



Programa de formación MACHINE LEARNING AND DATA SCIENCE MLDS

Facultad de
INGENIERÍA



Módulo 3

Big Data

Unidad 2

Bases NoSQL Columnares

Clase sincrónica

Facultad de
INGENIERÍA





Bienvenida

Jorge Eliécer Camargo Mendoza, PhD.

<https://dis.unal.edu.co/~jecamargom/>

jecamargom@unal.edu.co



UNIVERSIDAD
NACIONAL
DE COLOMBIA

Departamento de Ingeniería de Sistemas e Industrial

Facultad de Ingeniería

Universidad Nacional de Colombia

Sede Bogotá



Tabla de contenidos

- 1** ¿Qué son las bases NoSQL?
- 2** Tipos de bases NoSQL
- 3** Modelo CAP
- 4** Definición de bases de datos columnares y ejemplos
- 5** ¿Qué es Cassandra?
- 6** ¿Quién usa Cassandra?
- 7** Query-First Design
- 8** Anillos y tokens
- 9** Arquitectura de Cassandra
- 10** Partition y Clustering Keys
- 11** Conceptos CQL (CRUD)

Objetivos de aprendizaje**Unidad 2 – Bases NoSQL Columnares**

Al finalizar la unidad usted deberá ser capaz de:

1



Describir las diferencias entre los distintos tipos de bases de datos NoSQL y su apropiada selección por medio del modelo CAP.

2



Entender el modelo de datos y la arquitectura en bases de datos columnares como Cassandra.

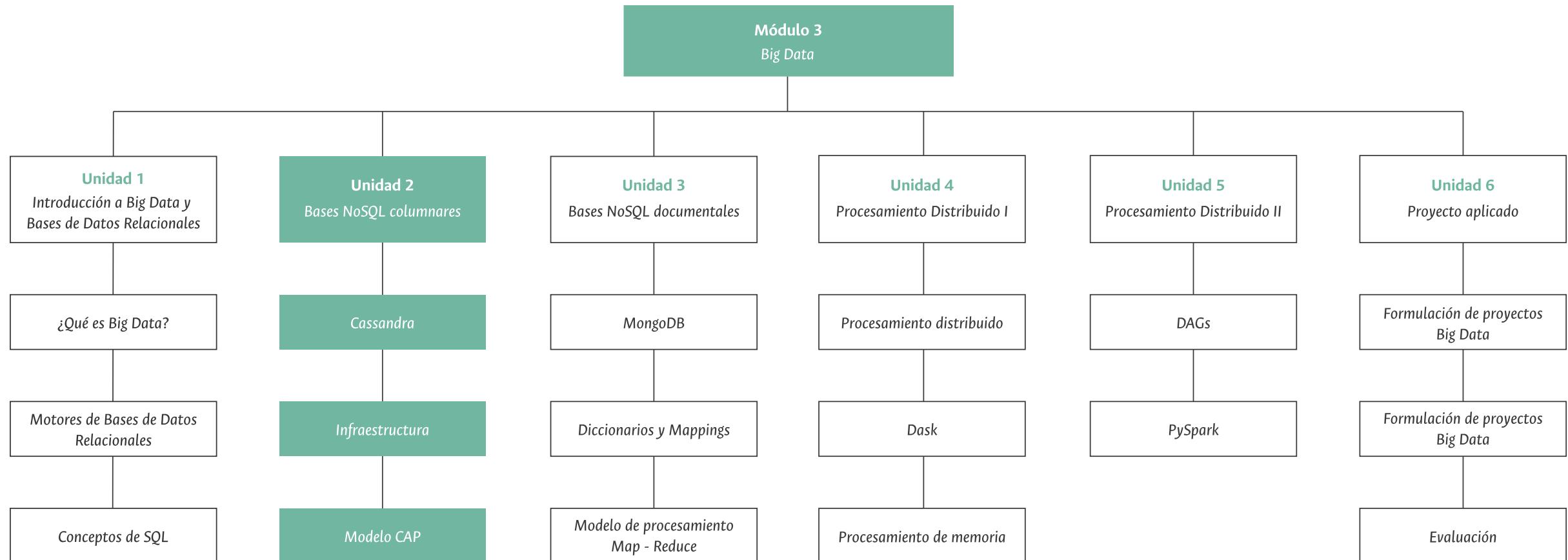
3



Utilizar el lenguaje de consulta CQL para manipular información desde la base de datos Cassandra y con el lenguaje de programación Python.



Mapa de contenidos de la unidad

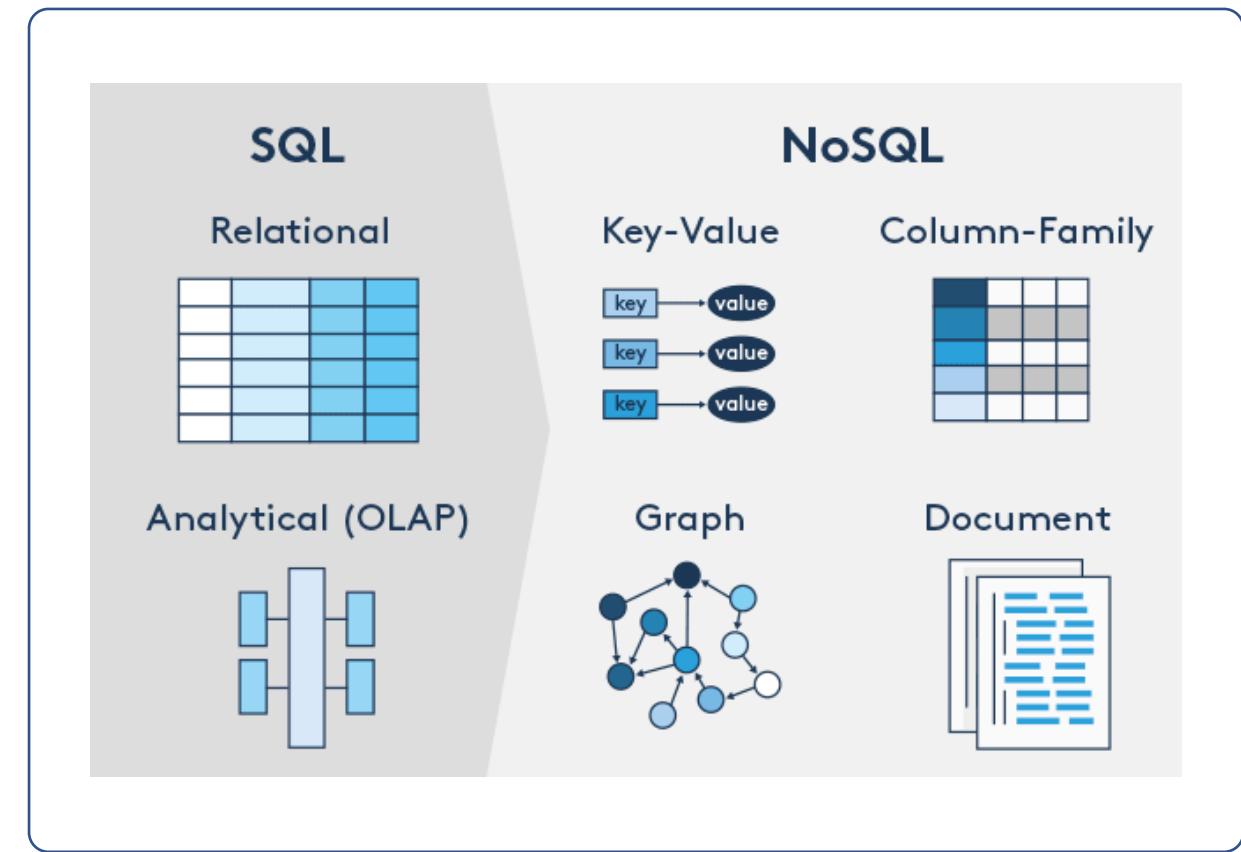


1

¿Qué son las bases NoSQL?

Las bases de datos NoSQL son aquellas que se destacan por el hecho de que pueden administrar grandes volúmenes de datos no estructurados de manera distribuida y con esquemas flexibles que se adaptan a los requisitos de las aplicaciones más modernas. El término NoSQL proviene de las siglas en inglés de *Not Only Structured Query Language*.

En las bases NoSQL los datos almacenados no requieren estructuras fijas como tablas, normalmente no soportan operaciones JOIN, no garantizan los principios ACID completamente y soportan una gran cantidad de usuarios concurrentemente.



¿Qué son las bases NoSQL?

Ventajas de las bases de datos NoSQL

01



Escalabilidad

Generalmente, las bases NoSQL ofrecen esquemas flexibles que permiten un desarrollo más rápido y más iterativo. El modelo de datos flexible hace que las bases de datos NoSQL sean ideales para **datos semiestructurados y no estructurados**.

Flexibilidad

02



Las bases de datos NoSQL generalmente están diseñadas para escalar usando clústeres distribuidos de hardware en lugar de escalar añadiendo capacidad a un servidor (**escalamiento vertical**), como las SQL, lo que permite un **escalamiento horizontal** que mejora el rendimiento de la base de datos y soportar mayor cantidad de usuarios concurrentes.

03



Arquitectura

Las bases de datos SQL son indicadas cuando la cantidad de datos no son extremadamente grandes, mientras que las NoSQL son ideales para manejar **grandes volúmenes de datos**.

Almacenamiento

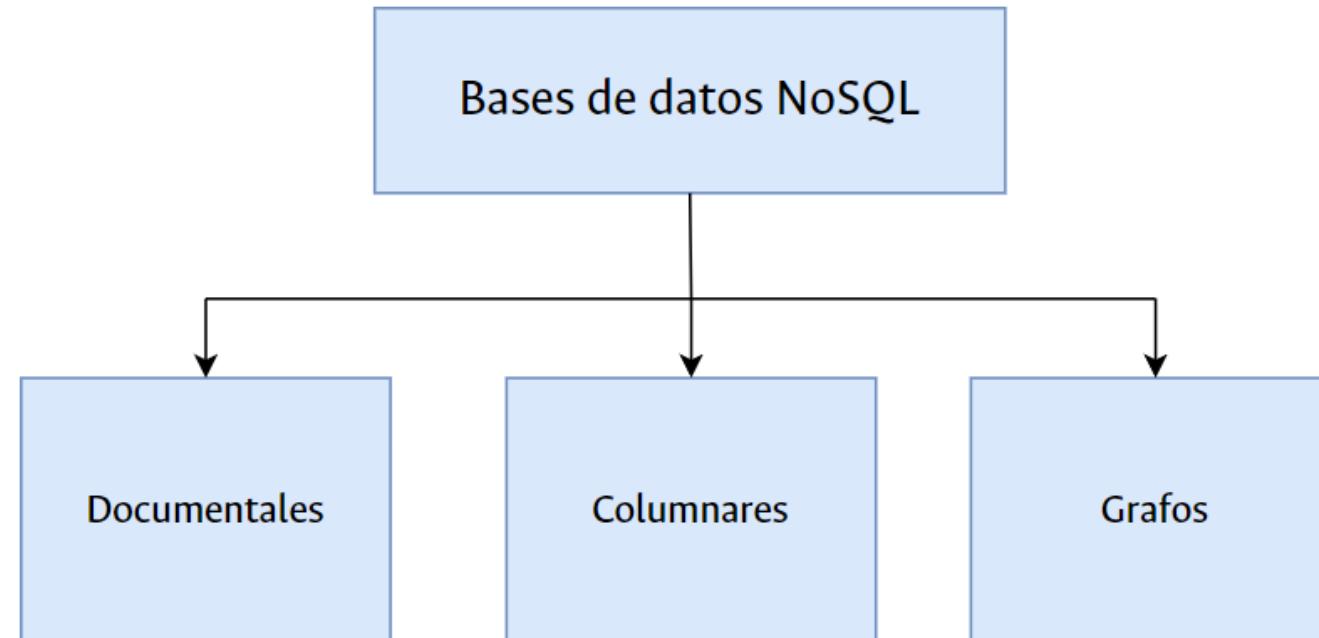
04



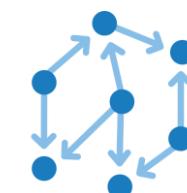
Ofrecen una arquitectura distribuida donde la información puede estar compartida en varias **máquinas de pocos recursos** mediante mecanismos de tablas hash distribuidas, reduciendo así los costos de una arquitectura costosa y monolítica como las SQL.

¿Cuáles son los Tipos de bases de datos NoSQL?

Hay tres tipos importantes de bases de datos NoSQL:



id_dueño	nombre	nacimiento
{1: 1001526983, 2: 1000253654, 3: 1010258695, 4: 1013717174}	{1:'Bolt', 2:'Golden', 3:'Crema', 4:'Luna'}	{1:'2017/02/23', 2:'2014/04/17', 3:'2020/05/21', 4:'2021/01/01'}



¿Cuáles son los tipos de bases NoSQL?

Documentales

Bases de datos que almacenan y consultan datos como documentos de tipo JSON. Conveniente:

- Cuando el diseño de la base de datos cambiará en el futuro o cambia constantemente porque estos motores permiten añadir o eliminar campos de manera muy flexible.
- Cuando las entidades a almacenar no necesariamente manejan los mismos campos, ya que estos motores permiten almacenar entidades con diferente cantidad de campos o incluso campos diferentes (mirar ejemplo).
- Porque es amigable con el desarrollo; es visualmente más cómodo y usa el mismo formato de documento implementado en las aplicaciones.



```
[{"year": 2013, "title": "Turn It Down, Or Else!", "info": {"directors": ["Alice Smith", "Bob Jones"], "release_date": "2013-01-18T00:00:00Z", "rating": 6.2, "genres": ["Comedy", "Drama"], "actors": ["David Matthewman", "Jonathan G. Neff"]}}, {"year": 2015, "title": "The Big New Movie", "info": {"plot": "Nothing happens at all.", "rating": 0}}]
```

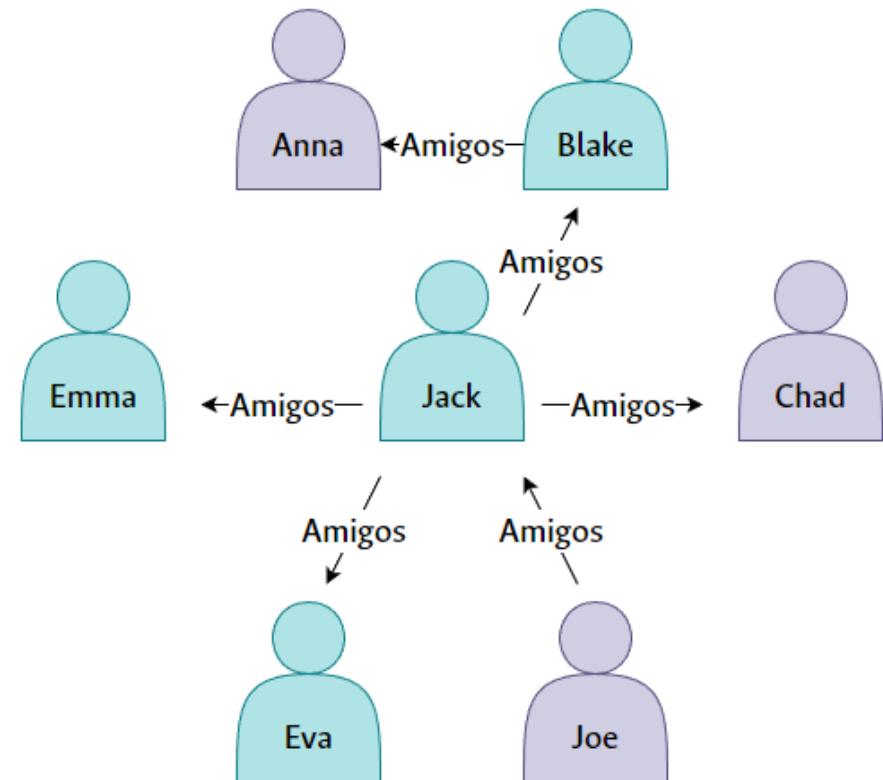
Una colección de dos documentos diferentes (Documento = Registro; Colección = Tabla)

¿Cuáles son los tipos de bases NoSQL?

Orientadas a grafos

Bases de datos diseñadas para almacenar y navegar relaciones rápidamente:

- Usan nodos para almacenar entidades de datos y arcos para almacenar relaciones entre las entidades.
- Cada arco tiene nodo inicial y final, tipo y dirección.
- Cada arco puede describir cualquier relación, por ejemplo, padre-hijo, pertenencia, acciones, etc.
- Recorrer entre las relaciones es fácil y rápido porque ya están calculadas y no se debe hacer el cálculo en el tiempo de consulta.
- Convenientes para implementar Apps donde prevalezcan las relaciones típicas de las redes sociales o motores de recomendaciones.



Pequeña Red Social



¿Cuáles son los tipos de bases NoSQL?

Orientadas a columnas

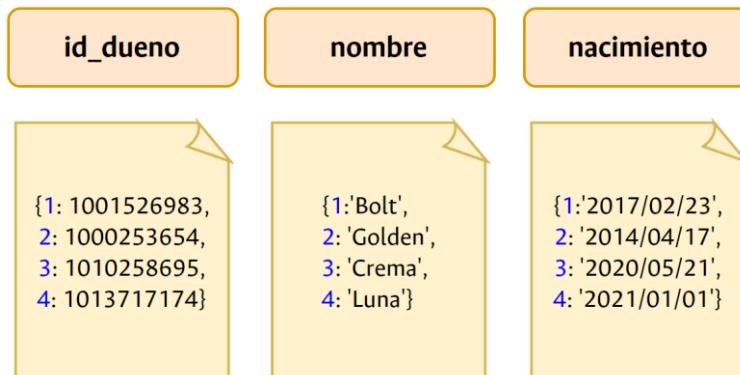
Bases diseñadas para almacenar y procesar datos en *columnas* en lugar de *filas*. El propósito es ser usadas en aplicaciones analíticas (machine learning, big data, etc) en lugar de transaccionales:

- El motor usa queries que intervienen sólo en las columnas, esto permite mejorar el tiempo de búsqueda y retorno de datos, ya que no tendrán que recorrer campos innecesarios sino sólo se recorrerán los datos adecuados. La lectura se ve beneficiada, pero la escritura no.



id	id_dueno	nombre	nacimiento
'1'	1001526983	'Bolt'	'2017/02/23'
'2'	1000253654	'Golden'	'2014/04/17'
'3'	1010258695	'Crema'	'2020/05/21'
'4'	1013717174	'Luna'	'2021/01/01'

Base de datos clásica relacional



Base de datos columnar

3

Modelos CAP

El teorema CAP establece que en cualquier sistema de gestión de datos distribuidos de forma masiva, solo se pueden garantizar dos de las tres propiedades:



Consistency (Consistencia)

Cuando una transacción cambia datos en un sistema distribuido que tiene nodos replicados debe reflejarse en todos los nodos de la base de datos, esto garantiza que siempre que se acceda a la información cualquiera de los nodos puede responder y la información siempre será la misma.



Availability (Disponibilidad)

Significa que el sistema debe operar continuamente y tener tiempos de respuesta aceptables. Este comportamiento debe persistir si hay más de un nodo y uno se encuentra en mal estado.

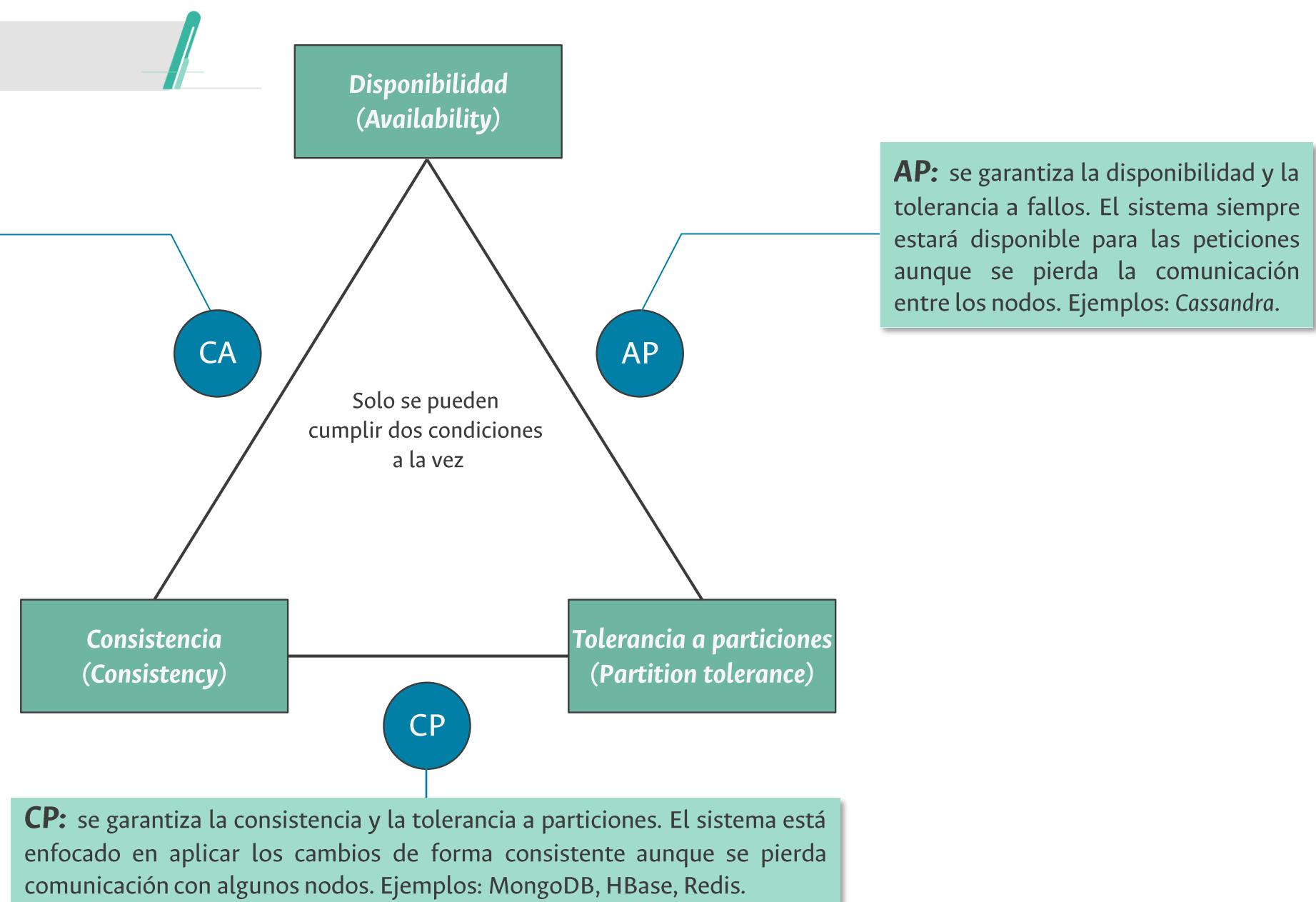


Partition Tolerance (Tolerancia a fallos)

Los fallos de los nodos o de las conexiones entre ellos en una red no deben afectar al sistema. Los nodos se pueden agregar o quitar en cualquier momento sin tener que detener la ejecución del entorno.

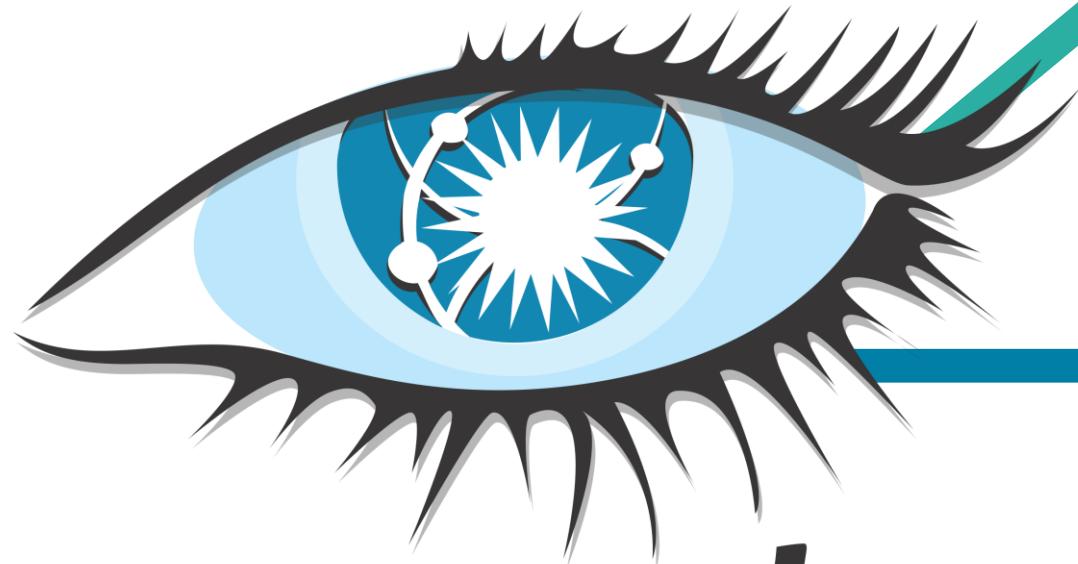
Modelos CAP

CA: no permite el particionado de los datos, porque se garantiza que estos siempre son iguales y el sistema estará disponible respondiendo a todas las peticiones. Ejemplos: RDBMSs (MySQL, PostgreSQL, etc), Neo4J.



5

¿Qué es Cassandra?



cassandra

Es un sistema de gestión de bases de datos de código abierto para bases de datos muy grandes, pero estructuradas

Pertenece a las bases de datos NoSQL columnares. Su filosofía es de tipo llave – valor.

Cassandra cuenta con un enfoque redundante, lo que reduce mucho la probabilidad de fallos.

Cassandra

Características

Características



Las instancias de Cassandra se distribuyen en nodos iguales que se comunican entre sí (P2P), es decir, no hay maestro-esclavo, lo que da buen soporte entre varios datacenters, con redundancia y réplicas síncronas.

Ofrece alta disponibilidad a cambio de ser eventualmente consistente. Si en alguno de los nodos se cae el servicio, no se degradará el sistema. La consistencia es eventual, pero tratable

Cassandra escala linealmente, lo que quiere decir que tiene un rendimiento de forma lineal respecto al número de nuevos nodos que añadamos.

Proporciona un rendimiento muy alto, especialmente durante las escrituras.

Ofrece una detección y recuperación de fallos transparentes para los nodos que no pueden ser fácilmente restaurados o reemplazados.

No existe el concepto el concepto de integridad referencial o llaves foráneas.

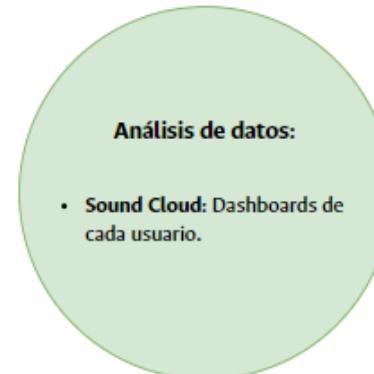
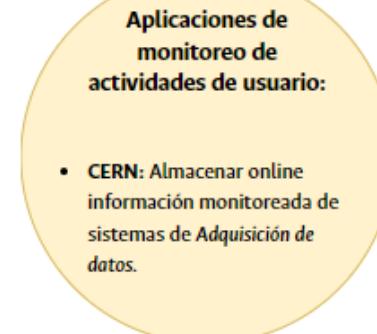
Cassandra partitiona las filas con el objetivo de reorganizarlas a lo largo distintas tablas.

5

¿Quién usa Cassandra? ¿Para qué puede usarse?



webex
by CISCO



NETFLIX

globo.com

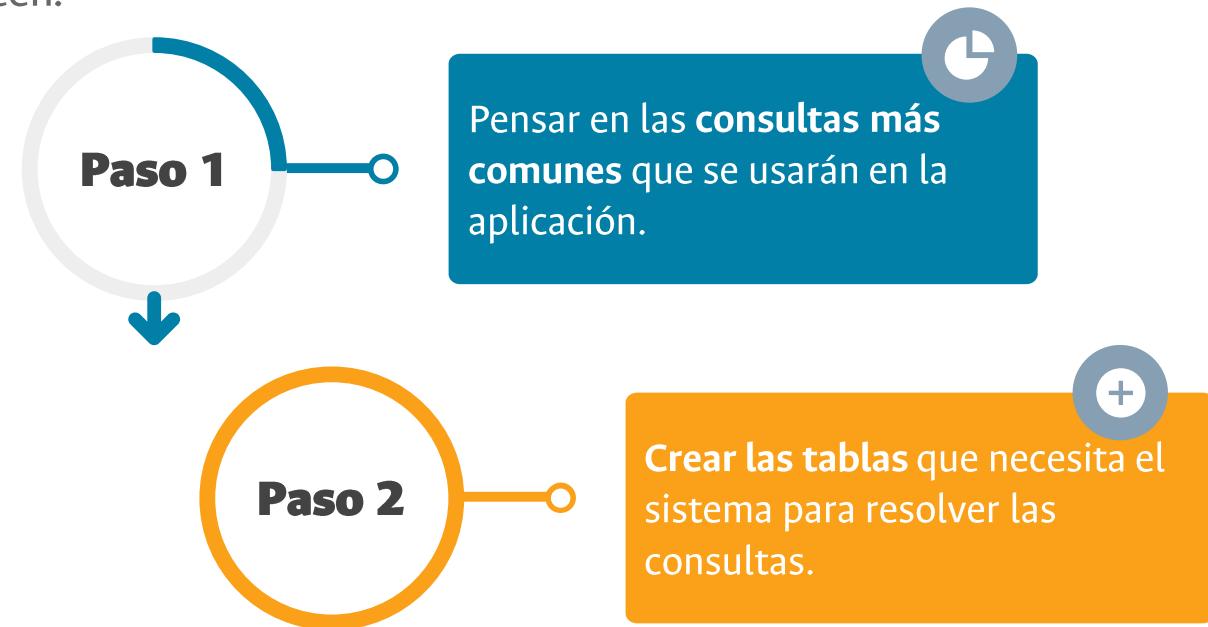
AppScale

7

Query – First Design



En Cassandra no se comienza con el modelo de datos como ocurre en las bases de datos relacionales, se inicia con el *Query-First Model/Design* o modelo de consulta. En lugar de modelar los datos primero y luego escribir las consultas en base a ellos, en Cassandra se modelan las consultas y los datos se organizan con base en los *queries* que se realicen.



7

Anillos y Tokens

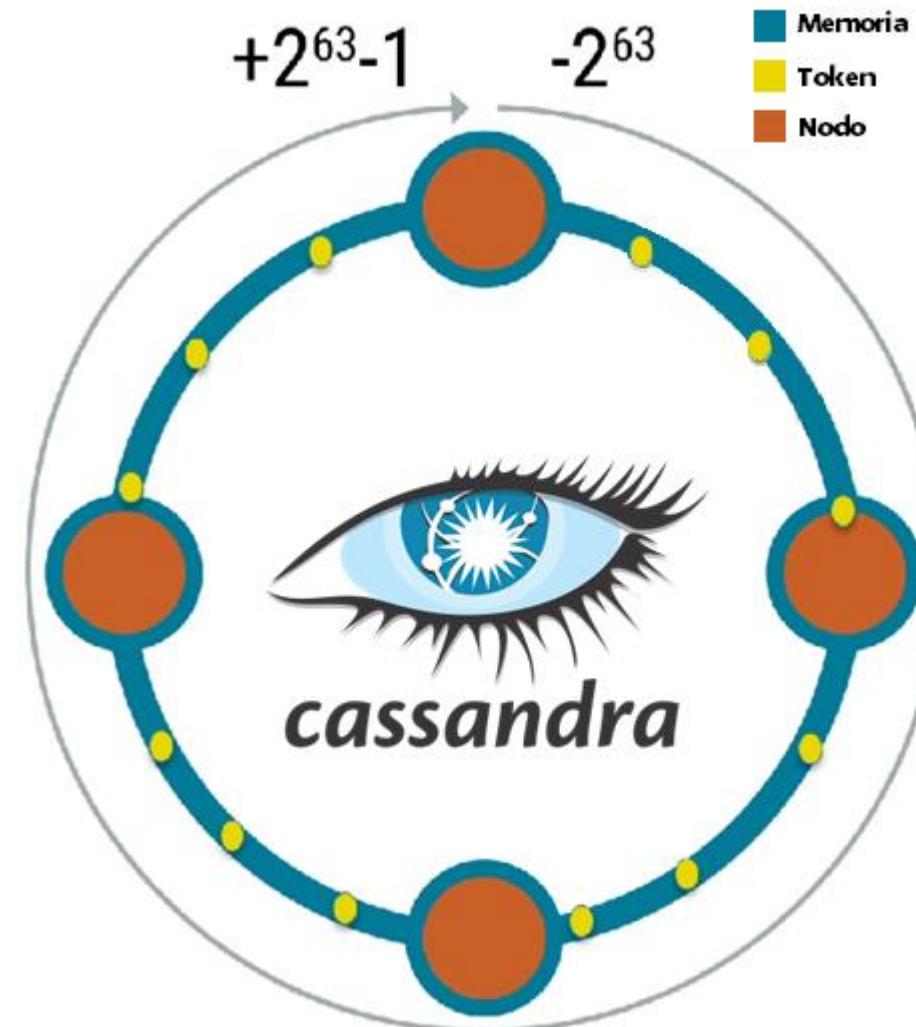
Anillos:

Son un grupo de nodos que se distribuyen, equitativamente, rangos de memoria (líneas azules) que guardarán los datos. Los nodos están representados como los círculos naranjados.

Token:

Es un valor entero, calculado mediante una función hash, que se encuentra entre -2^{63} y $(2^{63}) - 1$ (rango actual que posee Cassandra para almacenamiento) y representa una cota superior de un subrango de memoria que almacena datos. Los tokens son los puntos amarillos en el diagrama

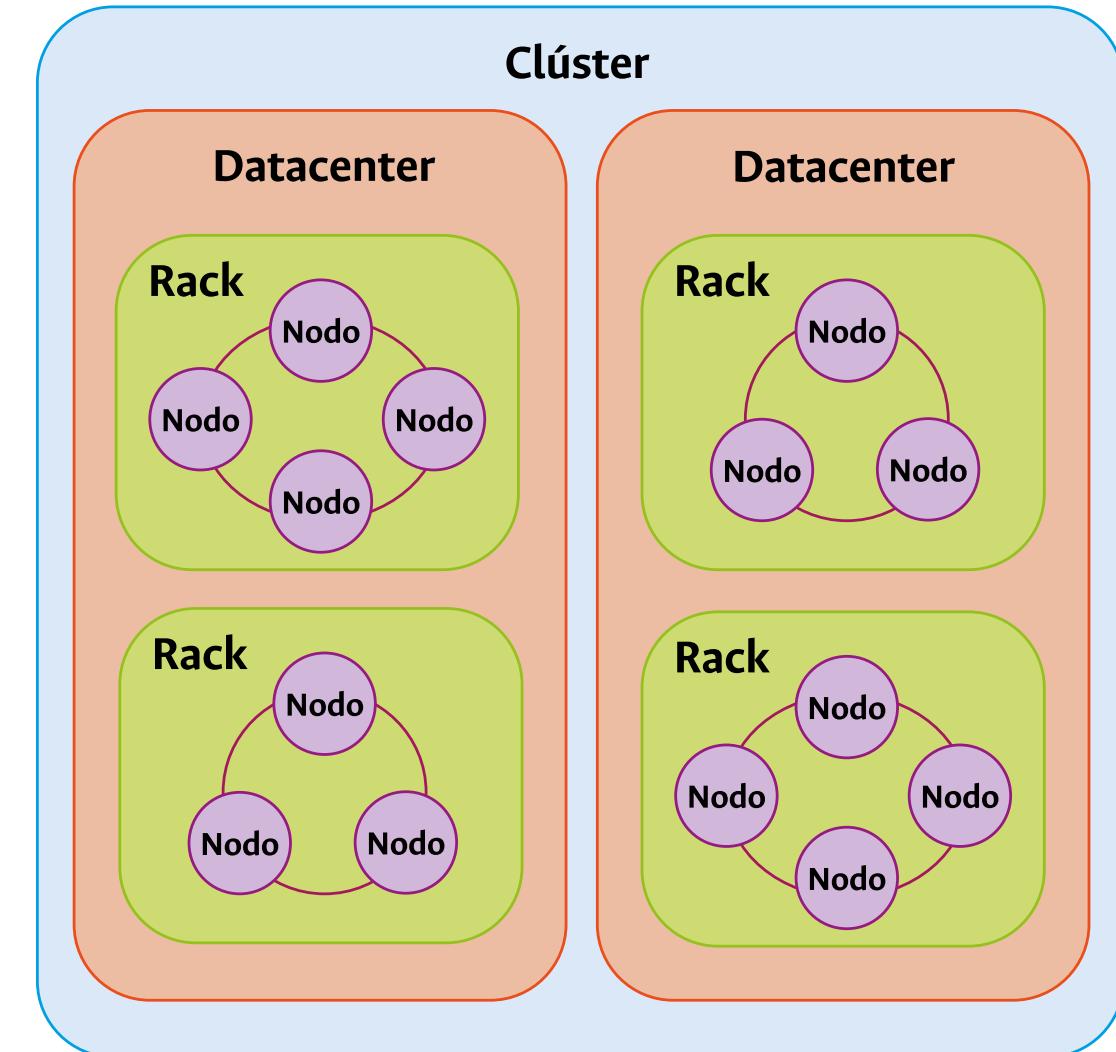
Relación Anillo-Token: Los grupos de nodos (anillos) se distribuyen rangos de memoria que a su vez están acotados por los tokens.



9

Arquitectura de Cassandra

- La arquitectura de Cassandra está creada teniendo en cuenta que alguna parte del sistema puede fallar, es por ello que está construida como un sistema peer-to-peer.
- Cada nodo se comunica con otro a través de Gossip, un protocolo que sirve para intercambiar información entre los nodos del clúster continuamente.
- La replicación en Cassandra ocurre de tal manera que cada conjunto de datos replicado no se encuentre dentro de un mismo rack usando una técnica denominada snitch.
- Un snitch determina a qué racks y datacenters pertenece un nodo en particular y, con respecto a eso, determina dónde terminarán las réplicas de los datos.



Arquitectura de Cassandra

Componentes



Clúster

Contiene uno o más *datacenters*. Uno o más *clústers* combinados forman una base de datos en Cassandra.

Datacenter

Conjunto lógico de *racks* configurados con fines de replicación. Esto ayuda a reducir la latencia y evitar que las transacciones se vean afectadas por otras cargas de trabajo. Están distribuidos geográficamente

Rack

Agrupación lógica de nodos. La base de datos utiliza *racks* para garantizar que las particiones se distribuyan correctamente en todos los nodos y así tener mayor tolerancia a fallos y disponibilidad. Están ubicados dentro de un mismo lugar.

Nodo

Componente básico de Cassandra. Representa una máquina funcional en la que se almacenan los datos. Se organizan en una arquitectura peer-to-peer con una topología de red en anillo.

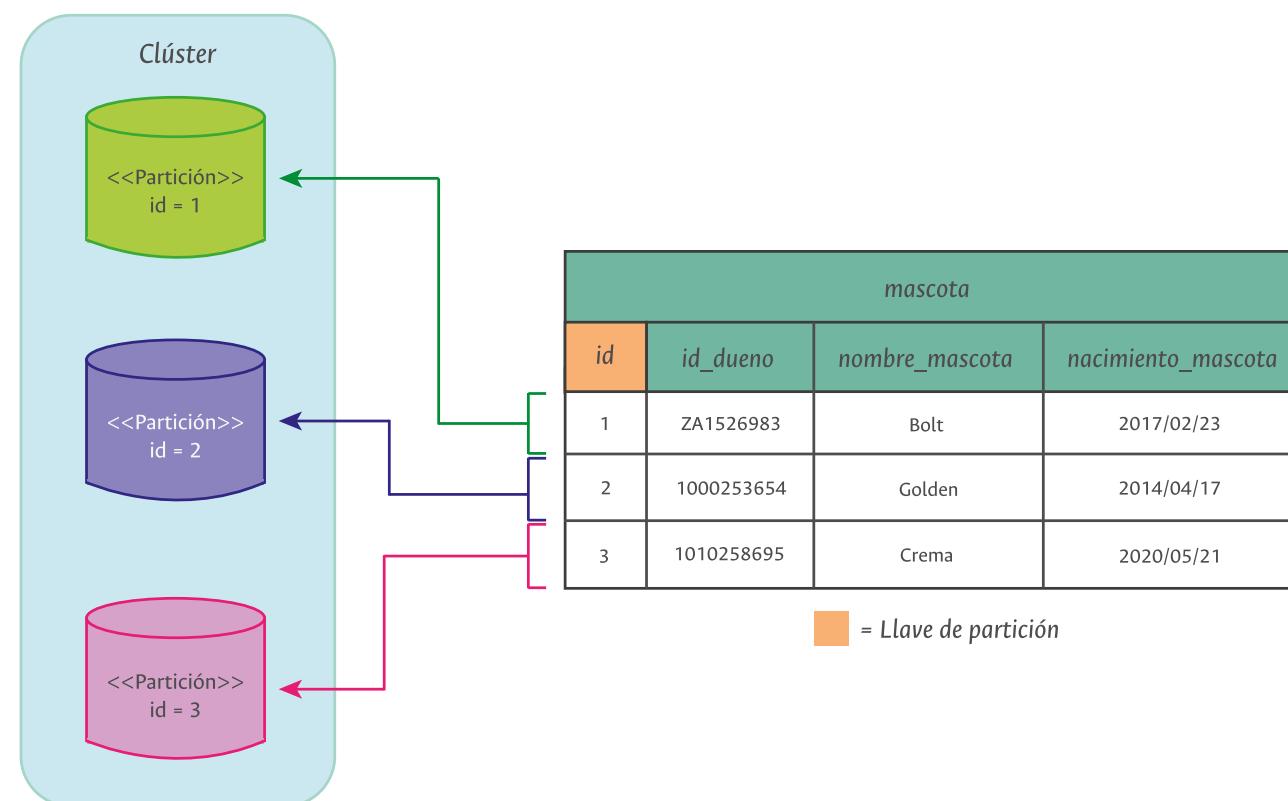
10

Llaves de partición y agrupamiento (Partition y Clustering Keys)



Partition keys (Llaves de partición)

Es la responsable de la distribución de datos a través de los nodos, es decir, la llave de partición determina qué nodo almacenará una fila (o filas) específica de la tabla. El objetivo principal de una clave de partición es distribuir los datos de manera uniforme en un clúster y permitir una consulta de datos de manera eficiente.



Llaves de partición y agrupamiento

Llaves de partición

Una llave de partición compuesta puede dividir los datos y almacenar aquellos que estén relacionados en particiones separadas.



```
CREATE TABLE mascota {  
    id                      TEXT,  
    id_dueno                INT,  
    nombre_mascota          TEXT,  
    nacimiento_mascota      TEXT,  
    PRIMARY KEY (id)  
}
```

Llave de partición



```
CREATE TABLE mascota {  
    id                      TEXT,  
    id_dueno                INT,  
    nombre_mascota          TEXT,  
    nacimiento_mascota      TEXT,  
    PRIMARY KEY ((id, id_dueno))  
}
```

Llave de partición compuesta

Llaves de partición y agrupamiento

Llaves de agrupamiento



Clustering keys (Llaves de agrupamiento)

Clasifican u ordenan los datos dentro de una partición. Estas llaves determinan el orden en el que se depositan los datos dentro de la partición. La llave de agrupamiento identifica un registro único dentro de la misma partición si la llave de partición no es un identificador único. Puede ser parte de la cláusula **WHERE** y/o **GROUP BY** en una consulta.



```
CREATE TABLE mascota {  
    id                      TEXT,  
    id_dueno                INT,  
    nombre_mascota          TEXT,  
    nacimiento_mascota      TEXT,  
    PRIMARY KEY (id, nombre_mascota)  
}
```

Se pueden especificar varias columnas al definir una llave primaria que contenga tanto llaves de partición como llaves de agrupamiento. La llave resultante es una **llave primaria compuesta**.

11

Conceptos CQL (CRUD)

CREATE

**Crear**

```
CREATE TABLE
    veterinaria.mascota (
        id TEXT,
        id_dueno INT,
        nombre_mascota TEXT,
        nacimiento_mascota TEXT,
        color_mascota TEXT,
        PRIMARY KEY ((id, id_dueno),
                     nombre_mascota)
    );
```

READ

**Leer**

```
SELECT
    *
FROM
    veterinaria.mascota
WHERE
    id = '1' AND
    id_dueno = 1001526983;
```

UPDATE

**Actualizar**

```
UPDATE
    veterinaria.mascota
SET
    color_mascota = 'marfil'
WHERE
    id = '3' AND
    id_dueno = 1010258695 AND
    nombre_mascota = 'Crema'
    ;
```

DELETE

**Borrar**

```
DELETE FROM
    veterinaria.mascota
WHERE
    id = '4' AND
    id_dueno = 1013717174
    ;
```

Conceptos CQL

Operaciones CRUD



Crear

En CQL, las sentencias CREATE e INSERT son muy parecidas a SQL:

```
● ● ●  
CREATE TABLE  
    veterinaria.mascota (  
        id TEXT,  
        id_dueno INT,  
        nombre_mascota TEXT,  
        nacimiento_mascota TEXT,  
        color_mascota TEXT,  
        PRIMARY KEY ((id, id_dueno),nombre_mascota)  
    );  
  
INSERT INTO  
    veterinaria.mascota (  
        id, id_dueno, nombre_mascota, nacimiento_mascota, color_mascota  
    )  
VALUES  
    ('1', 1001526983, 'Bolt', '2017/02/23', 'blanco')  
;
```

id	id_dueno	nombre_mascota	nacimiento_mascota	color_mascota
'1'	1001526983	'Bolt'	'2017/02/23'	'blanco'
'2'	1000253654	'Golden'	'2014/04/17'	'dorado'
'3'	1010258695	'Crema'	'2020/05/21'	'beige'
'4'	1013717174	'Luna'	'2021/01/01'	'negro'

Conceptos CQL

 Operaciones CRUD

Leer

Es obligatorio usar **TODAS** las llaves de partición para ejecutar cualquier sentencia WHERE o GROUP BY dentro de una sentencia SELECT:

● ● ●

```
SELECT
 *
FROM
    veterinaria.mascota
WHERE
    id = '1' AND
    id_dueno = 1001526983
;
```

id	id_dueno	nombre_mascota	nacimiento_mascota	color_mascota
'1'	1001526983	'Bolt'	'2017/02/23'	'blanco'
'2'	1000253654	'Golden'	'2014/04/17'	'dorado'
'3'	1010258695	'Crema'	'2020/05/21'	'beige'
'4'	1013717174	'Luna'	'2021/01/01'	'negro'

Conceptos CQL

Operaciones CRUD



Leer

Es opcional usar las llaves de agrupamiento para ejecutar cualquier sentencia WHERE o GROUP BY dentro de una sentencia SELECT :



```
SELECT
  *
FROM
  veterinaria.mascota
WHERE
  id = '1' AND
  id_dueño = 1001526983 AND
  nombre_mascota = 'Bolt'
;
```

id	id_dueño	nombre_mascota	nacimiento_mascota	color_mascota
'1'	1001526983	'Bolt'	'2017/02/23'	'blanco'
'2'	1000253654	'Golden'	'2014/04/17'	'dorado'
'3'	1010258695	'Crema'	'2020/05/21'	'beige'
'4'	1013717174	'Luna'	'2021/01/01'	'negro'

Conceptos CQL

 Operaciones CRUD

Leer

Esta consulta no funciona porque se está usando sólo una *llave de partición*:



```
SELECT
  *
FROM
  veterinaria.mascota
WHERE
  id = '1'
;
```



id	id_dueno	nombre_mascota	nacimiento_mascota	color_mascota
'1'	1001526983	'Bolt'	'2017/02/23'	'blanco'
'2'	1000253654	'Golden'	'2014/04/17'	'dorado'
'3'	1010258695	'Crema'	'2020/05/21'	'beige'
'4'	1013717174	'Luna'	'2021/01/01'	'negro'

Conceptos CQL

 Operaciones CRUD

Leer

Esta consulta no funciona porque **también** se está usando sólo una **llave de partición**:



```
SELECT
  *
FROM
  veterinaria.mascota
WHERE
  id_dueno = 1001526983
;
```



id	id_dueno	nombre_mascota	nacimiento_mascota	color_mascota
'1'	1001526983	'Bolt'	'2017/02/23'	'blanco'
'2'	1000253654	'Golden'	'2014/04/17'	'dorado'
'3'	1010258695	'Crema'	'2020/05/21'	'beige'
'4'	1013717174	'Luna'	'2021/01/01'	'negro'

Conceptos CQL

 Operaciones CRUD

Leer

Consulta incorrecta porque se está usando un campo que no es llave de partición ni llave de agrupamiento:

● ● ●

```
SELECT
  *
FROM
  veterinaria.mascota
WHERE
  id = '1' AND
  id_dueño = 1001526983 AND
  nombre_mascota = 'Bolt' AND
  nacimiento_mascota = '2021/01/01'
;
```



id	id_dueño	nombre_mascota	nacimiento_mascota	color_mascota
'1'	1001526983	'Bolt'	'2017/02/23'	'blanco'
'2'	1000253654	'Golden'	'2014/04/17'	'dorado'
'3'	1010258695	'Crema'	'2020/05/21'	'beige'
'4'	1013717174	'Luna'	'2021/01/01'	'negro'

Conceptos CQL

 Operaciones CRUD

Las sentencias UPDATE deben usar siempre **tanto las llaves de partición como las llaves de agrupamiento**:

● ● ●

```
UPDATE
    veterinaria.mascota
SET
    color_mascota = 'marfil'
WHERE
    id = '3' AND
    id_dueño = 1010258695 AND
    nombre_mascota = 'Crema'
;
```

id	id_dueño	nombre_mascota	nacimiento_mascota	color_mascota
'1'	1001526983	'Bolt'	'2017/02/23'	'blanco'
'2'	1000253654	'Golden'	'2014/04/17'	'dorado'
'3'	1010258695	'Crema'	'2020/05/21'	'marfil'
'4'	1013717174	'Luna'	'2021/01/01'	'negro'

Conceptos CQL

 Operaciones CRUD

Sentencia UPDATE incorrecta porque falta el uso de la *llave de agrupamiento*:



```
UPDATE
    veterinaria.mascota
SET
    color_mascota = 'marfil'
WHERE
    id = '3' AND
    id_dueno = 1010258695
;
```



id	id_dueno	nombre_mascota	nacimiento_mascota	color_mascota
'1'	1001526983	'Bolt'	'2017/02/23'	'blanco'
'2'	1000253654	'Golden'	'2014/04/17'	'dorado'
'3'	1010258695	'Crema'	'2020/05/21'	'marfil'
'4'	1013717174	'Luna'	'2021/01/01'	'negro'

Conceptos CQL

Operaciones CRUD



Borrar

Las sentencias **DELETE** funcionan usando **todas las llaves de partición o las llaves de partición y de agrupamiento al mismo tiempo**. Igual que las sentencias **SELECT**:

```
● ● ●  
DELETE FROM  
    veterinaria.mascota  
WHERE  
    id = '4' AND  
    id_dueño = 1013717174  
;
```

id	id_dueño	nombre_mascota	nacimiento_mascota	color_mascota
'1'	1001526983	'Bolt'	'2017/02/23'	'blanco'
'2'	1000253654	'Golden'	'2014/04/17'	'dorado'
'3'	1010258695	'Crema'	'2020/05/21'	'marfil'
'4'	1013717174	'Luna'	'2021/01/01'	'negro'



id	id_dueño	nombre_mascota	nacimiento_mascota	color_mascota
'1'	1001526983	'Bolt'	'2017/02/23'	'blanco'
'2'	1000253654	'Golden'	'2014/04/17'	'dorado'
'3'	1010258695	'Crema'	'2020/05/21'	'marfil'

Conceptos CQL

Operaciones CRUD



Borrar

Las sentencias **DELETE** funcionan usando **todas las llaves de partición o las llaves de partición y de agrupamiento al mismo tiempo**. Igual que las sentencias **SELECT** :

```
● ● ●  
DELETE FROM  
    veterinaria.mascota  
WHERE  
    id = '4' AND  
    id_dueño = 1013717174 AND  
    nombre_mascota = 'Crema'  
;
```

id	id_dueño	nombre_mascota	nacimiento_mascota	color_mascota
'1'	1001526983	'Bolt'	'2017/02/23'	'blanco'
'2'	1000253654	'Golden'	'2014/04/17'	'dorado'
'3'	1010258695	'Crema'	'2020/05/21'	'marfil'
'4'	1013717174	'Luna'	'2021/01/01'	'negro'



id	id_dueño	nombre_mascota	nacimiento_mascota	color_mascota
'1'	1001526983	'Bolt'	'2017/02/23'	'blanco'
'2'	1000253654	'Golden'	'2014/04/17'	'dorado'
'3'	1010258695	'Crema'	'2020/05/21'	'marfil'

Conceptos CQL

 Operaciones CRUD

Borrar

Sentence **DELETE** incorrecta porque se está usando sólo una *llave de partición*:



```
DELETE FROM
    veterinaria.mascota
WHERE
    id = '4'
;
```



id	id_dueno	nombre_mascota	nacimiento_mascota	color_mascota
'1'	1001526983	'Bolt'	'2017/02/23'	'blanco'
'2'	1000253654	'Golden'	'2014/04/17'	'dorado'
'3'	1010258695	'Crema'	'2020/05/21'	'marfil'
'4'	1013717174	'Luna'	'2021/01/01'	'negro'

Conceptos CQL

 Operaciones CRUD

Borrar

Sentencia **DELETE** incorrecta porque se está usando un *campo adicional que no es llave de partición o de agrupamiento*:

```
● ● ●  
DELETE FROM  
    veterinaria.mascota  
WHERE  
    id = '4' AND  
    id_dueño = 1013717174 AND  
    nombre_mascota = 'Crema' AND  
    nacimiento_mascota = '2021/01/01'  
;
```



id	id_dueño	nombre_mascota	nacimiento_mascota	color_mascota
'1'	1001526983	'Bolt'	'2017/02/23'	'blanco'
'2'	1000253654	'Golden'	'2014/04/17'	'dorado'
'3'	1010258695	'Crema'	'2020/05/21'	'marfil'
'4'	1013717174	'Luna'	'2021/01/01'	'negro'



Despedida

¡Gracias por su atención!

**Jorge Eliécer Camargo
Mendoza, PhD.**

<https://dis.unal.edu.co/~jecamargom/>

jecamargom@unal.edu.co



UNIVERSIDAD
NACIONAL
DE COLOMBIA

Departamento de Ingeniería de Sistemas e Industrial
Facultad de Ingeniería
Universidad Nacional de Colombia
Sede Bogotá



Referencias

- Alapati, S. (2018). *Expert Apache Cassandra Administration: Install, configure, optimize, and secure Apache Cassandra databases.* (3era edición). Recuperado de <https://n9.cl/0o9h7>
- Meier, A., Kaufmann, M. (2019). *SQL & NoSQL Databases: Models, Languages, Consistency Options and Architectures for Big Data Management.* (1era edición). Recuperado de <https://n9.cl/526zr>
- Cassandra Datacenter and Racks. (6 de marzo de 2019). DataStax. <https://www.datastax.com/blog/distributed-database-things-know-cassandra-datacenter-racks>
- Qué es el teorema CAP y cómo elegir la base de datos para tu proyecto. (21 de noviembre de 2017). Platzi. <https://platzi.com/blog/que-es-el-teorema-cap-y-como-elegir-la-base-de-datos-para-tu-proyecto/>
- Cassandra Partition Key, Composite Key, and Clustering Key (12 de enero de 2022). Baeldung. <https://www.baeldung.com/cassandra-cluster-datacenters-racks-nodes>
- Apache Cassandra: gestión distribuida de grandes bases de datos. (15 de octubre de 2020). IONOS. <https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/apache-cassandra/>



Recursos adicionales

- W3schools, (s. f.), SQL Tutorial, <https://www.w3schools.com/sql/>
- Tutorialspoint, (s. f.), SQL Tutorial, <https://www.tutorialspoint.com/sql/index.htm>



Derechos de imágenes

- McMurryjulie (s.f) [imagen] <https://pixabay.com/es/vectors/esquema-base-de-datos-tablas-de-datos-1895779/>
- 200 Degrees (s.f) [imagen] <https://pixabay.com/es/vectors/computaci%C3%B3n-en-la-nube-nube-1989339/>
- OpenClipart-Vectors (s,f) [imagen] <https://pixabay.com/es/vectors/computaci%C3%B3n-en-la-nube-host-servidor-2023902/>
- Wanicon. (s. f.). Sql server free icon. [Icono]. https://www.flaticon.com/free-icon/sql-server_2286723
- Freepik. (s. f.). File free icon. [Icono]. https://www.flaticon.com/free-icon/file_3143540
- Freepik. (s. f.). Reading free icon. [Icono]. https://www.flaticon.com/free-icon/reading_3043907
- Freepik. (s. f.). File free icon. [Icono]. https://www.flaticon.com/free-icon/file_2246622
- Freepik. (s. f.). File free icon. [Icono]. https://www.flaticon.com/free-icon/file_3143535
- Freepik. (s. f.). File free icon. [Icono]. https://www.flaticon.com/free-icon/cancel_753345



Derechos de imágenes

- Eucalyp. (s.f.). Machine learning. [Icono]. https://www.flaticon.com/free-icon/machine-learning_2857322?term=automatic&page=1&position=30
- Becris. (s.f.). Path. [Icono]. https://www.flaticon.com/free-icon/planning_3078973?term=route&page=1&position=5
- Pixel perfect. (s.f.). Import. [Icono]. https://www.flaticon.com/free-icon/data-transfer_2879510?term=transaction&page=1&position=39
- Phatplus. (s.f.). Work time. [Icono]. https://www.flaticon.com/free-icon/work-time_2255313?term=hard%20worker&page=1&position=24
- Freepik. (s.f.). Server free icon. [Icono]. https://www.flaticon.com/free-icon/server_689393
- Freepik. (s.f.). Colección de iconos de móviles vector gratuito. [Icono]. https://www.freepik.es/vector-gratis/colección-iconos-moviles_1132789.htm#page=2&query=pack+icons&position=20
- Freepik. (s.f.). Speedometer free icon. [Icono]. https://www.flaticon.com/free-icon/speedometer_1335747?term=internet%20velocity&page=1&position=14



Derechos de imágenes

- Freepik. (s.f.). Warning free icon. [Icono]. https://www.flaticon.com/free-icon/warning_595067
- Freepik. (s.f.). Network free icon. [Icono]. https://www.flaticon.com/free-icon/network_2824836
- Freepik. (s.f.). Compliance free icon. [Icono]. https://www.flaticon.com/free-icon/compliance_4474330
- Freepik. (s.f.). Help free icon. [Icono]. https://www.flaticon.com/free-icon/help_1598645
- ---
- Flaticon. (s.f.). Key free icon. [Icono]. https://www.flaticon.com/free-icon/key_3170748?related_id=3170748
- Flaticon. (s.f.). Pattern Recognition free icon. [Icono]. https://www.flaticon.com/free-icon/pattern-recognition_3344359
- Flaticon. (s.f.). Pattern Recognition free icon. [Icono]. https://www.flaticon.com/free-icon/bars-graphic_559952
- Flaticon. (s.f.). Molecule free icon. [Icono]. https://www.flaticon.com/free-icon/molecule_4542801
- Flaticon. (s.f.). Data Flow free icon. [Icono]. https://www.flaticon.com/free-icon/data-flow_1953319



Créditos

Facultad de
INGENIERÍA

Autores

Jorge Eliecer Camargo Mendoza, PhD

Asistente docente

Juan Sebastián Lara Ramírez

Edder Hernández Forero

Brian Chaparro Cetina

Rosa Alejandra Superlano Esquibel

Leonardo Avendaño Rocha

Alberto Nicolai Romero Martínez

Diseño instruccional

Claudia Patricia Rodríguez Sánchez

Diseño gráfico

Clara Valeria Suárez Caballero

Milton R. Pachón Pinzón

Rosa Alejandra Superlano Esquibel

Brian Chaparro Cetina

Diagramadora PPT

Daniela Duque

Diseño de imágenes

Rosa Alejandra Superlano Esquibel

Fecha

2022-II

