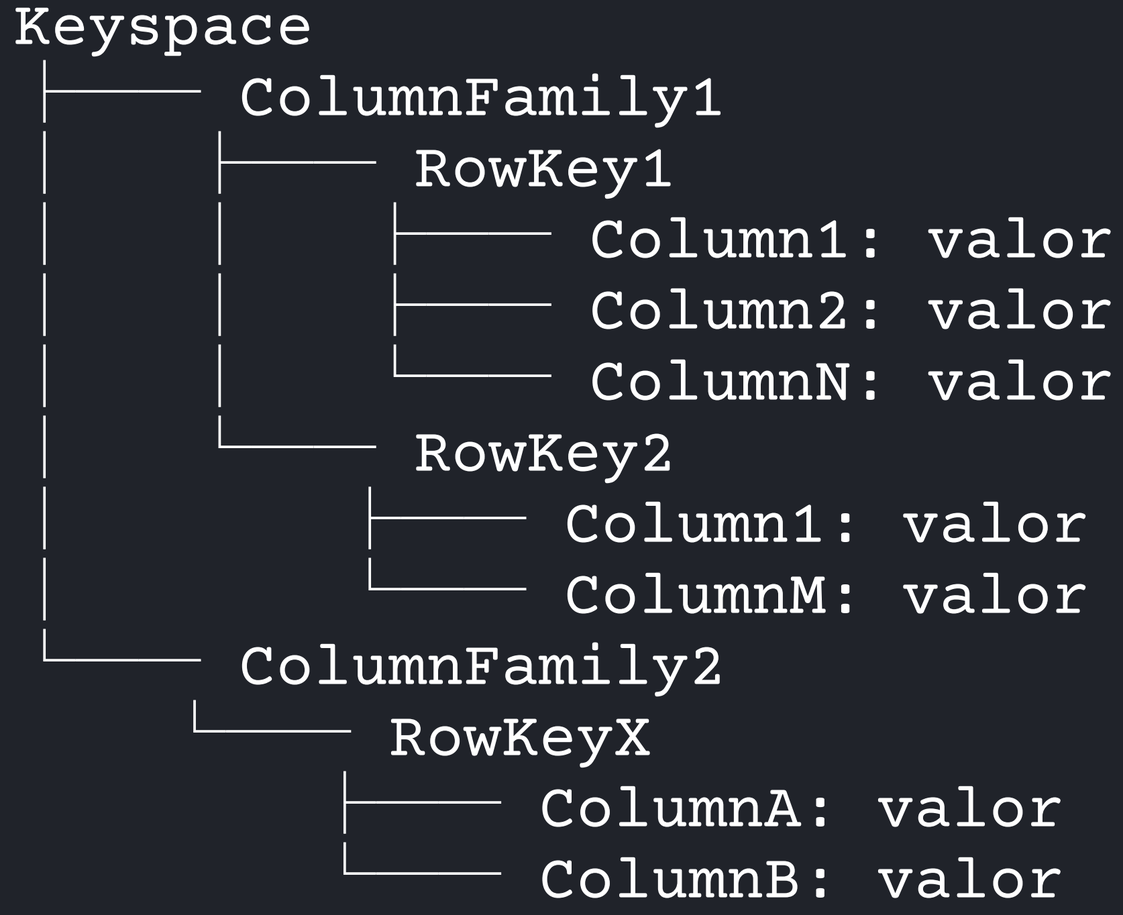
Características principales:

- **Escalabilidad masiva:** Diseñadas para manejar petabytes de datos distribuidos
- **Alto rendimiento:** Optimizadas para escrituras rápidas y lecturas por clave
- **Tolerancia a fallos:** Los datos se replican automáticamente en múltiples nodos
- **Esquema flexible:** Cada fila puede tener diferentes columnas dentro de la misma familia
- **Versionado:** Mantiene múltiples versiones de los datos con timestamps

Estructura del Modelo

Explicación de componentes:

- **Keyspace:** Es como una base de datos tradicional, contiene múltiples familias de columnas
- **Column Family:** Similar a una tabla, pero con esquema flexible. Define el tipo de datos que almacena
- **Row Key:** Identificador único de cada fila, similar a una clave primaria. Determina en qué nodo se almacenan los datos
- **Column:** Par nombre-valor con timestamp. Cada columna puede existir o no en diferentes filas
- **Timestamp:** Cada columna tiene una marca de tiempo que permite el versionado y resolución de conflictos



Diferencias clave con bases relacionales:

- No todas las filas necesitan tener las mismas columnas
- Las columnas se pueden agregar dinámicamente sin alterar el esquema
- Los datos se almacenan ordenados por Row Key para acceso eficiente

}

OPERACIONES:

1. INSERT

¿Cómo funcionan las inserciones?
En Column-Family, INSERT es una operación "upsert" - si la fila existe, actualiza las columnas; si no existe, crea una nueva fila.

Proceso interno:

- **Distribución:** El sistema calcula en qué nodo almacenar basándose en la Row Key
- **Timestamp:** Cada columna recibe automáticamente un timestamp del momento de inserción
- **Replicación:** Los datos se copian en múltiples nodos según el factor de replicación
- **Write-through:** La escritura se confirma cuando se almacena en el commit log

BASICO

```
INSERT INTO usuarios (id, nombres, correo, telefono,
ciudad)
VALUES ('009', 'Julian Navarro',
'julian@gmail.com', '900654321', 'Puno');
```

MULTIPLE

```
INSERT INTO usuarios (id, nombres, correo, telefono,
ciudad) VALUES ('010', 'Luis Perez', 'luis@gmail.com',
'982354322', 'Arequipa');
INSERT INTO usuarios (id, nombres, correo, telefono,
ciudad) VALUES ('011', 'Maria Ruiz',
'maria@gmail.com', '923654323', 'Cusco');
```

Características importantes:

- **Sin transacciones:** Cada INSERT es atómico solo a nivel de fila
- **Idempotente:** Ejecutar el mismo INSERT múltiples veces produce el mismo resultado
- **Alta velocidad:** Optimizado para miles de inserciones por segundo
- No hay restricciones de integridad: No valida claves foráneas automáticamente.

}

2.UPDATE

¿Cómo funcionan las actualizaciones?

UPDATE en Column-Family es similar a INSERT - crea nuevas versiones de las columnas con timestamps más recientes.

Proceso de actualización:

- **Versionado:** No sobrescribe datos, crea nuevas versiones con timestamp actual
- **Merge:** Al leer, el sistema devuelve la versión más reciente de cada columna
- **Compactación:** Periódicamente elimina versiones antiguas para liberar espacio
- **Parcial:** Puede actualizar solo algunas columnas sin afectaR otras.

BASICO

```
UPDATE usuarios
SET telefono = '911223344', ciudad = 'Tacna'
WHERE id = '009';
```

MULTIPLE

```
UPDATE usuarios SET ciudad = 'Ica' WHERE id =
'002';
UPDATE usuarios SET telefono = '923456789' WHERE id
= '003';
UPDATE usuarios SET nombres = 'Carlos Mendoza',
correo = 'carlosm@gmail.com' WHERE id = '004';
UPDATE usuarios SET ciudad = 'Tarapoto', telefono =
'912345678' WHERE id = '005';
```

Ventajas del versionado:

- **Recuperación de datos:** Posible acceder a valores históricos
- Resolución de conflictos: En escrituras concurrentes, gana el timestamp más reciente
- **Auditoría:** Mantiene trazabilidad de cambios
- **Rollback:** Posible recuperar estados anteriores

Limitaciones:

- Solo se puede actualizar por Row Key (clave primaria)
- No hay UPDATE masivo eficiente sin conocer las claves

}

3. DELETE

Column-Family ofrece múltiples niveles de eliminación desde columnas individuales hasta filas completas.

Tipos de DELETE:

- Column Delete: Elimina columnas específicas de una fila
- Row Delete: Elimina toda la fila y sus columnas
- Range Delete: Elimina por rangos de tiempo (útil para datos temporales)

}

COLUMN DELETE: La fila c2 sigue existiendo, pero las columnas producto y precio se eliminan.

```
DELETE producto, precio FROM compras WHERE id = 'c2';
```

ROW DELETE: Se elimina toda la fila con id = 'c3'

```
DELETE FROM compras WHERE id = 'c3';
```

RANGE DELETE: Elimina todas las filas del usuario '001' cuya fecha esté entre el 1 y el 4 de junio de 2025

```
DELETE FROM historial
WHERE usuario = '001'
AND fecha >= '2025-06-01 00:00:00'
AND fecha <= '2025-06-04 23:59:59';
```

Casos de uso de BD Column-Family:

- **Sistemas de recomendación:** Almacenan interacciones de usuario-producto para sugerencias personalizadas.
- **Análisis de logs:** Guardan grandes volúmenes de eventos para monitoreo y auditoría.
- **Aplicaciones IoT:** Recogen datos de sensores en tiempo real de forma eficiente.
- **Historial de transacciones:** Registran operaciones por usuario o cuenta de forma rápida y escalable.
- **Plataformas sociales:** Manejan publicaciones, comentarios y relaciones entre usuarios con alta velocidad.

```
Gracias {

}
```