APACHE CASSANDRA

Características principales (i)

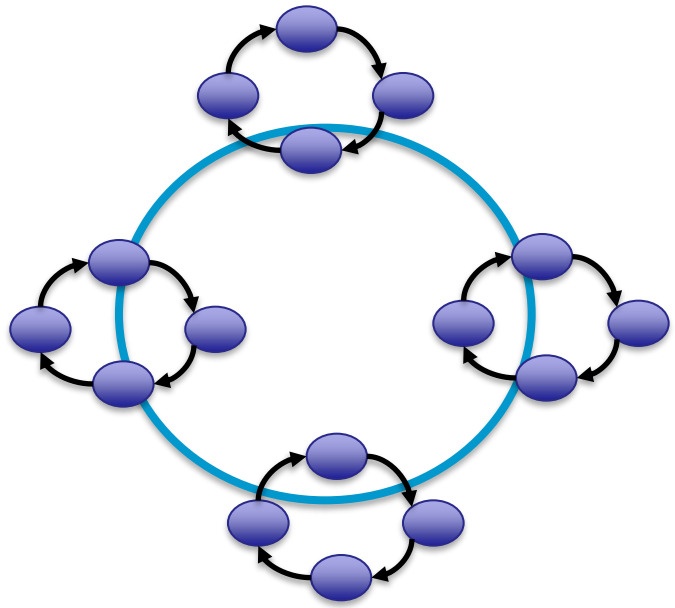
- Es una base de datos **distribuida** orientada a columnas, creada en 2008.
- Es **escalable horizontalmente**, consistente y tolerante a fallos.
- El diseño de distribución de Cassandra se basa en Dynamo de Amazon y su modelo de datos en BigTable de Google.
- Se puede decir que Facebook es el creador de Cassandra.
- Como motor de base de datos, es diferente totalmente de los sistemas de administración de bases de datos relacionales.
- El modelo de replicación es del estilo de Dynamo, y el concepto de modelo de datos "familia de columnas" es el más robusto.
- ¿Se usa? Sí. Algunas de las empresas más grandes como Facebook, Twitter, Cisco, Rackspace, eBay, Twitter, Netflix, entre otros lo emplean dentro de sus arquitecturas.

Características principales (ii)

- Escala linealmente (*si n nodos brindan x ope/seg, 2n brindan 2x ope/seg*).
- **No obedece** un patrón maestro esclavo.
- Es tolerante a fallos.
- Utiliza la replicación de los datos.
- Permite definir el nivel de consistencia
- Usa lenguaje CQL
- Permite la replicación en varios data center

- Crea un contenedor de Cassandra:

```
docker run --name lab_cassandra -d -p 9042:9042 cassandra:latest
```

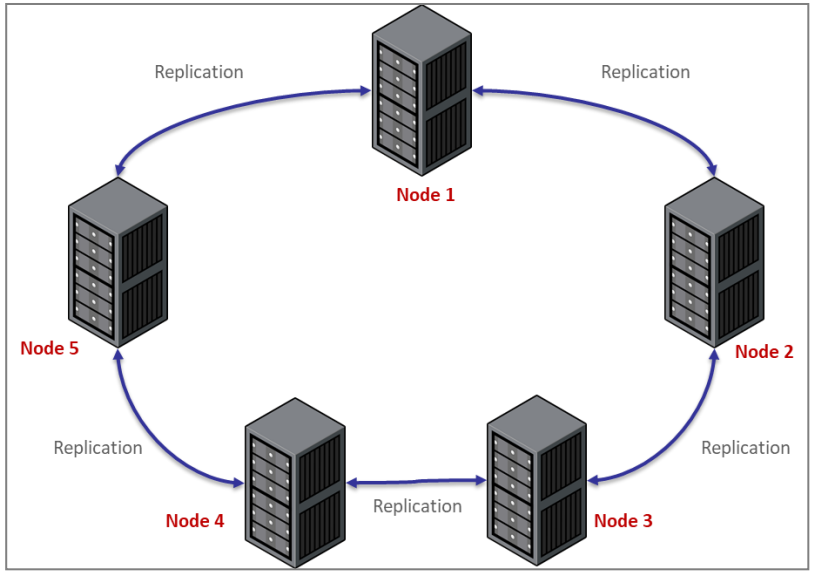


Componentes de Cassandra

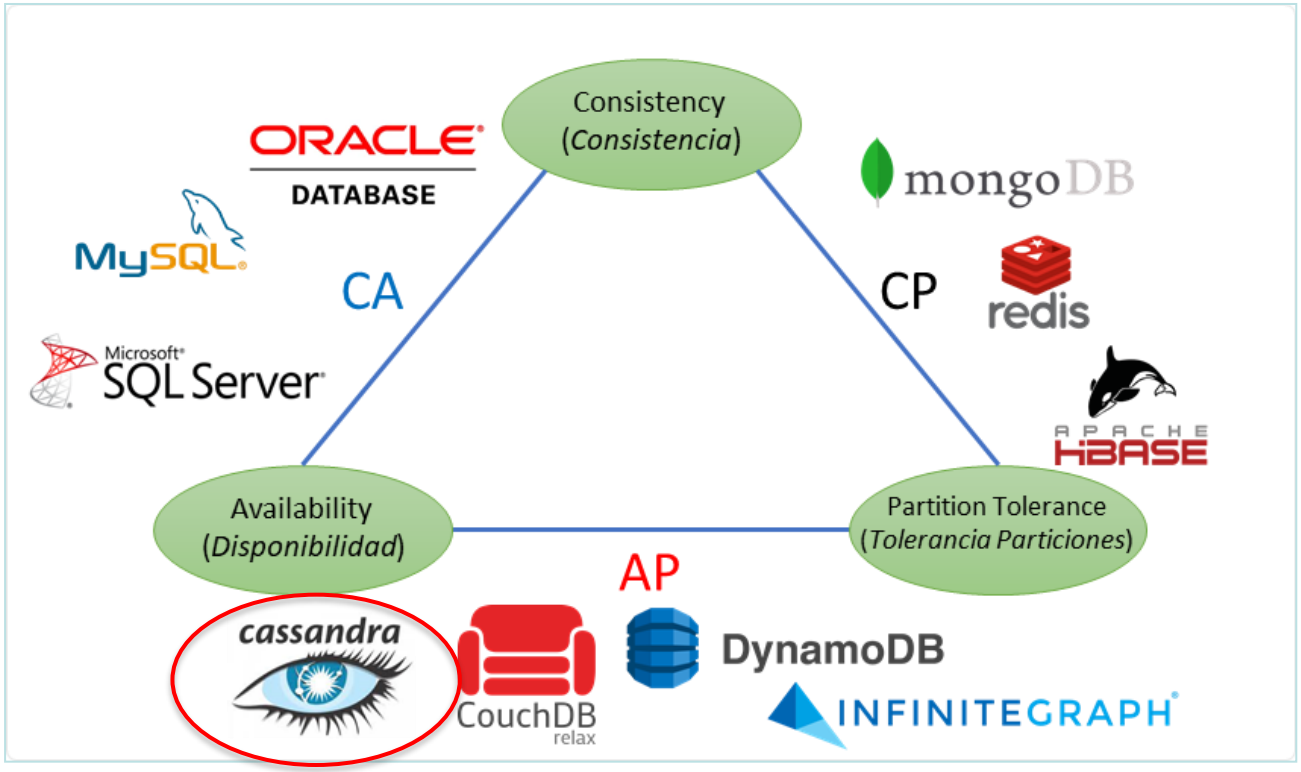
- **Nodo**: un lugar donde se almacenan los datos.
- **Centro de datos**: es una colección de nodos relacionados.
- **Clúster**: es un componente que contiene uno o más centros de datos.
- **Registro de confirmación**: mecanismo de recuperación de fallos, donde se escriben todas las operaciones de escritura.
- **Mem-table**: es una estructura de datos residente en memoria. Después del registro de confirmación, los datos pasan a la tabla en memoria. Para una familia de una sola columna existen varias mem-tables.
- **SSTable**: archivo de disco al que se vuelcan los datos de la tabla de memoria cuando su contenido alcanza un umbral.
- **Filtro Bloom**: algoritmos no deterministas muy rápidos, utilizados para probar si un elemento es miembro de un conjunto.

Conceptos generales

- Es una base de datos de tipo Clave-Valor compleja.
- Los datos se almacenan como tablas y columnas.
- Cada tabla tiene una clave principal.
- Aunque permite lecturas y escrituras muy rápidas, tiene una interfaz **SQL limitada** debido a su propia estructura.
- Los nodos de un clúster actúan como réplicas de un dato específico.
- Cassandra se encarga de devolver al usuario el valor más reciente.
- Después de atender al usuario, esta repara la lectura (en background) para actualizar los valores obsoletos.



Teorema CAP


Consistency
(Consistencia)
ORACLE
DATABASE
mongoDB
MySQL
CA
CP
redis
Microsoft
SQL Server
APACHE
HBASE
Availability
(Disponibilidad)
Partition Tolerance
(Tolerancia Particiones)
AP
cassandra
CouchDB
relax
DynamoDB
INFINITEGRAPH

Escritura

- Los registros de confirmación que residen en los nodos capturan toda actividad de escritura de los mismos nodos.
- Dichos datos se almacenan en la tabla de memoria siempre que esta esté libre.
- En el caso de que la tabla de memoria esté llena, los datos se escribirán en el archivo de datos **SStable**.
- La partición y replicación de todas las escrituras se lleva a cabo de forma automática en todo el clúster.
- La consolidación de las SSTables se realiza periódicamente, descartando los datos considerados como innecesarios.

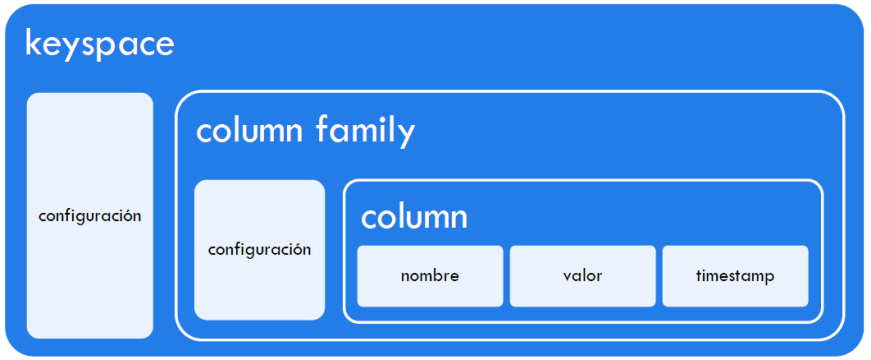
Lectura

- Cassandra consulta los valores de la tabla de memorias y establece la forma de ubicar la tabla SST que contenga los datos solicitados.
- Las operaciones de lectura se tipifican en solicitudes directas, de resumen y de reparación antes de ser delegadas a los coordinadores para que las envíen a las réplicas correspondiente.
- El primero paso es enviar una solicitud directa a una de las réplicas.
- El coordinador envía la solicitud de resumen a la cantidad de réplicas especificadas para garantizar el nivel de coherencia.
- Finalmente, comprueba que los datos devueltos están actualizados.
- Por último, el coordinador envía la solicitud de resumen a todas las réplicas restantes.
- En el caso de que algún nodo devuelva un valor desactualizado, la solicitud de reparación de lectura en segundo plano actualizará dichos valores.

Modelado de datos

Keyspace:

- Estructura más amplia del modelo de datos de Cassandra.
- Contenedor para la familia de columnas, las cuales guarda ordenadas.
- Almacena Column Family.
- Más o menos análogo al esquema de una BBDD relacional.
- Usado en Cassandra para separar aplicaciones.



Clúster:

- Conjunto de máquinas que dan soporte a Cassandra y son vistas por los clientes como una única máquina.

Modelado de datos

- Cassandra no soporta un modelo de datos relacional completo.
- Proporciona clientes con un modelo de datos simple que soportan un control dinámico sobre la disposición de los datos y el formato.
- El modelo de datos de Cassandra define cinco tipos de estructuras de datos.
- Cada columna es una tupla {nombre, valor, timestamp}.
- Los valores son todos suministrados por el cliente.
- El tipo de dato nombre y valor son matrices de byte de Java; timestamp es *long primitive*.
- Son inmutables para evitar problemas de *multithreading*.
- Se organizan dentro de las familias de columnas (Tablas).
- Se ordenan por un tipo: AsciiType, BytesType, LongType, TimeUUIDType y UTF8Type.

```
public final class ImmutablePoint {
    private final int x;
    private final int y;

    public ImmutablePoint(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }
}
```

Modelado de datos

- nombre: array de bytes que contiene el nombre o clave mediante la cual es posible indexar el dato.
- valor: array de bytes que contiene el valor.
- timestamp: indica el tiempo en el que fue almacenado el dato, Cassandra no soporta un modelo de datos relacional completo.
- column family: una abstracción explícita que representaba un contenedor de columnas dentro de un Keyspace. Con CQL (Cassandra Query Language), el término "Column Family" fue reemplazado funcionalmente por el concepto de Tablas.

- supercolum: existió este concepto, pero fue reemplazado por alternativas más eficientes y flexibles que simplifican el modelado de datos, mejoran el rendimiento y reducen la complejidad.

Modelo de datos

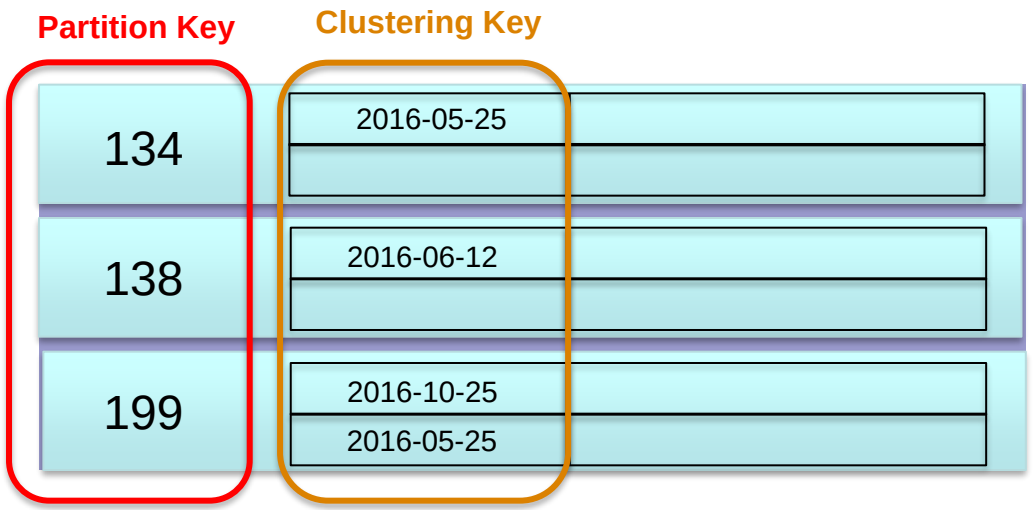
Column Name
Value
Timestamp

Se corresponde

Nombre
John
32345343249876

Keyspace — Se compone de — Columns Family

element.users_by_id




```
{
   "key": "user123",
   "columns": [
                {"column": "name", "value": John"},
                {"column": "age", "value": "30"}
    ]
}
```

```
CREATE TABLE users (
                user_id UUID PRIMARY KEY,
                name TEXT,
                age INT
); INSERT INTO users (user_id, name, age) VALUES (uuid(), 'John', 30);
```

element.users

```
SELECT  * FROM
WHERE user_id = '138'
AND timestamp >= '2016-06-12';
```

Obligatorio

unir LA UNIVERSIDAD EN INTERNET

Modelado de datos

Wide Rows:

- Se utiliza una clave de partición con múltiples filas ordenadas dentro de la misma partición para representar relaciones jerárquicas o agrupaciones de datos.
  - Permiten almacenar múltiples columnas asociadas con una clave primaria en un formato eficiente.

```
CREATE TABLE user_activities (
        user_id UUID,
        activity_date DATE,
        activity_name TEXT,
        details TEXT,
        PRIMARY KEY (user_id, activity_date)
) WITH CLUSTERING ORDER BY (activity_date DESC);
```

Set, List, Maps:

- Son tipos de datos integrados en Cassandra que permiten almacenar listas, conjuntos o mapas directamente en una columna.
- Son ideales para modelar datos donde una clave primaria necesita asociarse con un conjunto de elementos.

```
CREATE TABLE user_profiles (
        user_id UUID PRIMARY KEY,
        contact_info MAP<TEXT, TEXT>,
        tags SET<TEXT>,
        preferences LIST<TEXT>
);
```

Modelado de datos

Clustering Columns:

- Las Clustering Columns permiten organizar filas dentro de una partición según un orden específico.
- Esto reemplaza la necesidad de estructuras anidadas.
- Representan relaciones padre-hijo de manera eficiente sin la complejidad de las Super Columns.
- Específico para jerarquías o datos que requieren ordenación dentro de la misma partición.

```
CREATE TABLE orders (
        order_id UUID,
        item_id UUID,
        quantity INT,
        price DECIMAL,
        PRIMARY KEY (order_id, item_id)
) WITH CLUSTERING ORDER BY (item_id ASC);
```

unir LA UNIVERSIDAD EN INTERNET

Storage Attached Indexes (SAI):

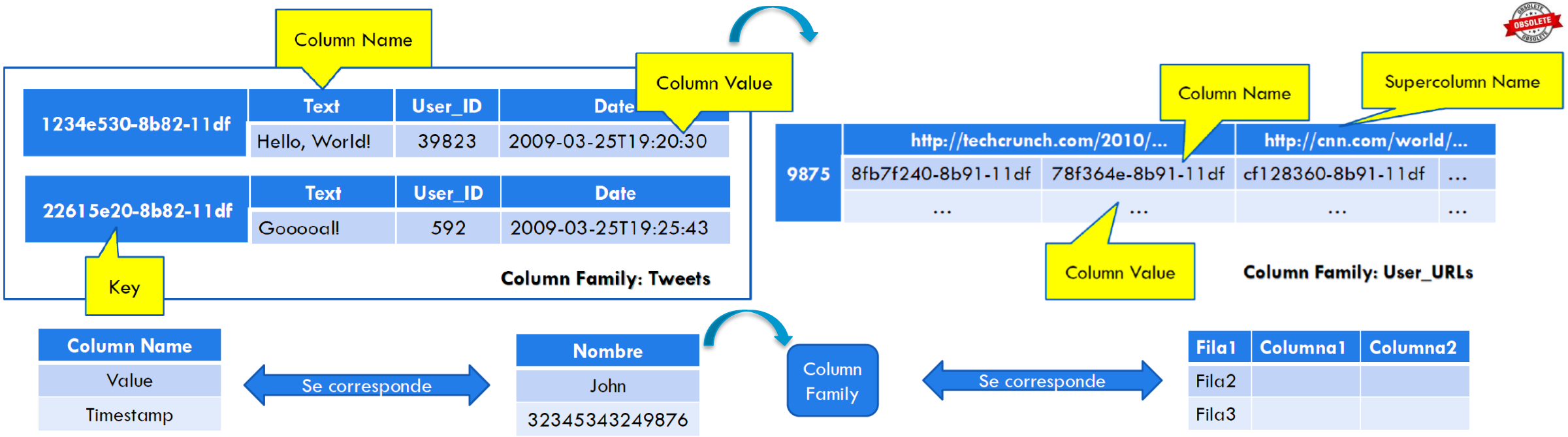
- Introducidas en Cassandra 5.0, estos índices avanzados facilitan las búsquedas eficientes en columnas que no son clave primaria.
- SAI permite realizar consultas más complejas y es una evolución frente a modelos que requerían Super Columns para búsquedas específicas.

```
CREATE CUSTOM INDEX ON user_data (email)
USING 'StorageAttachedIndex';
```

Denormalización:

- Denormalizar datos creando varias tablas con vistas específicas es una práctica común para modelar relaciones.
- Optimiza la lectura al costo de aumentar el almacenamiento, lo cual es más eficiente.
- Divide la lógica de datos en tablas planas relacionadas en lugar de jerarquías complejas.

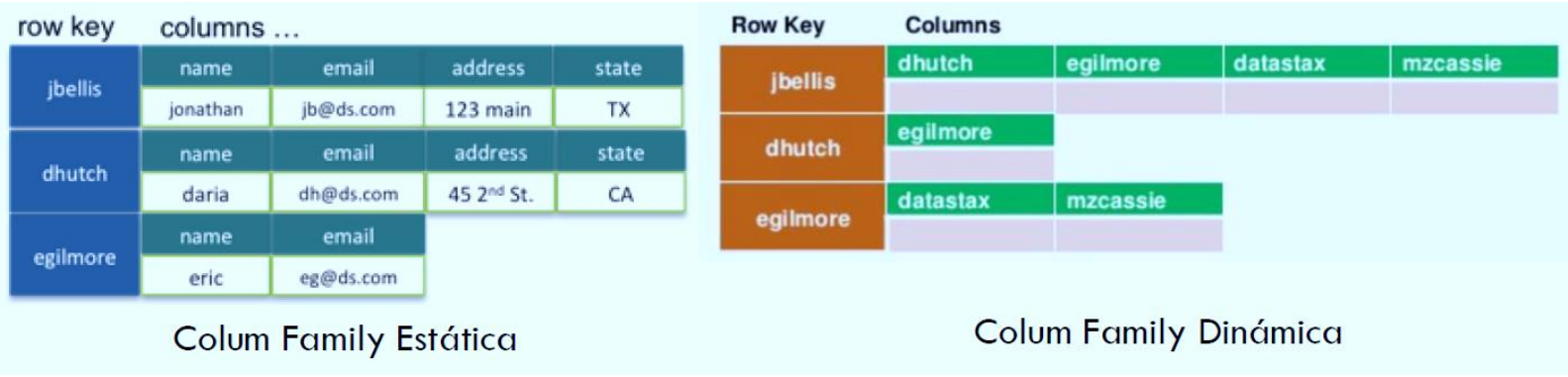
Esquema de datos



Esquema de datos

Tipos de Column Family:

- **Estática**: utiliza un conjunto estático de columnas (similar a una tabla de BBDD relacional).
- **Dinámica**: se generan las columnas según se van necesitando, evitando crear columnas que no se usarán.



Cassandra vs. SGBD

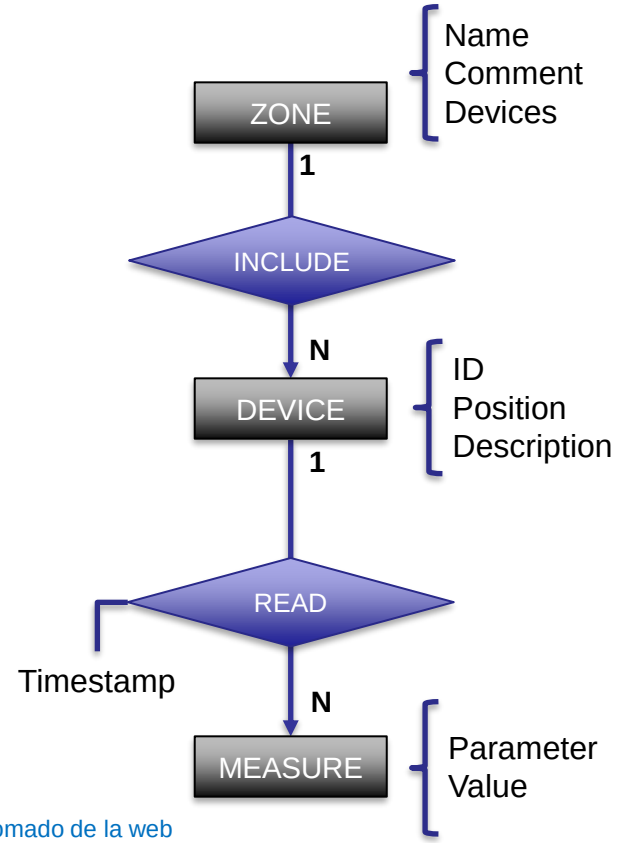
Modelado de datos

- **Conocer tus datos**.
- **Conocer previamente tus consultas**.
- Conocer el flujo de trabajo del sistema
- Tener claro aspectos de optimización
  - Tamaño de los datos
  - Número de particiones y su tamaño
  - Coste de las consultas
  - Necesidad de unir datos de cara al usuario

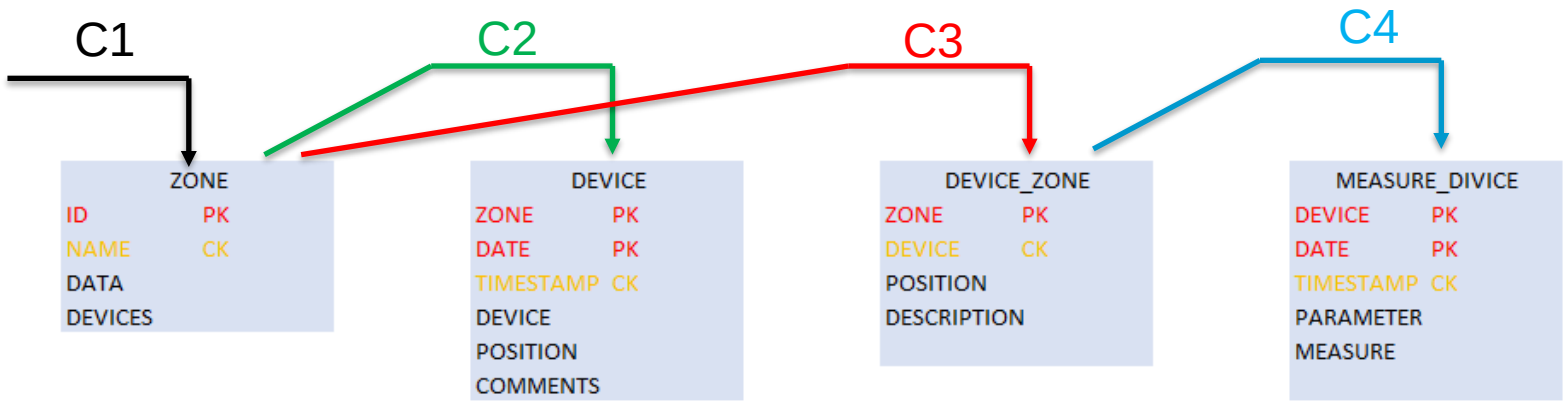
	Cassandra	SGBD
Modelo de datos	- Diseñado para consultas específicas. - El esquema es ajustado a las nuevas consultas.	- Normalizado, sin tener en cuenta las consultas. - SQL puede "devolver casi todo" gracias a JOINS.
Características	- Sin JOINS, relaciones ni claves ajenas. - Una tabla separada por consulta.	

	Datos	Media Escrituras	Media Lecturas
MySQL	>50 GB	~300 ms	~350 ms
Cassandra	>50 GB	0,12 ms	15 ms

Modelado de datos



Ejemplo tomado de la web



Cassandra Query Language CQL3

- Lenguaje de alto nivel.
- Prácticamente igual al lenguaje **SQL**.
- CQL vs SQL: no permite realizar consultas con JOINs ni subconsultas.
- Consola de línea de comandos basada en Python. `<cassandra-path>bin/cqlsh`

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

DESCRIBE KEYSPACE test;

USE test
```

Cassandra Query Language CQL3

```
CREATE TABLE datat ( ID int, deptID int, first_name varchar, last_name varchar, PRIMARY KEY
(ID, deptID) );

INSERT into datat ( ID, deptID, first_name, last_name ) VALUES (201, 28, 'ana', 'potter');
INSERT into datat ( ID, deptID, first_name, last_name ) VALUES (202, 27, 'pepe', 'potter');

SELECT * from datat;

ALTER TABLE datat ADD address text;
ALTER TABLE datat DROP address;

DROP TABLE datat;
DELETE FROM datat WHERE ID = 202;

UPDATE datat SET address = 'Calle Lopez' WHERE ID = 201 and deptID = 28;
```

Consideraciones

No existe integridad referencial.

- No existe el concepto de JOIN.

Las opciones de consulta para recuperar datos son limitadas.

La ordenación de datos es una decisión de diseño.

- No existe la instrucción GROUP BY.

No hay soporte para operaciones atómicas.

- Si la operación falla, los cambios pueden producirse.
- Primero se piensa sobre las consultas, después sobre el modelo de datos.
- Distribuida: es conveniente usarla cuando hay muchos datos distribuidos en múltiples servidores.
- Rendimiento de escritura siempre es excelente, pero el rendimiento en la lectura depende de los patrones de escritura

Muchas gracias por tu atención