

# Bases de datos no relacionales (Teórico)



# Introducción

Bienvenidos al módulo de bases de datos no relacionales. En este módulo abordaremos una introducción a las bases de datos no relacionales, también mencionadas como Bases de datos No SQL, y sobre todo nos enfocaremos, a bases de datos de tipo clave – valor, como MongoDB y DynamoDB

Una vez finalizado este módulo serás capaz de:

- Conocer los fundamentos de Bases de datos no SQL.
- Realizar la Instalación de Mongo Compass y su Configuración.
- Acceder a una Bases de datos No SQL.
- Conocer el concepto de Colecciones.
- Conocer los Fundamentos básicos de DynamoDB



# Tema 1. Bases de datos no SQL

## Objetivo

El objetivo de este tema es brindar una descripción básica de la de las bases de datos no SQL. Una vez avanzado en el desarrollo de los conceptos principales, abordaremos las bases de datos documentales, como MongoDB.

Una vez finalizado este tema serás capaz de:

- Conocer los conceptos de las bases de datos relacionales
- Conocer diferentes SGBD, pagos y gratuitos

## Conceptos fundamentales de las BD Relacionales

Una base de datos es un conjunto de datos relacionados entre si. Por datos entendemos hechos conocidos que pueden registrarse y que tienen un significado implícito. Por ejemplo, un número de teléfono, un número de documento, un nombre, etc.

Una base de datos tiene las siguientes propiedades implícitas:

Una base de datos representa algún aspecto del mundo real.

Una base de datos es un conjunto de datos lógicamente coherente, con cierto significado inherente. Una colección aleatoria de datos no puede considerarse propiamente una base de datos.

Toda base de datos se diseña, construye y prueba con datos para un propósito específico. Está dirigida a un grupo de usuarios y tiene ciertas aplicaciones preconcebidas que interesan a dichos usuarios.

Tomemos el conjunto de toda la información que es relevante a la operación de una organización: para un banco, será información relativa a cuentas, clientes, empleados, equipos, etc. Toda esa información, apropiadamente organizada y codificada se colocará en la base de datos de la organización. Ningún programa de aplicación tendrá acceso directo a los archivos que componen la base de datos, sino que interpondremos entre estos archivos y los programas de

aplicación un nuevo nivel de software, el Sistema de Gestión de Base de Datos (SGBD). El SGBD provee acceso a la información a un alto nivel de abstracción: en lugar de manipular archivos, registros, índices, cilindros, el programa de aplicación se maneja enteramente en términos de clientes, cuentas, saldos, etc., que son traducidos por el SGBD a su implementación física.

Un sistema de gestión de base de datos es un conjunto de programas que permite a los usuarios crear y mantener una base de datos. Por lo tanto, un SGBD es un software de propósito general que facilita el proceso de definir, construir y manipular bases de datos para diversas aplicaciones.

- Definir la base de datos consiste en especificar los tipos de datos, las estructuras y las restricciones de los datos que se almacenarán en ella.
- El proceso de construir una base de datos consta de la carga de los datos mismos en algún medio de almacenamiento controlado por un SGBD.
- En la manipulación de una base de datos intervienen funciones como consultar la base de datos para obtener datos específicos y/o actualizarlos.

## Ejemplos de SGBD

Existen en el mercado, muchos gestores de bases de datos, siendo unos destacados en un punto y otros en otro. Para este caso, los agrupamos por si son libres o no, omitiendo por el momento nuestra preferencia o nuestro análisis subjetivo sobre cada uno de los gestores aquí propuestos. Se resaltan solamente algunos de ellos, a saber:

### SGBD Libres

- PostgreSQL
- MySQL
- SQLite
- Firebird

- DB2 ExpressC
- MariaDB

### **SGBD Pagos**

- dBase
- Fox Pro
- IBM DB2
- IBM Informix
- Microsoft SQL Server
- Oracle
- Sybase



## **Tema 2. Bases de datos No SQL**

### **Objetivo**

El objetivo de este tema es brindar una descripción básica del concepto de las bases de datos No SQL, abordando los conceptos de los diferentes tipos de bases de datos No relacionales.

Una vez finalizado este tema serás capaz de:

- Conocer los conceptos de las bases de datos relacionales
- Conocer diferentes SGBD, pagos y gratuitos

El término NoSQL se utiliza para indicar las bases de datos que no cumplen con la estructura tradicional de una base de datos transaccional, que no logran cubrir las normas ACID ( Atomicidad, Consistencia, Aislamiento-*"Isolation"*-, Durabilidad) y que fueron ideadas para la manipulación de grandes volúmenes de datos, tanto para lectura como para escritura.

Hoy en día, muchas bases de datos no transaccionales comparten la organización de los datos con bases de datos que sean transaccionales, permitiendo interactuar compartiendo información.

Las bases de datos NoSQL son pensadas como distribuidas, administrando grandes cantidades de documentos, indexando de los mismos, generando estructuras de 5, 20, 50, y más terabytes de información. La forma de manipular este gran conjunto de información se lleva a cabo a través de la distribución de la información en distintos servidores y luego componentes hash (por ejemplo, tablas hash) que permiten identificar dónde se alojó la información. Esto permite a su vez, duplicar información en diferentes servidores, a modo de redundancia, generando una alta tolerancia a fallas y además una capacidad de resolver consultas más rápidamente.

La gestión de las transacciones suele realizarse a través de procesos intermedios, que se ejecutan a la par de la colección de información, casi de modo independiente, realizando así una mejoría en la velocidad de grabación de información. Estos procesos (o middleware) intentan asegurar la calidad de las transacciones tal cual realiza un motor transaccional con las reglas ACID. La forma de acceder a la información depende del motor de base de datos a utilizar. Algunos de los cuales tienden a sugerir que XQuery puede ser la mejor interfaz para acceder a la información, mientras que otros apuntan hacia vectores de información o a diccionarios.

## **Tipos de bases de datos NoSQL**

Dentro de las bases de datos NoSQL, se encuentran distintas corrientes que han generado diferentes motores de bases de datos que se pueden utilizar según la necesidad de cada aplicación o construcción de información.

### **● Orientadas a Documentos**

La mayoría de las bases de datos NoSQL están dentro de este subconjunto. Las bases de datos orientadas a documentos, se definen como un programa que sirve para guardar, buscar y restaurar archivos en base a un conjunto de información semiestructurada. Esto

quiere decir, que de los distintos documentos a almacenar, se podrán tener cierta información común para todos ellos, mientras que para algunos tendremos información parcial sobre temas que solo involucran a alguno de los documentos. A diferencia de las bases de datos transaccionales donde hay relaciones entre distintos conjuntos, aquí solo existen conjuntos de documentos y diferenciados según la información que se almacene de cada uno de ellos. Todo dentro de este tipo de bases de datos gira en torno a los documentos. Por ejemplo el tipo de dichos documentos. Pueden ser documentos tipo XML, YAML, JSON, PDF, Word, etc., es decir, tipo texto con un formato definido en el interior del documento, o tener una estructura binaria. De cada documento, como decíamos anteriormente, se guarda solo la información que se pudo coleccionar del mismo.

Por ejemplo, de cuatro documentos diferentes pertenecientes a la misma colección, podemos tener:

#### Documento 1

```
{
  _id: 1
  Nombre: "Juan",
  Apellido: "Lopez",
  Usuario: "jlopez"
}
```

#### Documento 2

```
{
  _id: 2
  Nombre: "Ana",
  Apellido: "Garcia",
  Hermanos:
    [
      { Nombre: "Diego" }
      { Nombre: "Luis" FNacimiento: "01/02/1980" }
      { Nombre: "Maria" Edad: "32" }
    ]
}
```

#### Documento 3

```
{
  _id: 3
  Nombre: "Ana",
  Apellido: "Rodriguez",
  Hermanos:
    [
      { Nombre: "Diego" }
    ]
}
```

#### Documento 4

```
{
  _id : 4,
  Nombre: "Diego",
  Apellido: "Perez",
  Colores:
    [ "azul", "blanco" ]
}
```

Como vemos, cada documento contiene una estructura que comparte con otro, mientras que puede contener información única para el documento o que comparta con un subconjunto del total de documentos. Si esto fuera planteado como una base de datos transaccional, descubriremos que para poder almacenar esta información debemos mantener muchas columnas que estarán parcialmente vacías, y por cada documento que encontremos con una nueva información, deberíamos generar una nueva columna que contenga ese dato, mientras que para el resto del conjunto esa columna estará vacía.

### Clave

Cada documento, estará indexado bajo una clave, lo cual permitirá encontrarlo rápidamente. Esta clave puede ser la dirección física del archivo, como también una URL o un Path relativo. También puede ser un valor numérico. Lo importante es que para cada documento dentro de una colección, se tenga una clave y esta clave sea única para todo el conjunto de datos. Para los ejemplos de documentos anteriores hemos definido un elemento como clave única el atributo “\_id”.

### Organización



Los archivos se organizarán bajo etiquetas (o Tags), colecciones, metadata, o estructuras a modo de árbol, que permita indexar la información de los documentos. La organización será definida por quien implementa el motor de datos, utilizando estas estructuras u otras que crea conveniente para la localización de la información. Cada elemento, se conocerá como par *Clave/Valor*, porque tendrá la clave definida por la etiqueta y el valor asociado. Por ejemplo Nombre: “ana”.

Este par, llamado Clave/Valor, no debe confundirse con el identificador único del documento. Este identificador único debe estar presente en cada documento y se deberá llamar igual en todos. También estará formado por un par Clave/Valor. Por ejemplo el código: “1”.

## Motores de bases de datos NoSQL / Documentales

A continuación, listamos algunos de los motores de bases de datos NoSQL que administran su información de forma documental:

- MongoDB
- CouchDB
- CouchBase

### Lenguaje de programación

La mayoría de bases de datos documentales utilizan programación orientada a objetos, mientras que la información se puede consumir con JSON.

Ejemplo:

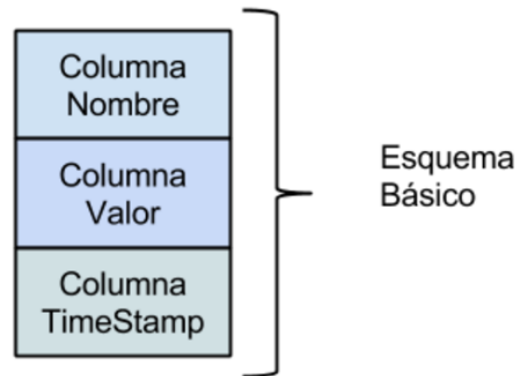
```
db.createCollection("miColeccion");
```

### Comparativa con una base de datos Relacional

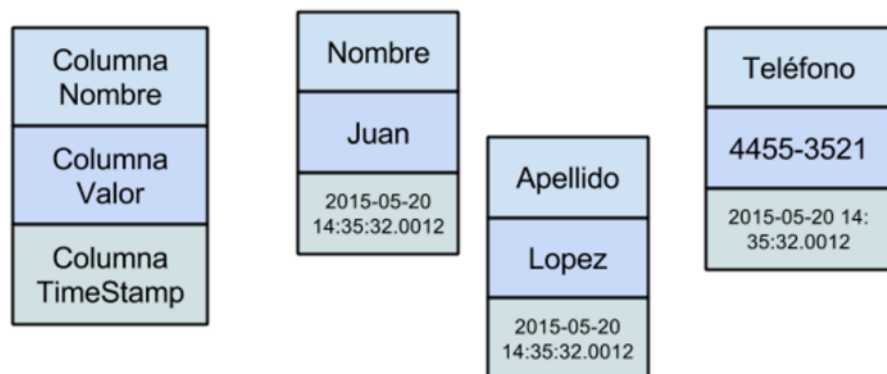
Modelo Relacional	Modelo Orientado a Documentos
Tablas	Colecciones
Filas	Documentos
Columnas	par armado por clave/valor
Juntas	No existe

- **Orientadas a Columnas**

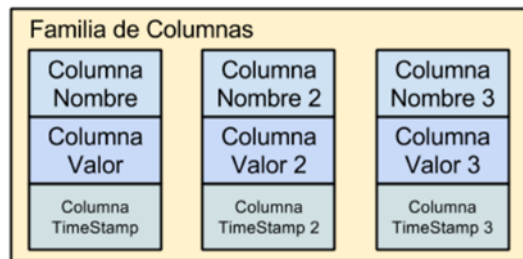
Este tipo de base de datos crea columnas y familias de columnas. Cada columna, cuenta con una clave y un valor. No existe el concepto de tablas, pero si el concepto de familia de columnas. Por cada dato que queramos guardar, vamos a tener que indicar el nombre de la columna y el del valor. Cabe señalar, que internamente se almacena la fecha en que se ingresa esta información.



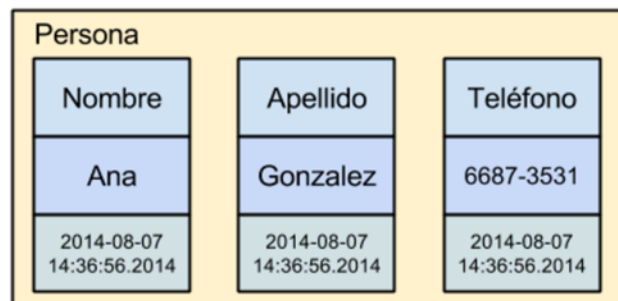
Esto hace que en cada elemento de información conste el nombre de la columna, su valor y la fecha y hora que se generó. A modo de diccionario. Veamos algunos ejemplos.



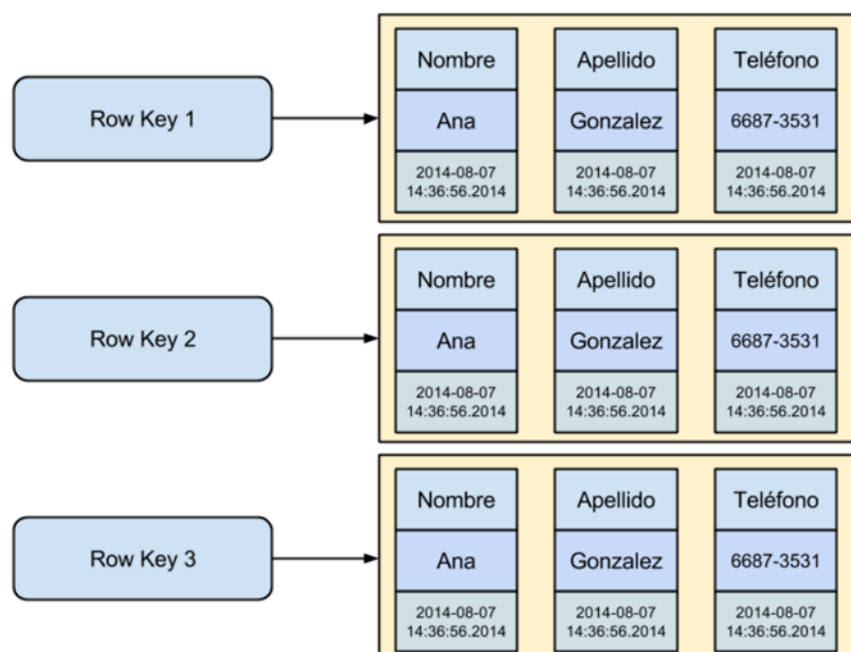
Como decíamos, un conjunto de columna valor genera una familia de columnas. Una familia es un concepto similar a las tuplas o filas en una base de datos relacional.



Veamos un ejemplo de una familia de Columnas:



En algunas documentaciones las familias de columnas se definen como súper columnas. Pero manteniendo el mismo concepto, agrupar columnas que forman parte de un todo. A cada familia se le asigna una clave que la identifique, permitiendo así diferenciar una familia de otra.

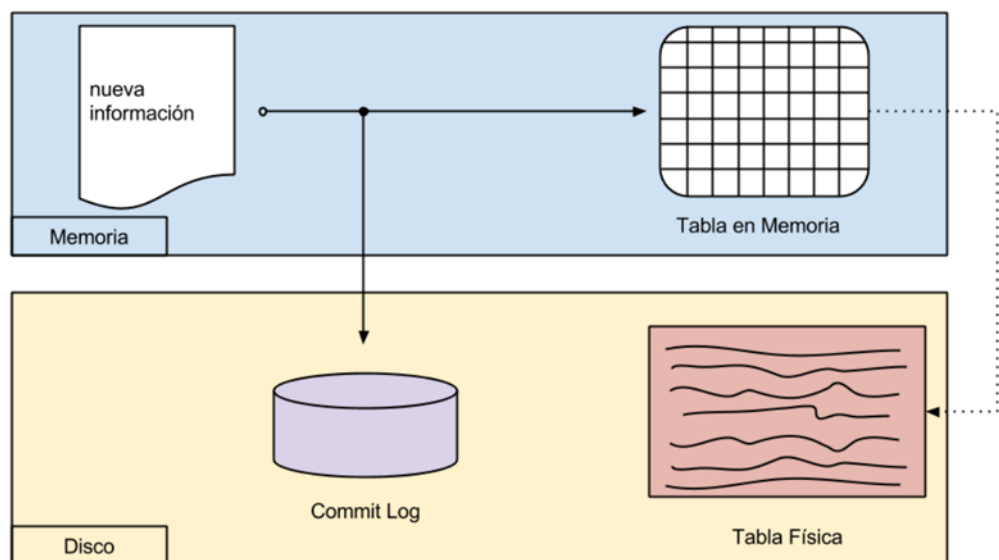


## Clave

El concepto “clave” es cómo tal cual existe y se piensa en las bases de datos transaccionales. Es decir, un valor único que identifica al resto de información y sirve para diferenciar un elemento de otro.

## Organización

Dentro de estas bases de datos se prioriza el almacenar lo más rápido la información, desestimando las reglas ACID. Para esto lo que se realiza es almacenar en una tabla en memoria la nueva información, mientras que en paralelo se almacena en un *log* de operaciones. Este almacenaje en un *log* hace perdurable en el tiempo las distintas operaciones. Por otro lado, las tablas en memoria se vacían periódicamente en tablas a disco.



## Motores de bases de datos NoSQL / Orientados a Columnas

Algunos de los motores de bases de datos que utilizan este formato para almacenar su información son:

- Apache Cassandra
- Amazon Redshift
- Apache HBase

## Lenguaje de programación

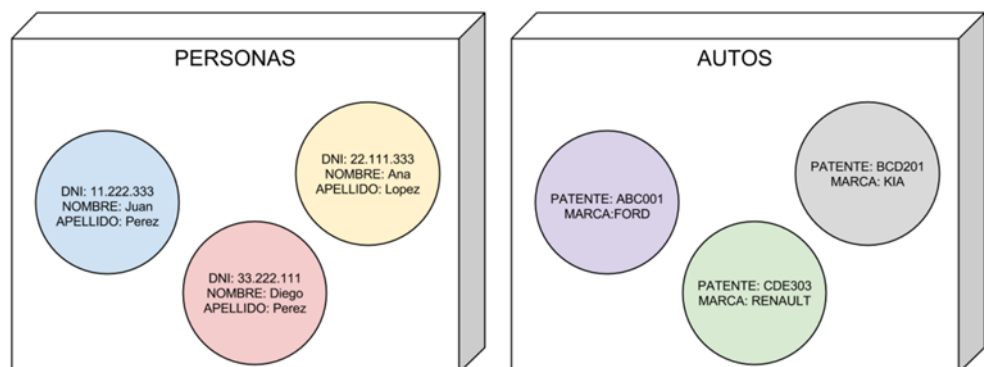
El lenguaje utilizado en este tipo de bases de datos suele ser un SQL modificado o adaptado a cada motor, por ejemplo CQL para Casandra.

### Comparativa con una base de datos Relacional

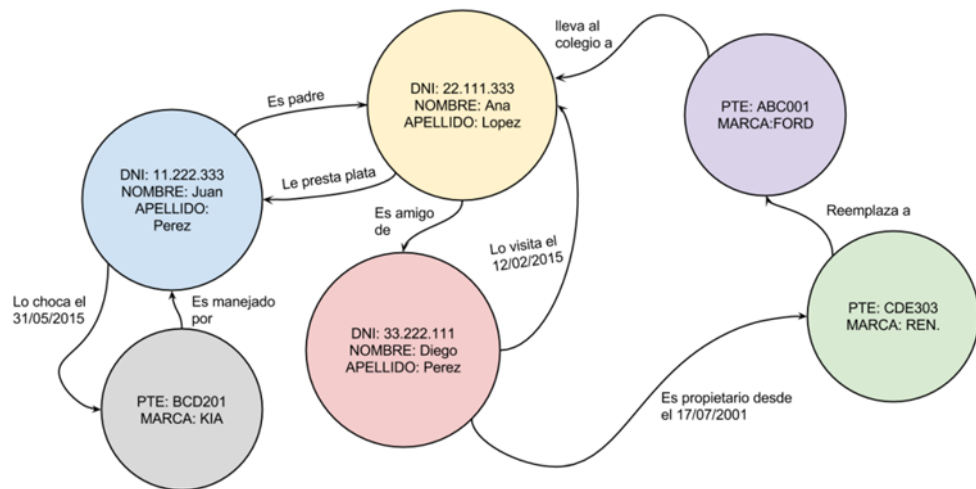
Modelo Relacional	Modelo Orientado a Columnas
Tablas	Conjunto de familias de columnas
Filas	Familia de columnas
Columnas	Nombres de columnas de una familia
Juntas	no aplica

### • Orientadas a Grafos

Las bases de datos orientadas a grafos nacen a partir de la necesidad de vincular elementos con ciertas características mediante diferentes tipos de relaciones. Cada elemento de este tipo de base de datos puede pensarse como una tupla dentro de un modelo relacional, y cada vínculo como una relación entre dos tuplas cualesquiera, sean del mismo conjunto como de otro. Cada nodo o elemento, va a pertenecer a una clase que lo agrupa y define meta características básicas para los elementos de su tipo.



Luego, entre estos nodos y según transcurra la actividad entre ellos a través del sistema, habrá diferentes conexiones que generarán información.



Es importante destacar que este tipo de bases de datos se utilizan en las webs semánticas, tales como aquellas utilizadas por las policías, como por grandes organizaciones que necesitan diversos cruces de información, como las agencias espaciales.

El mapeo con la programación orientada a objetos es directo y además proporciona un gran dinamismo a la hora de escalar el sistema.

### Clave

Para formalizar la clave, debe uno situarse en la metadata y definir cuál atributo o conjunto de atributos serán quien determinen la forma unívoca de identificar un nodo de otro. Por otro lado, es preciso aclarar que las conexiones entre nodos pueden ser unidireccionales, como bidireccionales. Además que cada una de estas conexiones contará con un tipo, un nombre y una fecha de creación.

### Organización

Este tipo de bases de datos se almacenan en archivos con configuración propietaria de cada uno de los diseñadores, pero dentro del marco teórico podemos decir que se genera un set de estructuras (struct's) que se almacenan en disco con punteros a otras estructuras, definiendo así las conexiones entre los nodos.



## Tema 3. Mongo DB

### Objetivo

El objetivo de este tema es brindar una introducción a MongoDB y a su sistema de gestión visual, llamada Compass.

Una vez finalizado este tema serás capaz de:

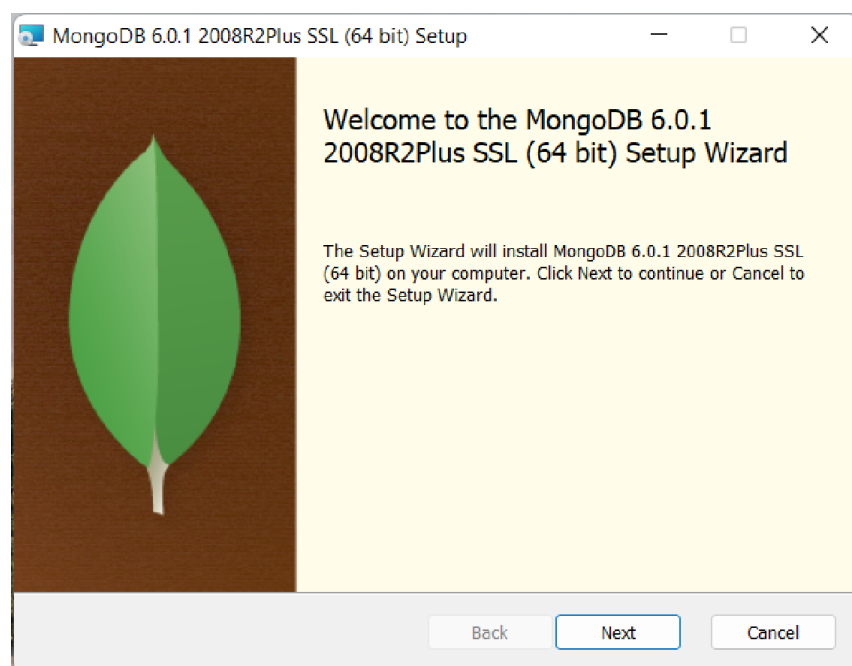
- Conocer los conceptos fundamentales de MongoDB
- Instalar MongoDB localmente y Compass
- Acceder al motor, crear una base de datos y una colección.
- Consultar datos de la colección.

### Instalación de MongoDB

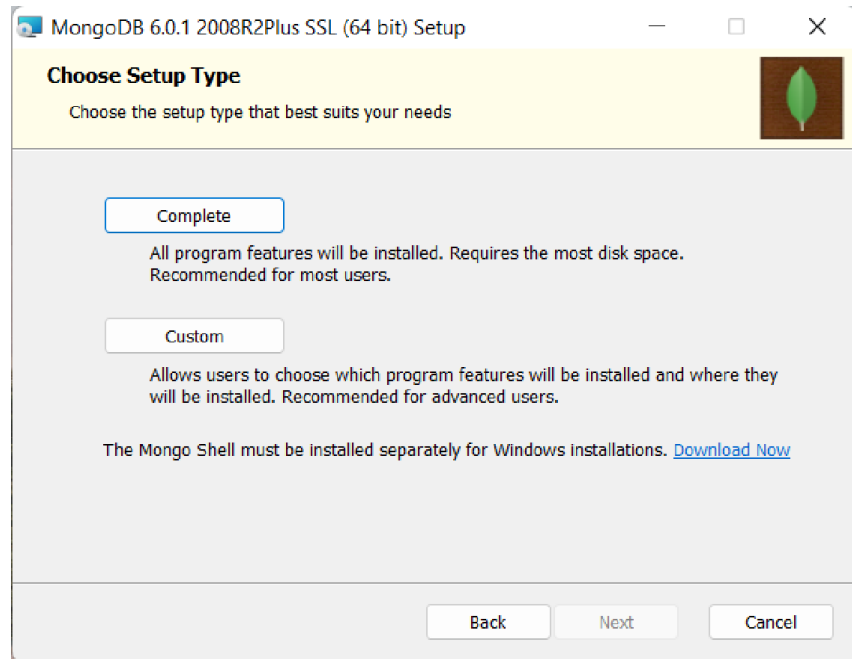
Desde esta URL comenzaremos con la descarga:  
<https://www.mongodb.com/try/download/community>

Una vez descargado el mismo, ejecutamos el instalador para comenzar con la instalación del motor de MongoDB.

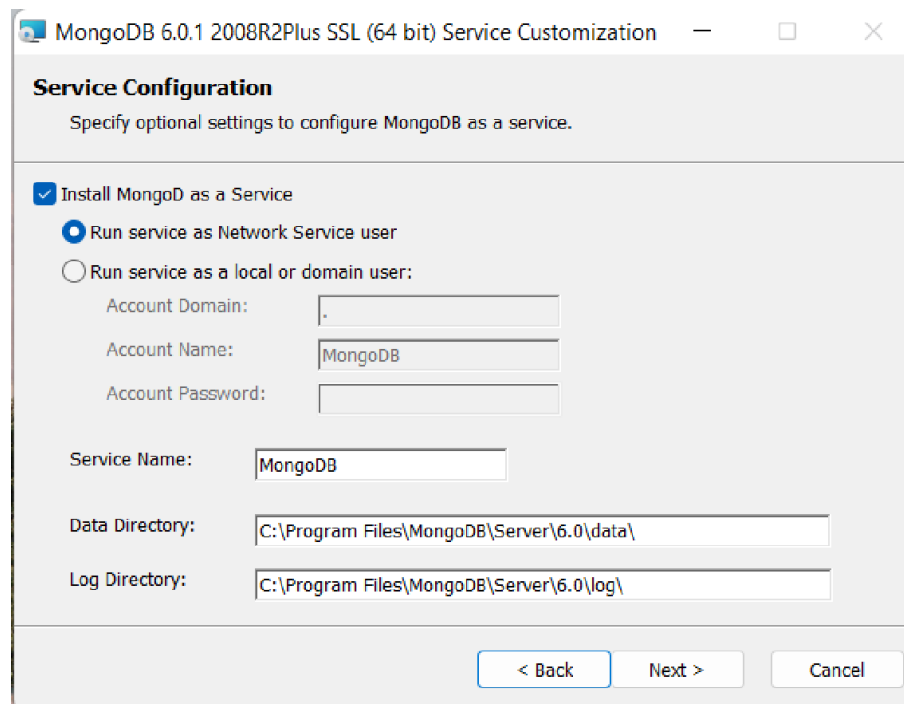
El sistema nos mostrará la siguiente pantalla:



Presionamos “Next” y en la pantalla siguiente aceptamos los términos de licenciamiento. Luego, aparecerá la siguiente pantalla:



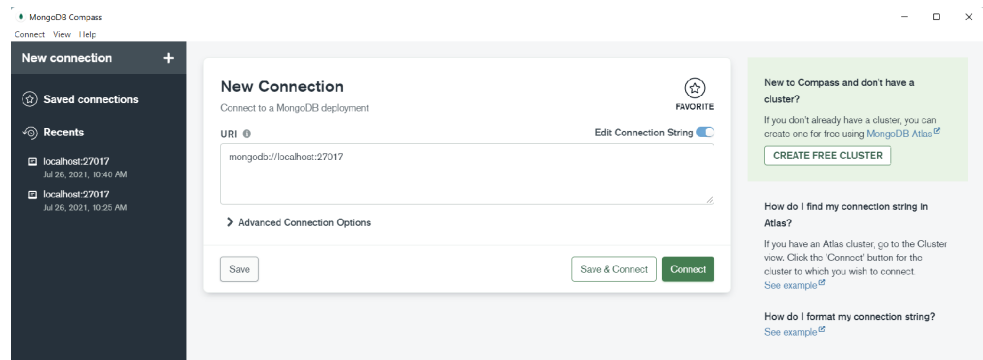
Seleccionamos la opción “Complete”. Luego, aparecerá la siguiente pantalla:



Dejamos las opciones por defecto. Instalar Mongo como servicio y los directorios para los datos y logs. Presionamos “Next”. El sistema comenzará con la instalación. Para finalizar, puede que se necesite



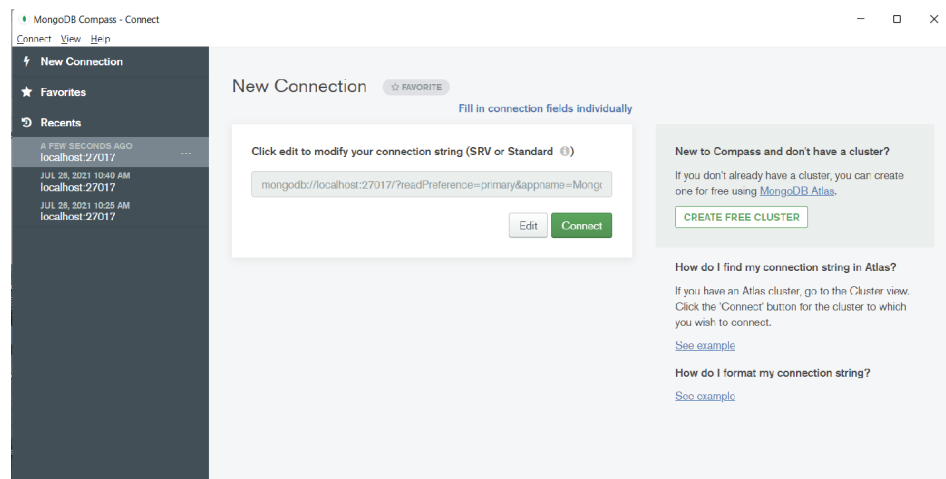
reiniciar el equipo. Al finalizar la instalación, aparecerá la siguiente pantalla:



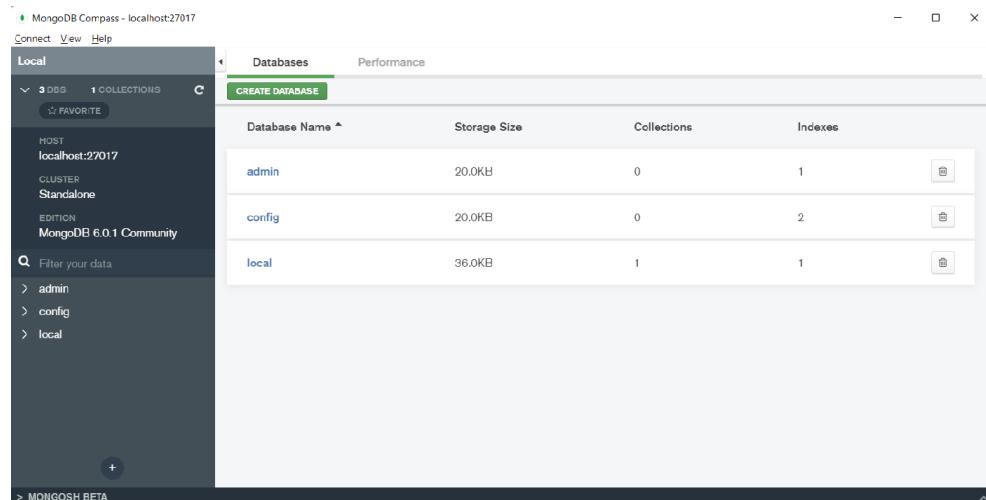
La pantalla anterior pertenece a Compass. Esto nos indica que MongoDB fue correctamente instalado.

## Conexión a MongoDB

Desde la siguiente pantalla nos conectaremos al motor:



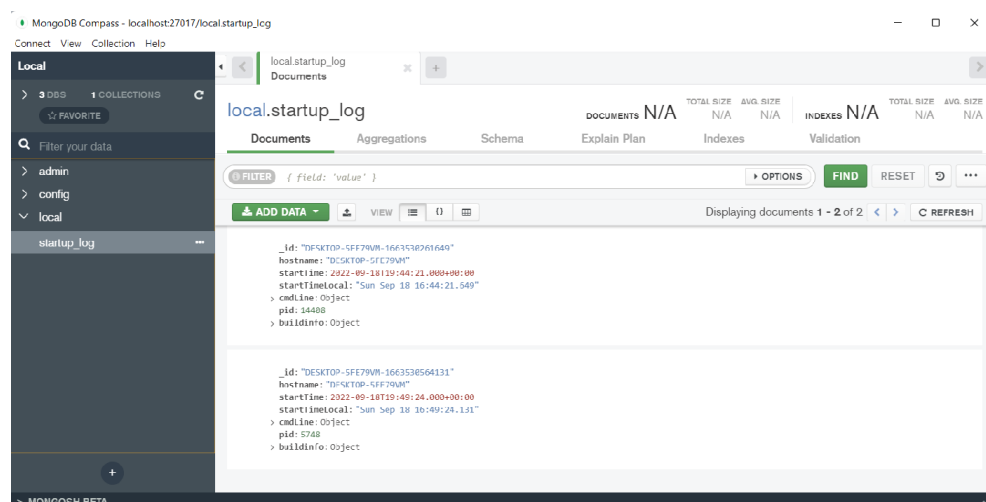
Podríamos ingresar la URL de la base de datos a conectarnos. En este caso no ingresaremos nada, ya que la instalación es local. Presionamos “Connect” y nos aparecerá la siguiente pantalla:



Ya dentro de Compass veremos:

En el centro de la pantalla, las Bases de datos existentes. Con la posibilidad de crear una nueva a través del botón “Create Database”.

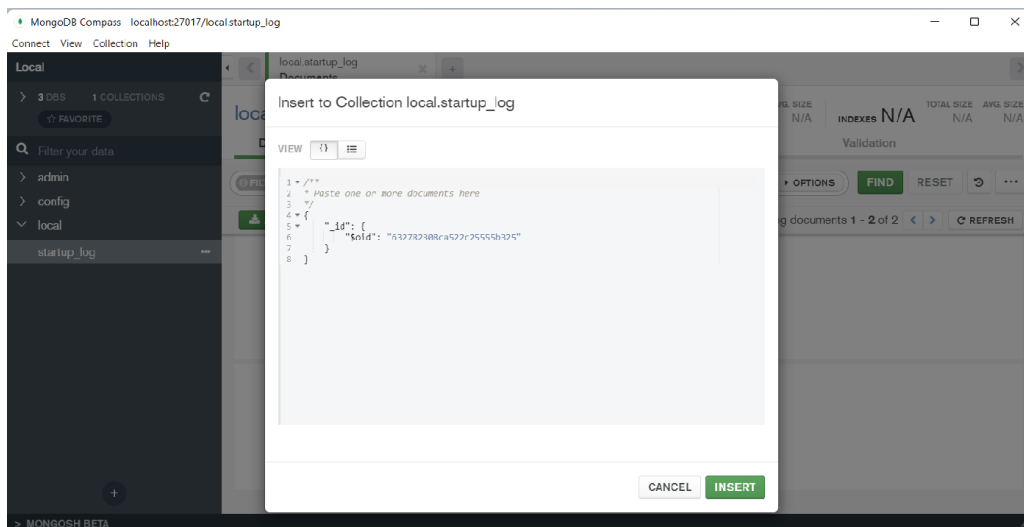
En el menú de la izquierda, se listaran las bases de datos, con la posibilidad de abrir las misma para poder visualizar el listado de colecciones existentes (Recordemos que una colección es lo que una tabla es a una base de datos relacional). Desde las colecciones, podremos acceder a los documentos existentes dentro de ella. La siguiente pantalla muestra como se visualizan los documentos:



En esta pantalla, dentro de Filters, podríamos aplicar cualquier condición que permita reducir los documentos listados, a través de un criterio de búsqueda.

Por otro lado, también podríamos insertar un documento a esta colección. Presionando el botón “Add Data”, se listan las siguientes opciones:

- Import File: Para ingresar un documento a través de un archivo json
- Insert document: El cual nos desplegara la siguiente pantalla:



Como vemos, desde este editor podremos generar nuestro JSON, el cual se ingresará como documento a nuestra colección.

### Conexión a MongoDB desde PowerShell (PS)

Si no contamos con Compass, podríamos interactuar con el motor a través de los siguientes comandos.

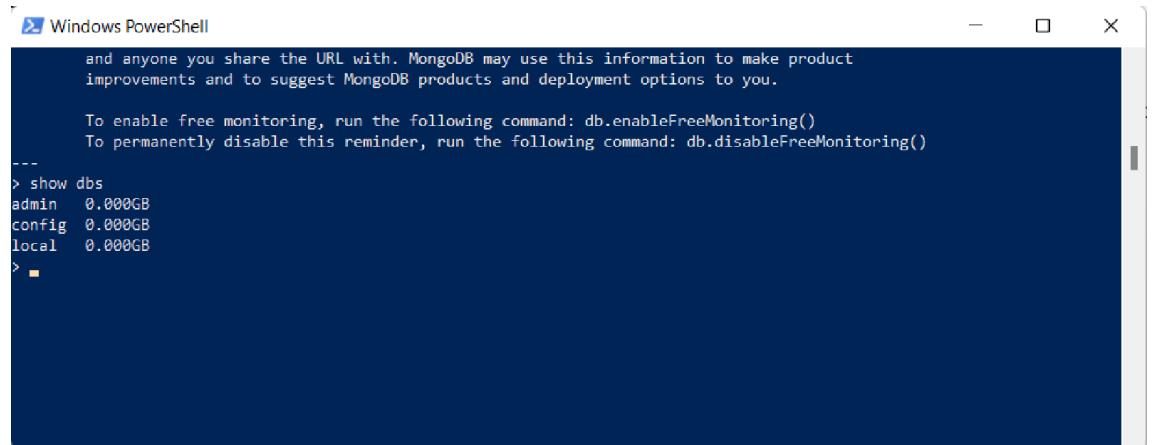
Desde la carpeta BIN que contiene los binarios de mongo, podríamos conectarnos al motor local con el siguiente comando:

***PS C:\Program Files\MongoDB\Server\5.0\bin> .\mongo.exe 27017***

Ya conectados desde PS, podremos consultar las Bases existentes con este comando:

***show dbs***

PS nos mostraria los siguiente:



```

Windows PowerShell

and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()

---
> show dbs
admin    0.000GB
config  0.000GB
local   0.000GB
>

```

Como vemos, son las mismas bases que vimos desde Compass, ya que estamos conectados al mismo motor.

Ahora bien, veamos una serie de comandos básicos que podemos utilizar conectados al motor desde la consola, y ya no desde compass.

### Crear base de datos

use basemongo

**Nota.** Este comando creará la base “basemongo” pero cabe mencionar que podremos ver la base creada cuando asignemos una colección a la misma.

**Para chequear que estamos en la base creada en el paso anterior :**

db

Este comando nos muestra la base sobre la que estamos posicionados.

**Mostrar todas las funciones que podemos hacer a la base de datos en uso**

db.help()

### Crear la colección cliente

db.createCollection("Cliente")

### Eliminar la Colección Cliente

db.Cliente.drop()

## Mostrar las colecciones existentes

show collections

## Insertar un solo documento en la colección Cliente

```
db.Cliente.insertOne({_id:1,Descripcion:'Razón Social
1',CiudadCliente:'Mendoza'})
```

## Insertar más de un documento en la colección Cliente.

```
db.Cliente.insertMany([{_id:2,Descripcion:'Razón Social
1',CiudadCliente:'Mendoza'},{_id:3,Descripcion:'Razón Social
2',CiudadCliente:Cordoba}])
```

## Filtrar según operadores

- **El Operador \$eq Encuentra los valores que son iguales al valor de búsqueda**

Veamos un Ejemplo

```
db.Cliente.find({_id:{$eq:1}})
```

Este comando listará aquellos clientes cuyo campo “\_id” sea igual a 1.

- **El Operador \$gt encuentra los valores que son mayores al valor de búsqueda**

Veamos un Ejemplo

```
db.Cliente.find({_id:{$gt:1}})
```

Este comando listará aquellos clientes cuyo campo “\_id” sea mayor al valor 1.

- **El operador \$gte encuentra los valores que son mayores o iguales al valor de búsqueda**

Veamos un Ejemplo

```
db.Cliente.find({_id:{$gte:1}})
```

Este comando listará aquellos clientes cuyo campo “\_id” sea mayor o igual al valor 1.

- **El operador \$in encuentra cualquiera de los valores que coinciden con los elementos del array de búsqueda.**

Veamos un ejemplo

```
db.Cliente.find({_id:{$in:[2,3]} })
```

Este comando listará aquellos clientes cuyo campo “\_id” sea igual a 2 o 3.

- **El operador \$lt encuentra los valores que son menores al valor de búsqueda**

Veamos un ejemplo

```
db.Cliente.find({_id:{$lt:3} })
```

Este comando listará aquellos clientes cuyo campo “\_id” sea menor a 3

- **El operador \$lte encuentra los valores que son menores o iguales al valor de búsqueda.**

Veamos un Ejemplo

```
db.Cliente.find({_id:{$lte:3} })
```

Este comando listara aquellos clientes cuyo campo “\_id” sea menor o igual a 3

- **El operador \$ne encuentra los valores que no son iguales al valor de búsqueda.**

Veamos un Ejemplo

```
db.Cliente.find({_id:{$ne:[3]} })
```

Este comando listará aquellos clientes cuyo campo “\_id” sean distintos a 3

- **El operador \$nin encuentra cualquiera de los valores que no coinciden con los elementos del array de búsqueda.**

Veamos un Ejemplo

```
db.Cliente.find({_id:{$nin:[2,3]} })
```

Este comando listará aquellos clientes cuyo campo “\_id” no coincida con los valores 2 o 3.

### **Buscar Clientes con filtro:**

```
db.Cliente.find({_id:1})
```

Este comando buscará el cliente cuyo campo \_id es igual a 1

```
db.Cliente.find({CiudadCliente:'Mendoza'})
```

Este comando buscará aquellos clientes que cumplan la condición que su ciudad sea igual a “Mendoza”

```
db.Cliente.find({CiudadCliente:'Mendoza'}).forEach(printjson)
```

Este comando buscará aquellos clientes que cumplan la condición que su ciudad sea igual a “Mendoza” e imprimirá el resultado en formato JSON.

### **Contar la cantidad de documentos de una coleccion:**

```
db.Cliente.count()
```

### **Eliminar un Documento**

```
db.Cliente.deleteOne({_id:2})
```

### **Eliminar aquellos documentos que cumplen un criterio**

```
db.Cliente.deleteMany({CiudadCliente:'Mendoza'})
```

### **Realizar un Update a un solo registro**

```
db.Cliente.updateOne({_id:2},
    {$set:{Descripcion:'Nueva Razón Social'}}
)
```

### **Realizar un Update a varios registros**

```
db.Cliente.updateMany({_id:{$lte:1}},
    {$set:{Descripcion:'ArtActualizado'}}
)
```

Para finalizar este tema, hemos visto cómo acceder a MongoDB desde una herramienta visual y trabajar básicamente con colecciones y documentos. Por otro lado, dejamos de lado la herramienta para poder conocer aquellos comandos básicos que nos permiten trabajar con mongodb desde una consola y crear una base de datos, colecciones y documentos.





# Tema 3. Conceptos de Dynamo DB

## Objetivo

El objetivo de este tema es brindar una la herramienta DynamoDB de Amazon.

Una vez finalizado este tema serás capaz de:

- Conocer qué es Dynamo DB
- Conocer los conceptos básicos de este motor
- Conocer sus casos de uso

## ¿Qué es DynamoDB?

DynamoDB es un servicio de base de datos dentro de AWS (Amazon Web Services) con la diferencia de que no entra en el tipo de base de datos relacional o SQL. Se basa en el sistema de almacenamiento clave-valor, donde se nos permite crear tablas de alta escalabilidad y durabilidad, con DynamoDB no tiene que preocuparse de la configuración de hardware como lo sería con una tabla relacional, las cuales requieren la implementación de una instancia de servicio de computo.

Está optimizada para tener un buen rendimiento y escalabilidad horizontal añadiendo más nodos al clúster. Si el tamaño de los datos crece, no se va a penalizar en el rendimiento de la base de datos de una manera muy significativa.

Además, tiene alta disponibilidad (99.999% de uptime SLA, esto es menos de 5 minutos al año de caídas) y durabilidad de los datos. Para ello, almacena varias copias de los datos en diferentes nodos. Si uno de estos nodos falla, otro puede tomar su lugar como nodo primario.

DynamoDB es un servicio gestionado, y aunque hay servidores y sistemas de almacenamiento por detrás, es totalmente transparente y no debemos preocuparnos de la gestión del hardware o la seguridad.

DynamoDB distribuye los datos en particiones, que pueden estar en distintas máquinas, tanto físicas como virtuales para aumentar la redundancia.

## Conceptos Básicos

En DynamoDB las **tablas** son las colecciones de elementos, y los elementos son colecciones de atributos o pares clave-valor. La clave primaria de una tabla está compuesta de una clave de partición y de una clave de clasificación (sort key).

También, se puede usar un índice secundario global o GSI, que permite realizar consultas filtrando por columnas que no sean la clave de partición o de clasificación.

## Casos de Uso

DynamoDB es ideal para aplicaciones que tienen patrones de acceso bien conocidos. Antes de usar este servicio es recomendable conocer cómo acceder a los datos para diseñar una estructura de tablas correcta desde el primer instante.

Para trabajar con las APIs de DynamoDB, al ser una base de datos clave-valor, se proporcionará generalmente la clave para obtener el valor asociado almacenado. Es común usar un ORMs (Object Relational Mapper) que por detrás implementan las APIs, dependiendo del lenguaje que se quiera usar. Para Java tenemos DynamoDBMapper y para Python Boto3.

## Cierre

A lo largo de este módulo repasamos los conceptos básicos de las bases de datos no relacionales, enfocándonos en MongoDB. En este sentido, aprendimos a realizar la Instalación del motor de MongoDB, Mongo Compass y su configuración. Aprendimos también el concepto de colecciones y documentos. Además, aprendimos a realizar consultas básicas sobre la base de datos. Por otro lado, aprendimos los fundamentos básicos de DynamoDB.

# Referencias

Introducción a las Bases de Datos. Autor: A. Palomares.

Introducción a las Bases de Datos No Transaccionales. Autor: A. Palomares.

<https://www.mongodb.com/es/what-is-mongodb>

[https://docs.aws.amazon.com/es\\_es/amazondynamodb/latest/developerguide/Introduction.html](https://docs.aws.amazon.com/es_es/amazondynamodb/latest/developerguide/Introduction.html)

<https://aprenderbigdata.com/dynamodb/>

tiiiiiit by 