UNIVERSIDAD
COMPLUTENSE
MADRID

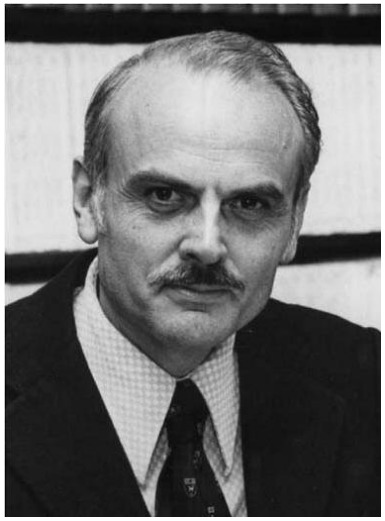
Apache Cassandra

Sistemas de Gestión de Datos y de la Información

Manuel Guerrero Moñús

Bases de datos relacionales

- Edgar Frank Codd, empleado de IBM.
- En 1970:
  - Codd publica el trabajo que siembra las bases de las BDR.
  - “Un modelo relacional de datos para grandes bancos de datos”.
    - Relaciones (tablas) y tuplas (filas).
    - Formas normales → Minimización de las redundancias.
  - Por desgracia, IBM no apoya las ideas de Codd.
- Codd siguió insistiendo y finalmente le hacen caso a regañadientes.
  - Se crea el lenguaje SEQUEL, usado en System R en 1978.
- En 1979 Larry Ellison crea la la compañía Oracle.
  - Comercializan productos basados en las ideas de Codd.
  - Optimizan SEQUEL, surgiendo SQL.



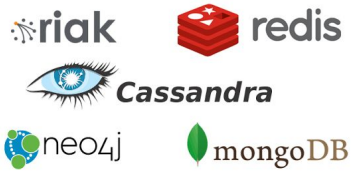
Bases de datos relacionales

- Principales ventajas:
  - Modelo estándar: Se basan en el modelo relacional y utilizan el lenguaje SQL.
  - Concurrencia: Emplean transacciones ACID para salvaguardar la consistencia de los datos.
  - Persistencia: Almacenan grandes cantidades de datos de forma segura.
  - Integración: Coordinación de aplicaciones, modelo productor consumidor.
- Principales desventajas:
  - Impedancia: Las estructuras de datos programadas y el modelo relacional no coinciden.
  - Bases de datos centralizadas: Toda la información se ubica en un único servidor.
  - Empleo de relaciones y tuplas: El esquema de datos es rígido y los valores son simples.

Bases de datos NoSQL

- En 1998 Carlo Strozzi utiliza por primera vez el concepto en 1998 para definir su propia BBDD, esta no utilizaba el lenguaje de consultas SQL, pero sí se basa en el modelo relacional.
- A medida que nos acercamos al año 2000, surge el fenómeno Big Data:
  - En 1997 Michael Cox y David Ellsworth conciben el término.
  - En 2001 Doug Laney especifica sus tres características principales:
    - Volumen → Gran cantidad de datos, petabytes o exabytes.
    - Velocidad → Se generan y se quieren procesar muchos datos en muy poco tiempo.
    - Variedad → Los esquemas de datos son cambiantes y los datos son heterogéneos.
- Posteriormente, comienzan a surgir nuevas bases de datos que pueden cubrir las necesidades del ámbito Big Data o relacionados con él (IoT), estas son las NoSQL.
  - Recuerda:
    - NoSQL no implica no usar el SQL, implica no basarse en el modelo relacional.
    - Hay quienes prefieren el término NOSQL → Not Only SQL.

Bases de datos NoSQL



- Características principales:
  - No utilizan el lenguaje de consultas SQL, pero pueden ser muy parecidos.
  - Orientadas a agregados: Manejan datos de tipo simple y otras estructuras más complejas.
  - No tienen un esquema de datos fijo, pueden añadirse libremente los datos que se deseen.
  - La mayoría de estas bases de datos están pensadas para ejecutarse sobre un clúster.
    - Los datos deben distribuirse de forma equitativa entre varios nodos (máquinas).
      - Sharding → Distintos conjuntos de datos se distribuyen entre varios nodos.
      - Replicación → Se realizan copias de los datos que se guardan en varios nodos.
    - Sus modelos de consistencia no suelen seguir las propiedades ACID, si no las BASE.
      - Básicamente Disponible – Basically Available.
      - Estado flexible – Soft state.
      - Eventualmente consistente – Eventually consistent.

Clasificación según el Teorema CAP

Las bases de datos cumplen dos de estas propiedades:

- Consistencia - Consistency: Toda lectura es respondida con la información más reciente o un error.
- Disponibilidad - Availability: No hay garantía de obtener la información más reciente.
- Tolerancia al particionamiento - Partitioning: El clúster funcionará pese a que algunos nodos fallen.

¿Que propiedades y problemas tienen nuestras bases de datos?

- Oracle, MySQL y Neo4j.
  - Propiedades: Consistencia y Disponibilidad.
  - Problema: Si falla algún nodo, la comunicación el clúster se cae.
- MongoDB, Redis.
  - Propiedades: Consistencia y Particionamiento.
  - Problema: No se puede asegurar siempre el acceso a la información.
- Cassandra, CouchDB.
  - Propiedades: Disponibilidad y Particionamiento.
  - Problema: Tal vez obtengamos datos antiguos o no actualizados.

Clasificación según el modelo de datos

Documentos

Clave-Valor

Grafos

Columna

¿Qué es Cassandra?

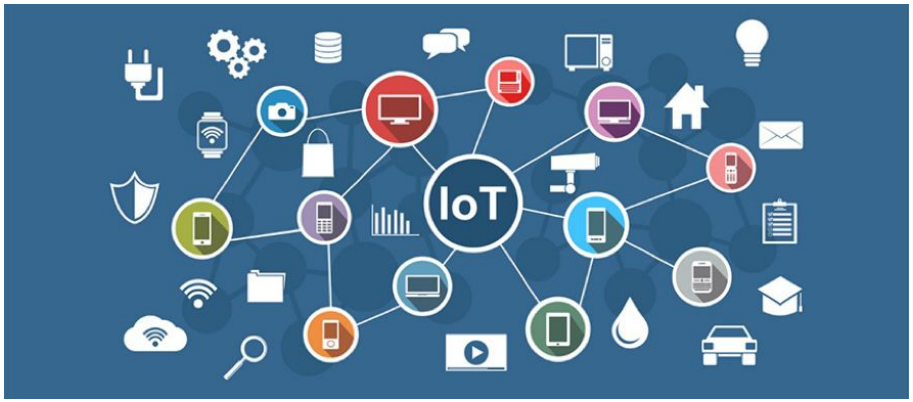
Es una base de datos orientada a columna escrita en Java que fué desarrollada por Facebook para integrarla en su motor de búsqueda para buscar mensajes en la bandeja de entrada. Su diseño se inspiró en las bases de datos NoSQL: BigTable (Google) y DynamoDB (Amazon).

Es un proyecto de la Apache Foundation desde 2009, por lo que es de código abierto.

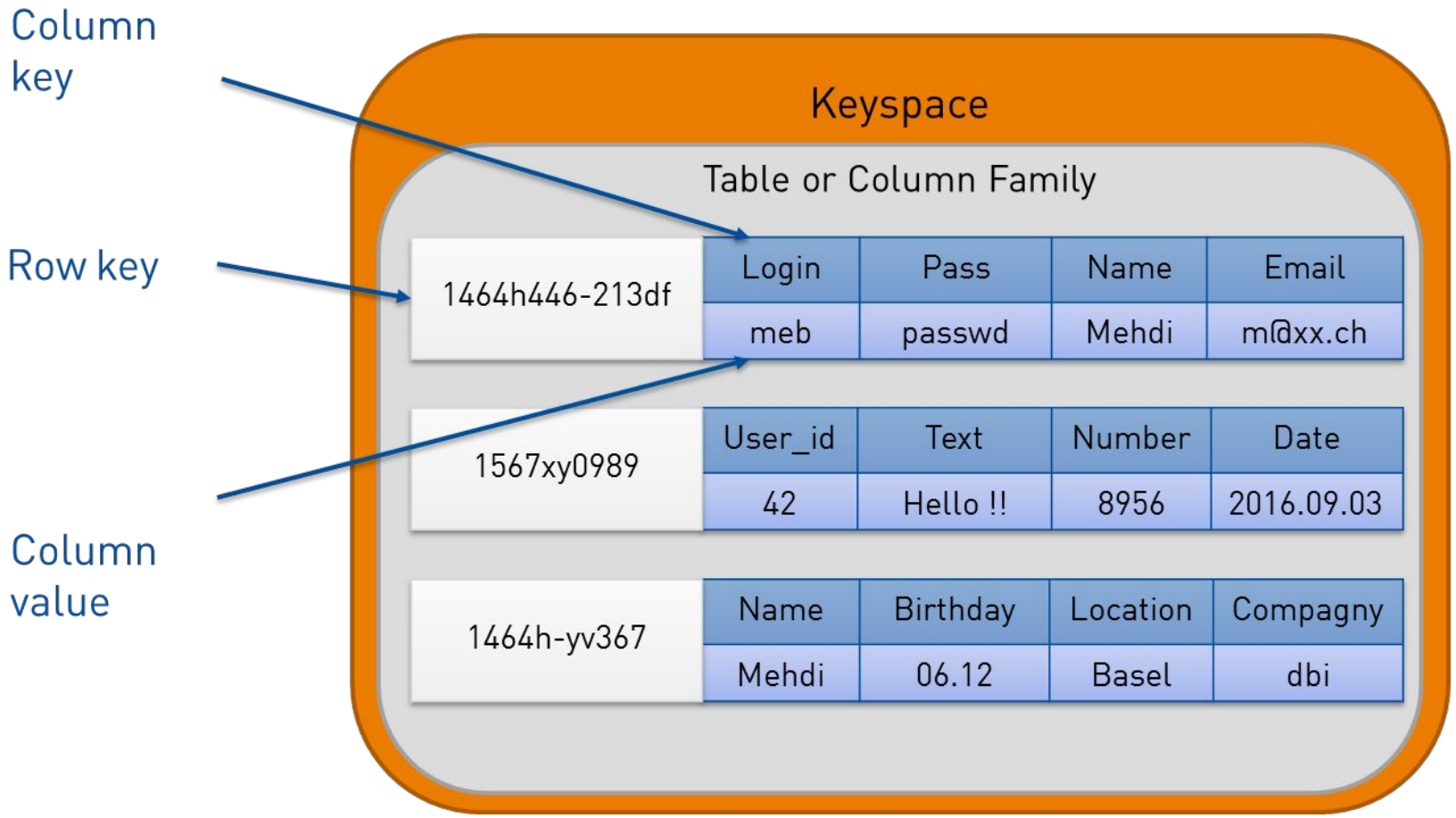
¿Qué características tiene?

- Altamente escalable.
- Muy rápida para operaciones de escritura.
- Permite la distribución y replicación de datos.
- Da soporte al modelo de cómputo MapReduce.
- Transacciones compatibles con ciertas propiedades ACID, concretamente AID.

Escenarios de uso destacados



Conceptos básicos



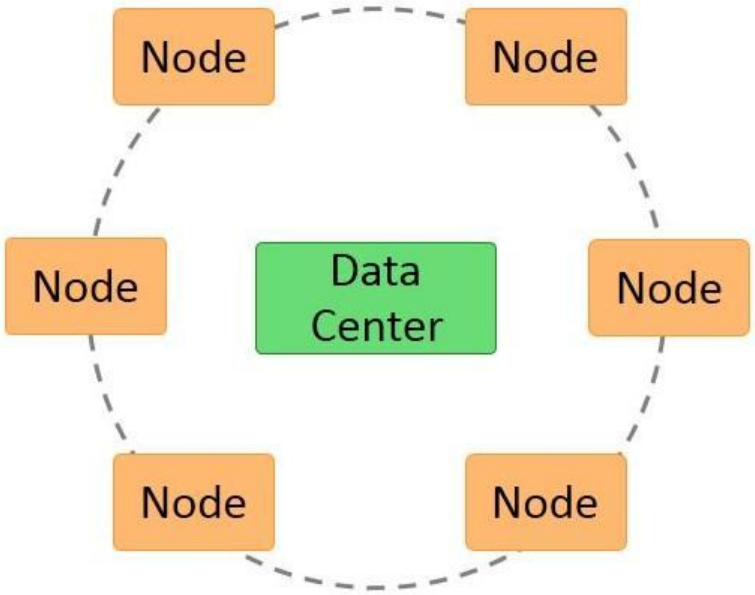
Tipos de datos

Data Type	Constants	Description
ascii	strings	Represents ASCII character string
bigint	bigint	Represents 64-bit signed long
blob	blobs	Represents arbitrary bytes
Boolean	booleans	Represents true or false
counter	integers	Represents counter column
decimal	integers, floats	Represents variable-precision decimal
double	integers	Represents 64-bit IEEE-754 floating point
float	integers, floats	Represents 32-bit IEEE-754 floating point
inet	strings	Represents an IP address, IPv4 or IPv6
int	integers	Represents 32-bit signed int
text	strings	Represents UTF8 encoded string
timestamp	integers, strings	Represents a timestamp
timeuuid	uuids	Represents type 1 UUID
uuid	uuids	Represents type 1 or type 4
varchar	strings	Represents uTF8 encoded string
varint	integers	Represents arbitrary-precision integer

Collection	Description
list	A list is a collection of one or more ordered elements.
map	A map is a collection of key-value pairs.
set	A set is a collection of one or more elements.

Componentes de la arquitectura de un clúster Cassandra (Parte I)

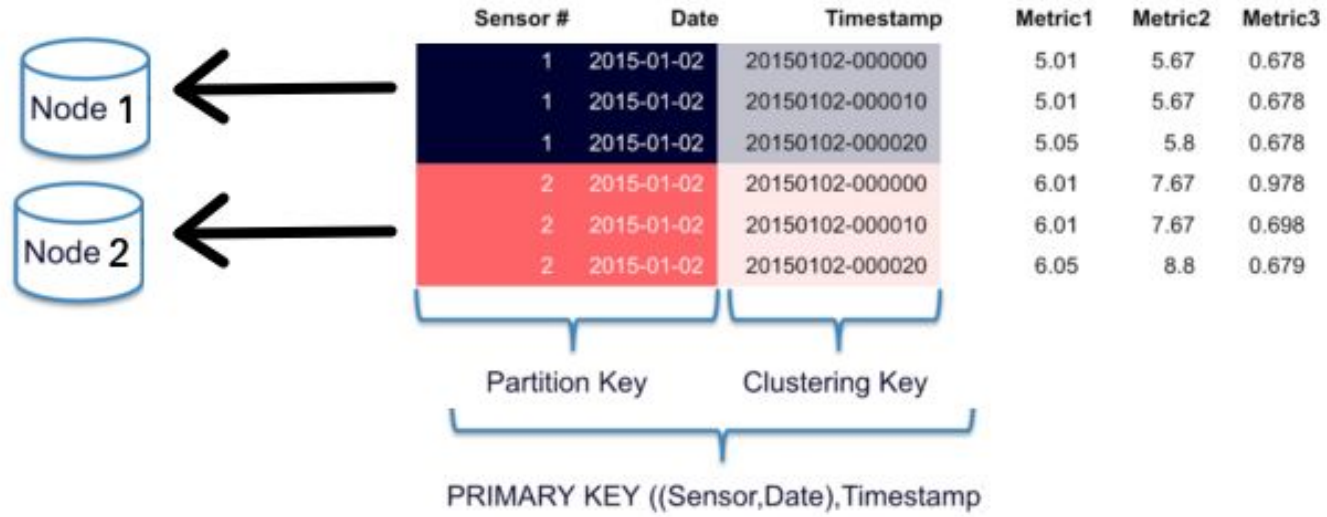
1. Nodos: Equipos donde se almacena la información.
2. Cluster: Una o más colecciones de nodos denominados Data Center.
3. Gossip: Protocolo de red empleado por los nodos para comunicarse.
4. Partitioner: Función que establece cómo se van a distribuir los datos entre los nodos del cluster.



- Cluster en forma de anillo.
- Modelo de distribución peer-to-peer.
  - No hay nodos primarios, todos los nodos son iguales.
  - Todos los nodos valen para recibir peticiones.
- Uno o más nodos del cluster pueden actuar como réplicas. Estrategias:
  - SimpleStrategy.
  - NetworkTopologyStrategy.

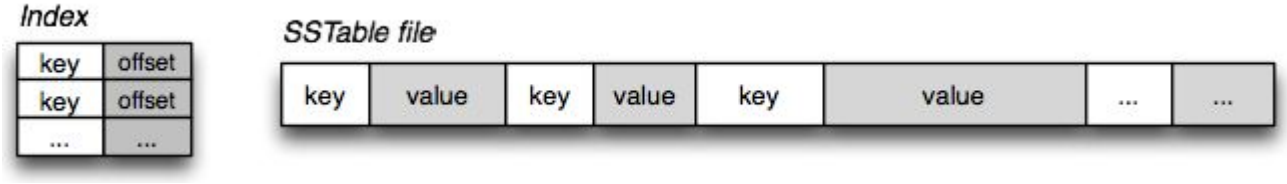
Distribución y ordenación de la información

- La distribución de la información se realiza a partir de la clave de partición.
  - Se toma el primer elemento de la clave primaria y se calcula sobre el un código hash, este indica en qué nodo del cluster se almacenan o almacenarán los datos.
  - Cada nodo gestiona un rango de claves de partición.
- ¿Interesa que la información a almacenar esté ordenada?
  - Claves de agrupación: El resto de la PK indica orden de almacenamiento en una partición.



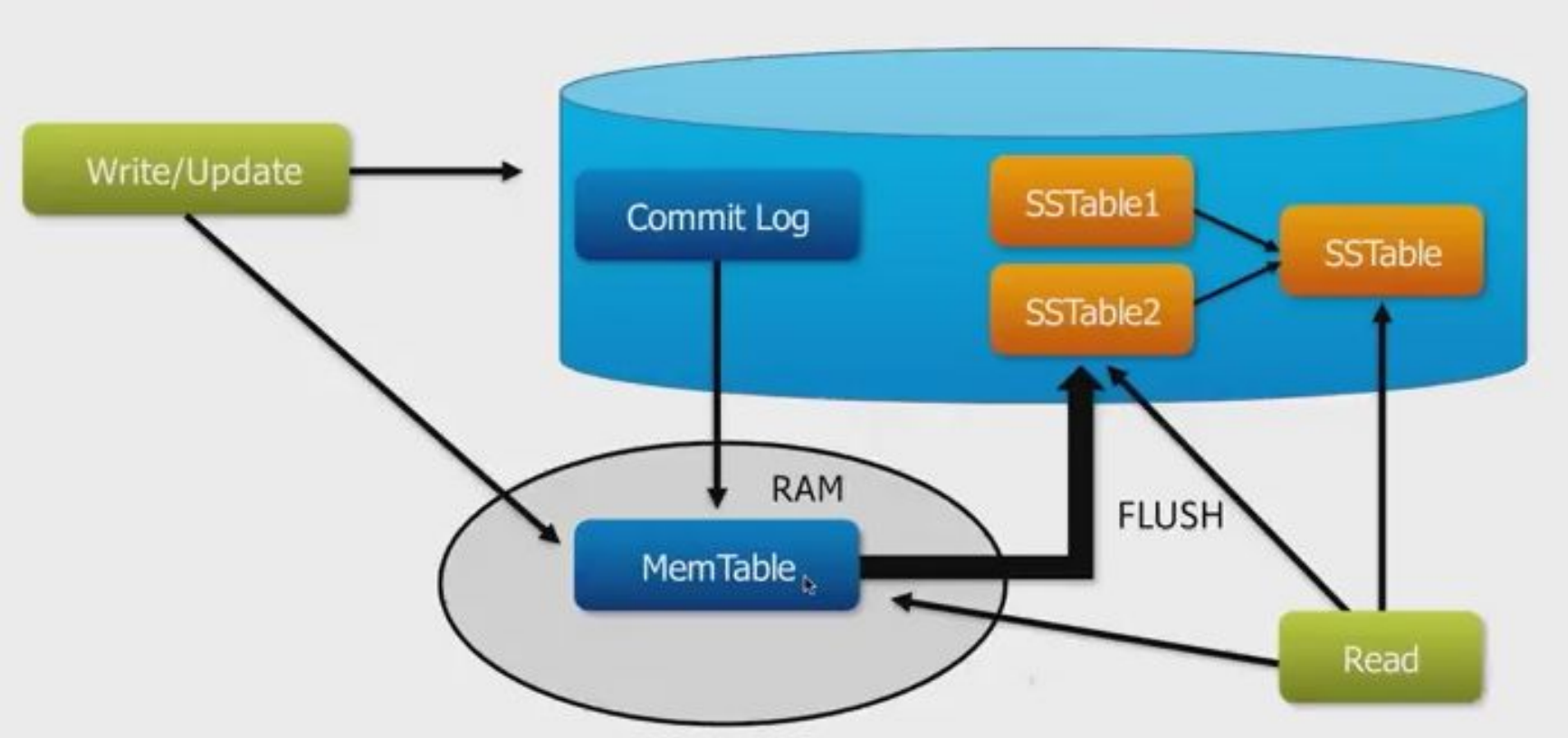
Componentes de la arquitectura de un clúster Cassandra (Parte II)

5. MemTable: Caché de datos tipo clave-columnas, muy rápida, alojada en memoria principal.
6. SSTable: Ficheros que se alojan en el disco duro de cada nodo.
    - Resultan de volcar en el disco duro una MemTable cuando crece demasiado (flush).
    - Una vez generados, son mayoritariamente inmutables.
    - Problema: Una partición puede estar almacenada a lo largo de varios SSTables.
        - Solución: Se compactan varias SSTables en una más grande, en esta nueva SSTable, los datos obsoletos se eliminan, los más recientes están ordenados e indexados.



7. Commit Log: Un fichero alojado en el disco duro de cada nodo.
    - Registra todas las operaciones de escritura o actualización, útil para recuperarse ante fallos.

Funcionamiento interno



Escritura y actualización

1. Un nodo recibe una operación de escritura.
2. Este nodo mediador envía a todos los nodos réplica la operación de escritura.
3. El nodo receptor anota en el Commit Log la operación de escritura.
4. Se envía la información a la MemTable, esta almacenará la información rápidamente.
    - Si la clave primaria ya existía, entonces los datos se reescriben.
5. Si la MemTable está demasiado llena, esta se vacía (flush) y se genera un archivo SSTable.
6. Respuesta exitosa si registra la escritura en el Commit Log y guarda los datos en el MemTable.
    - Se puede configurar el factor de escritura para indicar cuántas réplicas deben responder.

Borrado

1. Funciona como un upsert, marcando la fila de un SSTable como eliminada.
2. Durante el proceso de compactación la fila se borra físicamente del disco.
    - Si el nodo está caído, se intentará contactar con él durante un tiempo “de gracia” para realizar el borrado, en caso de superarse este tiempo, el nodo generará inconsistencias.

Lectura

- Hay varios escenarios:

1. El nodo que recibe la petición de lectura localiza un nodo con la información requerida.
    1.1. El nodo recibe los datos y luego los devuelve *
2. El nodo que recibe la petición localiza un conjunto de nodos con la información requerida.
    2.1. Recibe los datos de ellos y comprueba si existen inconsistencias en los datos.
        2.1.1. Hay inconsistencias.
            2.1.1.1. Se devuelven los datos más recientes *
        2.1.2. No hay inconsistencias.
            2.1.2.1. Se devuelven los datos *
3. Igual que 2, pero actualizando los datos inconsistentes antes de devolverlos.

* NOTA: Lectura muy rápida si los datos están en la MemTable, lenta si están en una o más SSTable.

Diseño de una base de datos Cassandra

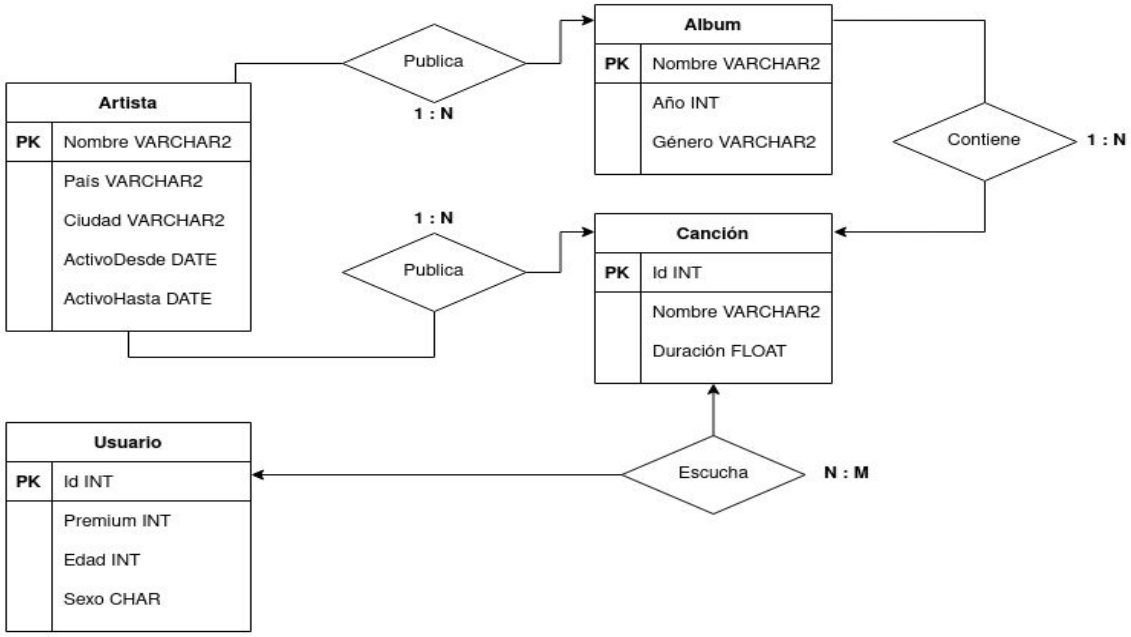
El modelo de datos viene dirigido por las consultas:

1. Necesidad de lecturas eficientes.
    - Problemas: Las lecturas son más caras y no es posible usar JOIN.
    - Solución: Lo ideal sería leer la información en la partición de un único nodo.
        - Crear una tabla para cada consulta, no importa si aumenta la duplicidad de los datos.
2. Determinar las claves de partición y agrupación para repartir y almacenar la información eficientemente.
3. Existen condiciones de filtrado (WHERE) específicas.
    - Los campos de la clave de partición son obligatorios, los de agrupación no.
    - Una clave de partición solamente soporta filtrar con los operadores “=” e “IN”.
    - Se puede filtrar sobre las columnas de agrupación con “<”, “>”, “=”, “>=”, “<=”, etc.
    - Es posible filtrar por campos que no forman parte de la clave de partición, muy desaconsejado.
        - Otra opción puede ser crear un índice secundario, no conviene abusar de ellos.

Ejemplo

Supongamos que acabamos de terminar los estudios y nos ofrecen trabajo en **Spotify** para crear una base de datos que pueda ser utilizada por su su nueva plataforma web para gestionar su catálogo de productos musicales y para la extracción de información que permita saber qué clase de música le gusta a sus clientes.

Decidimos hacer un estudio del contexto en el que se implanta nuestra base de datos para saber qué información tenemos que gestionar y en base a ella poder generar nuestro modelo de datos. Elaboramos pues, el modelo del dominio y lo representamos a través de un diagrama Entidad-Relación.



Ejemplo

Después de elaborar el modelo de dominio nos ponemos a pensar en qué características debería de tener nuestra base de datos para poder realizar su cometido de la forma más eficiente posible.

Después de pensar un rato extraemos las siguientes conclusiones:

1. Necesitamos un servicio que esté contínuamente disponible para ser usado muy frecuentemente.

2. Estamos gestionando un volumen de datos muy elevado, no solo por la información disponible de las canciones, álbumes y músicos, si no también por toda la que se va a generar como resultado de monitorear contínuamente la actividad de los usuarios.

3. Se requiere una alta velocidad para la ingesta de datos, pero también para proporcionarlos.

4. Los usuarios van a buscar la información por los identificadores más representativos de los datos, los nombres de las canciones, de los álbumes o los artistas.

5. Una vez que los usuarios se identifican en la plataforma nos interesa saber qué escucha cada uno, tal vez podríamos recomendarles alguna canción, álbum o artista.

6. Hay tanta información que no parece apropiado juntar datos de diferente origen para elaborar unos mejores informes.

Por lo que finalmente, decidimos que utilizar Cassandra va a ser la mejor opción para gestionar los datos.

Ejemplo

Nos ponemos a trabajar y elaboramos el nuevo modelo de datos partiendo de la información que interesa ser consultada.

Principalmente nos interesa lo siguiente:

1. Los datos de un artista a partir de su nombre.
2. Todos los álbumes de un artista particular.
3. La información de un álbum dado su nombre.
4. Todos los álbumes de un género musical concreto.
5. La información de una canción dado su nombre.
6. Las canciones que conforman un álbum particular.
7. Información de un usuario a partir de su identificador.
8. Canciones escuchadas por un usuario en particular.

Ejemplo

El modelo de datos que resuelve las consultas anteriores es el siguiente:

Artista_De_Nombre	
K	Nombre TEXT
	País TEXT
	Ciudad TEXT
	ActivoDesde DATE
	ActivoHasta DATE

Album_Con_Nombre	
K	NombreAlbum TEXT
CD	AñoAlbum INT
	GéneroAlbum TEXT
	NombreArtista TEXT

Canción_Con_Nombre	
K	NombreCanción TEXT
CD	AñoAlbum INT
CA	NombreAlbum TEXT
CA	NumeroCanción INT
	DuraciónCanción INT
	GéneroAlbum TEXT

Usuario_Con_Id	
K	Id UUID
	Premium BOOLEAN
	Sexo TEXT
	Edad INT

Albums_Del_Artista	
K	NombreArtista TEXT
CD	AñoAlbum INT
CA	NombreAlbum TEXT
	GéneroAlbum TEXT

Albums_Con_Género	
K	GéneroAlbum TEXT
CD	AñoAlbum INT
CA	NombreAlbum TEXT
	NombreArtista TEXT

Canciones_Del_Album	
K	NombreAlbum TEXT
CD	AñoAlbum INT
CA	NumeroCanción INT
	NombreCanción TEXT
	DuraciónCanción INT
	GéneroAlbum TEXT

Usuario_Escucha_Canción	
K	IdUsuario UUID
K	Fecha DATE
CD	Tiempo TIMESTAMP
	NombreAlbum TEXT
	AñoAlbum INT
	Canción TEXT
	Duración FLOAT

Recuerda Cassandra

- Se debe usar cuando:
  - La cantidad de operaciones de escritura es muy elevada en comparación con las lecturas.
  - Las consultas son realizadas a partir de una clave primaria bien conocida.
  - Los datos pueden dividirse uniformemente a través de múltiples nodos.
  - Interesa un procesamiento distribuido y muy escalable.
- No debe de usarse cuando:
  - Los registros de las tablas están identificados por una secuencia o autoincremento.
  - Existen demasiadas formas de acceder a las tablas.
  - Se requiere el uso de transacciones ACID.
  - Si se han de usar JOIN o subconsultas.

¿Dónde puedo probar Cassandra?

- DataStax es una empresa que proporciona Cassandra como un servicio.
- Concretamente, es la base de datos de las empresas Netflix, eBay y Adobe.
- Permite crear una cuenta gratuita con fines de estudio o aprendizaje.
    - Registrarse y listo, no requiere realizar un proceso de instalación ni configurar la BBDD.
    - Incorpora:
        - Una consola de comandos para utilizar el CQL.
        - Un tablero para la visualización de métricas.
    - Espacio considerable para pruebas: 5GB.
- Dirección: https://astra.datastax.com/register

MUCHAS GRACIAS