Bases de datos no estructuradas

Unidad 1: Introducción a las bases de datos no estructuradas

Descargable

Índice

Introducción

En esta primera unidad, explorarás las características distintivas de las bases de datos NoSQL y cómo se diferencian de las tradicionales bases de datos estructuradas (SQL). También profundizarás en los elementos esenciales de las bases de datos documentales, con un enfoque especial en MongoDB, analizando sus características y beneficios para el manejo de datos no estructurados. A lo largo de esta unidad, se espera que diferencies las bases de datos no estructuradas y las bases de datos estructuradas a través del reconocimiento de sus características; asimismo, está diseñado para que inicies tu conocimiento en Mongo DB.

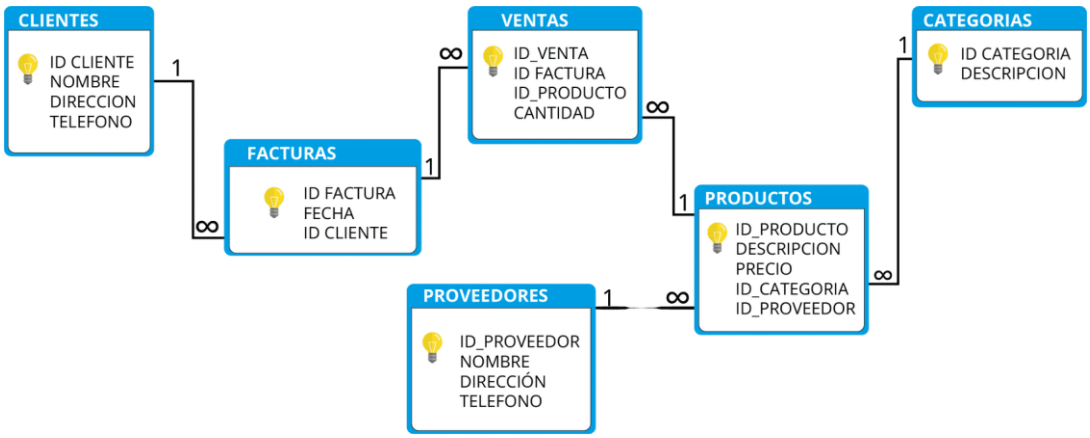
Bases de Datos No Estructuradas: Características y tipos

Características de las bases de datos no estructuradas y estructuradas

Una base de datos es un conjunto de datos de diferentes tipos organizados según las necesidades de una organización. Estas son otras definiciones fundamentales:

a) **Modelo de datos:** Es una representación gráfica del almacenamiento de información.

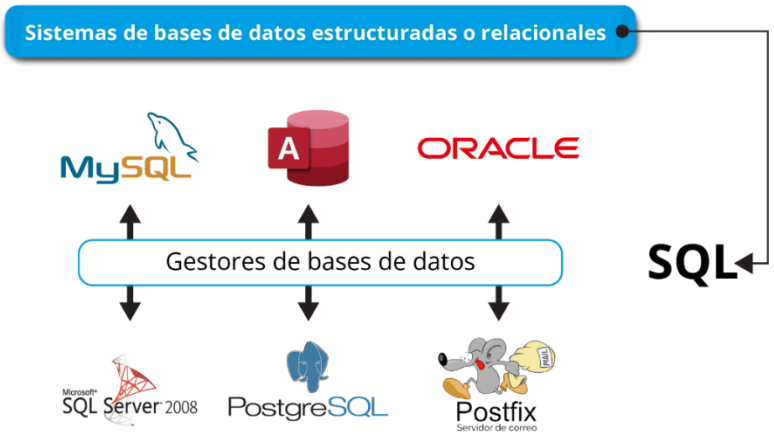
b) **Modelo de datos estructurado o relacional:** Es una representación gráfica que comprende relaciones entre tablas de diversa información donde las asociaciones están representadas entre claves primarias y foráneas.



c) **Sistemas gestores de base de datos estructuradas o relacionales:** Se trata de softwares de almacenamiento de información que se base en el modelo relacional y que tiene como lenguaje SQL.

Ejemplos de sistemas de bases de datos estructuradas o relacionales

Figura 1: Sistemas de bases de datos estructuradas o relacionales

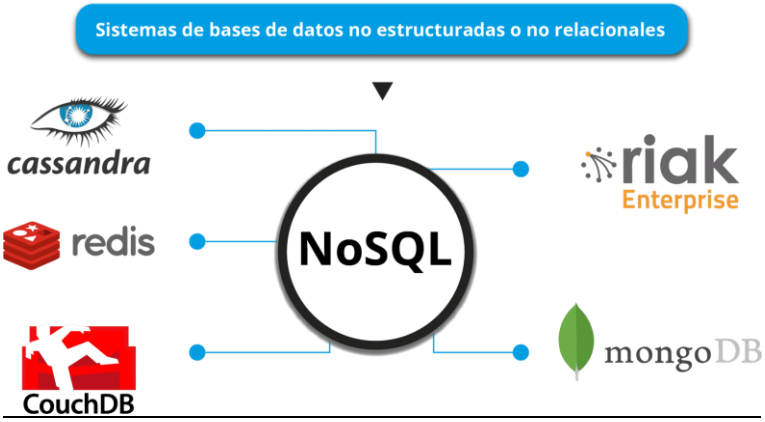


Fuente: Elaboración propia

d) Sistemas de **base de datos no estructuradas o no relacionales:** Son softwares de almacenamiento de información que no se basa en un modelo relacional y que tiene como lenguaje NoSQL.

Ejemplos de sistemas de bases de datos no estructuradas o no relacionales

Figura 2: Sistemas de bases de datos no estructuradas o no relacionales



Fuente: Paramio, C.

Características y propiedades de las bases de datos estructuradas o relacionales

Una base de datos relacional es un tipo de base de datos que almacena y proporciona acceso a puntos de datos relacionados entre sí. Se basan en el modelo relacional, una forma intuitiva y directa de representar datos en tablas.

Este tipo de base de datos presenta las siguientes características formales:

- En una base de datos relacional, cada fila en una tabla es un registro con una ID única, llamada clave.
- Las columnas de la tabla contienen los atributos de los datos
- Cada registro suele tener un valor para cada atributo, lo que simplifica la creación de relaciones entre los puntos de datos.
- El lenguaje usado en este tipo de base de datos es SQL.

Son cuatro las propiedades cruciales que definen las transacciones de las bases de datos relacionales: **atomicidad, consistencia, aislamiento y durabilidad.** Estas suelen conocerse como ACID.

Figura 3: Modelo ACID

Fuente: Escobedo, A. (2022, 18 octubre).

a) Atomicidad: La atomicidad define todos los elementos que conforman una transacción completa de base de datos.

b) Consistencia: Define las reglas para mantener los puntos de datos en un estado correcto después de una transacción.

c) Aislamiento: Impide que el efecto de una transacción sea visible a otros hasta que se establezca el compromiso, a fin de evitar confusiones.

d) Durabilidad: Garantiza que los cambios en los datos se vuelvan permanentes una vez que la transacción ha sido confirmada y se ha alcanzado un compromiso.

Ventajas

- Mayor soporte y más variedad de herramientas debido a que lleva más tiempo en el mercado.
- Es útil para manejar y obtener los datos.
- Permite agregar otros servidores de SQL, por ejemplo, una persona puede acceder a la base de datos de otra.

Desventajas

- No es flexible (antes de ingresar los objetos, deben estar correctamente validados).
- Mientras más compleja sea la base de datos, requiere mayor procesamiento y eso se puede ver reflejado en el rendimiento y consumo de recursos.

Características y propiedades de las bases de datos no estructuradas o no relacionales

Estas bases carecen de una estructura definida para el almacenamiento de información, lo que permite que las transacciones de datos sean significativamente más rápidas que en las bases de datos estructuradas.

Este tipo de base de datos presenta las siguientes características formales:

- No utilizan tablas ni columnas para almacenar sus datos: Los datos se almacenan en documentos.
- No utiliza SQL para la gestión de datos.
- Son base de datos escalables, pueden crecer en tamaño con el paso del tiempo.
- Son flexibles en cuando a los formatos de uso.
- El acceso es rápido debido al tener una estructura simple.
- Están diseñadas para almacenar grandes volúmenes de datos.
- En general, se tratan de proyectos de código abierto.
- Se ejecutan en clúster (serie de computadoras pequeñas y de gran procesamiento).

Se deben considerar los siguientes aspectos:

a) **En cuanto a su esquema:** Permite almacenar documentos con diferentes estructuras. Además, almacenar y consultar datos sin necesidad de realizar JOIN, utiliza Documentos embebidos.

b) **En cuanto a su escalabilidad:** Es horizontal, por lo que se pueden agregar equipos sin alterar el rendimiento.

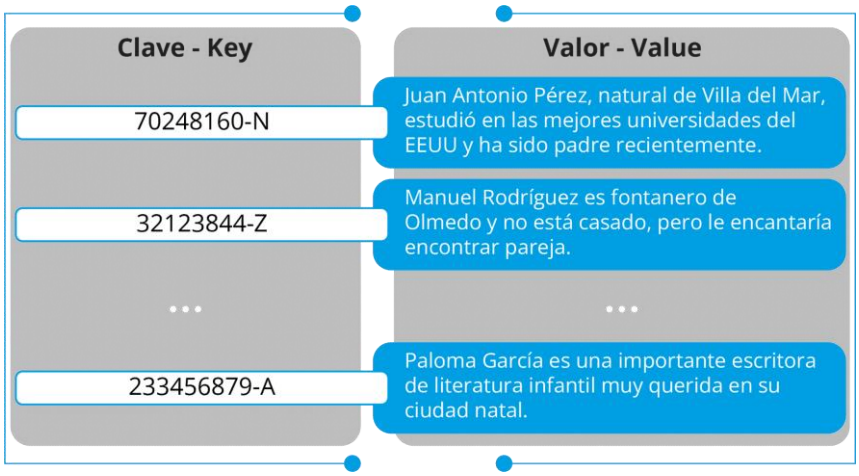
c) **En cuanto a la velocidad:** Su procesamiento de datos y retorno de valores es mucho más rápido que en las bases de datos estructuradas.

Tipos de base de datos no estructuradas

a) **Clave – Valor**

Almacena datos como un conjunto de pares clave-valor en los que una clave sirve como un identificador único.

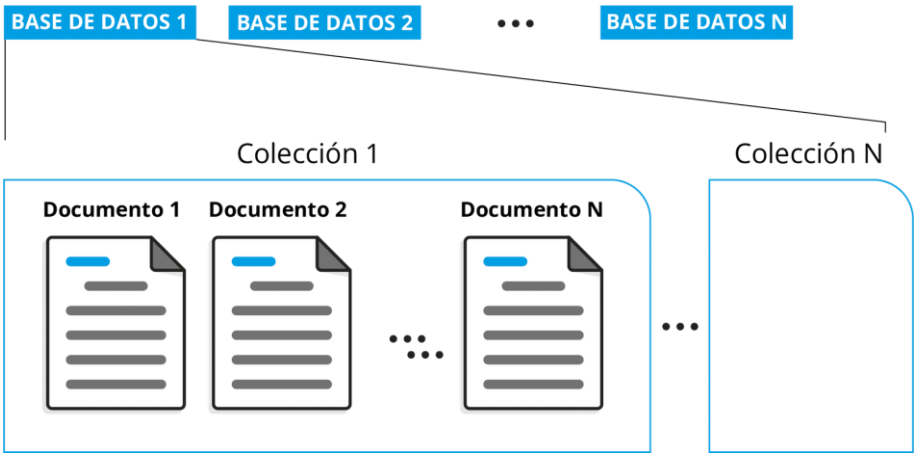
Ejemplos: Riak, Redis, Dynamo, Vodemort.



b) **Documentos**

Se representan en formato XML, JSON o BSON, son flexibles, no utilizan una estructura rígida y hay reducción de la complejidad en las consultas para datos asociados.

Ejemplos: MongoBD, CouchDB



c) **Columnas**

Basado en un modelo de almacenamiento «tabular». Por cada entrada, hay una columna, por lo tanto, los datos de cada entrada están dispuestos uno debajo del otro; permiten grandes volúmenes de datos, poseen gran escalabilidad y disposición de la información.

Ejemplos: Cassandra, Hypertable, Hbase, SimpleDB

Número	1.	2.	3.
Apellido	Skywalker	Kenobi	Organa
Nombre	Luke	Obi-wan	Leia
Clave	3FN-Z768	7TR-K345	8NN-R266

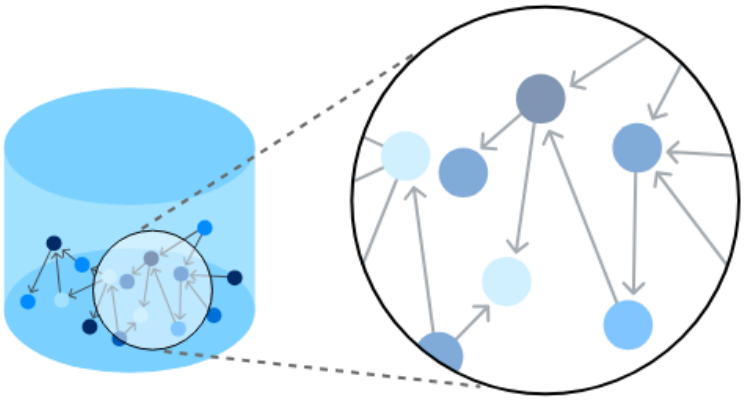
d) **Grafos**

Los datos se modelan como un conjunto de relaciones entre elementos, tienen alto rendimiento en consultas de relaciones de proximidad entre datos, y no para ejecutar consultas globales. Existe flexibilidad en la definición de atributos y longitud de

registros.

Ejemplos: Neo4j, Infinite Graph

Figura 4: Grafos de Neoj4j



Fuente: Neo4j. (s.f.).

Ventajas:

1. **Aplicaciones de big data:** grandes volúmenes son manejados fácilmente por las bases de datos NoSQL.
2. **Administración de la base de datos:** Requieren menos administración práctica, cuenta con capacidades de distribución de datos y reparación automática, modelos de datos simplificados y menos requisitos de ajuste y administración.
3. **Versatilidad:** Las posibilidades de crecimiento en el volumen de datos o la posibilidad de incluir cambios sobre la forma en la que ingresan los datos sin necesidad de alterar la estructura, permite adaptarse de forma rápida a un entorno de alto dinamismo.
4. **Crecimiento horizontal:** Son altamente escalables, si se requiere instalar mayor cantidad de nodos para ampliar la capacidad, se puede hacer sin problemas. Esto no interrumpe la usabilidad o consultas dentro de la base de datos.

5. **Economía:** No se necesitan servidores con gran cantidad de recursos para operar. La adaptabilidad y flexibilidad permiten empezar con bajos niveles de inversión en equipos e ir ampliando la capacidad a medida de las necesidades.

Desventajas

1. **Atomicidad:** Algunas de estas bases de datos no incorporan la atomicidad de información. Esto puede derivar en que la información no sea consistente entre nodos.

2. **Software poco documentado:** Puede adolecer de que algunas operaciones sean limitadas por la falta de información sobre las herramientas y características.

3. **Baja estandarización:** No se tiene un criterio plenamente definido entre los motores que se utilizan en este tipo de base de datos. El lenguaje tiende a variar según el tipo de base de datos que se vaya a utilizar.

4. **Herramientas GUI:** la mayoría de las bases de datos NoSQL no contienen una interfaz gráfica. Requiere conocimiento especial para poder ejecutar algunas de ellas.

A continuación, señalaremos las diferencias entre estos dos tipos de bases:

Base de Datos Estructuradas o Relacionales	Base de Datos No Estructuradas o No Relacionales
Utilizan el lenguaje de comunicación estándar SQL	Utilizan APIs comunicación diferentes, muchas de ellas reconocen el formato JSON

Esquema rígido definido a priori	Esquema flexible, se puede ir definiendo según se incorporan nuevos datos
Garantiza las propiedades ACID (Atomicity, Consistency, Isolation and Durability)	No garantiza las propiedades ACID (Atomicity, Consistency, Isolation and Durability)
Tienen un modelo de datos único	Suelen escalar bien horizontalmente y tener varios modelos de datos
Tienen filas orientadas	Favorecen la escalabilidad, principalmente la horizontal
	Suelen ser distribuidas y de código abierto
	Normalmente, no soportan operaciones JOIN

Revisemos otros cuadros comparativos que nos ayudarán a establecer otras diferencias.

Cuadro Comparativo 1

TIPOS DE BASES DE DATOS

SQL	NOSQL
Relacional	No Relacional
Basada en Tablas	Basada en Documentos
Mantiene estructura definida	Su estructura puede ser dinámica
Tienen redundancia controlada	Permite la redundancia de datos
Para aplicaciones que requieren datos consistentes (sin errores)	Para aplicaciones que requieren grandes cantidades de datos a consultar
Escalables verticalmente: aumentar las prestaciones del servidor	Escalable horizontalmente: agregar más servidores

Cuadro Comparativo 2

La diferencia fundamental entre ambos tipos de base de datos radica en que las base de datos NoSQL no utilizan modelo relacional

01	Debido al tiempo en el mercado, tienen más soporte	Optimizado para consultas con grandes cantidades de datos	01
02	Formada por tablas que contienen campos estructurados	Almacenamiento no estructurado, lo que permite alta escalabilidad	02
03	Necesitan más recursos, entre más compleja. mayor procesamiento	Abierta, por tanto flexible, no necesita tantos recursos para su ejecución	03

Las Bases de datos NoSQL no son sustitutas de las base de datos SQL: son una alternativa que ofrece otras posibilidades como el procesamiento de grandes volúmenes de datos. Cuando los datos deben ser consistentes sin dar posibilidad al error, se debe usar bases de datos SQL.

Elementos de las Bases de Datos no Estructuradas documentales y características de MongoDB

Elementos de las bases de datos no estructuradas documentales

Iniciemos revisando el papel de las bases de datos en las aplicaciones software. Se pueden diferenciar dos usos muy comunes:

1. Permiten integrar múltiples aplicaciones dando soporte al almacenamiento centralizado de la información, la complejidad de la integración es una desventaja, por ejemplo, cuando se requiere hacer algún cambio sobre los datos almacenados.

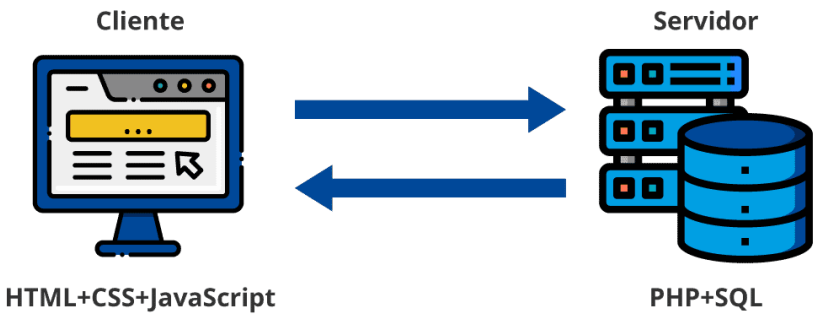
2. Se usa mediante una única aplicación, esto hace que la mantención y la realización de cambios sea más fácil.

Veamos el caso de la arquitectura basada en servicios web

Es común que se utilice en estos tiempos una arquitectura basada en servicios, que se basa en servicios web. En dicho caso, la comunicación se realiza bajo HTTP, lo que permite un

mecanismo de comunicación ampliamente utilizado.
Si se utiliza SQL como medio de interacción, los datos deben ser estructurados como relaciones. Por otro lado, como son servicios web, se pueden usar otras estructuras con registros anidados o listas representados como documentos XML o JSON.
En este contexto, se puede elegir la base de datos que se desee, ya que existe un **desacoplamiento entre el almacenamiento y los servicios**.

Figura 5: Relación Cliente - Servidor



Limitaciones:

a) **Integración:** La posibilidad de que muchas aplicaciones compartan los mismos datos y compartir la base de datos con algunas políticas de control de concurrencia.

b) **Modelo estándar:** Es un modelo que ha sido exitoso en la historia debido a que todas las bases relacionales trabajan con los mismos conceptos y el mismo lenguaje, SQL

c) **Impedancia:** Diferencia entre el modelo relacional y las estructuras de memoria. las bases de datos relacionales usan el concepto de fila y columna, no pudiendo utilizar otras estructuras como listas o registros anidados. Por esto el trabajo con las estructuras de la memoria son más rápidas que el trabajo de las bases de datos relacionales.

d) **Integración en un clúster:** Un clúster es un conjunto de máquinas que permiten implementar soluciones para escalar una aplicación. Las bases de datos relacionales no están diseñadas para ejecutarse sobre un clúster. Solo algunas trabajan con discos compartidos.

e) **Costo:** Las bases de datos relacionales están pensadas para ejecutarse en un único servidor, por lo que si se quieren ejecutar en clúster requieren de varias instancias de base de datos, lo que aumenta el costo económico.

Debido a las limitaciones expuestas, algunas empresas como Google y Amazon que desarrollaron sistemas basados en clúster como big tables, de Google, y Dynamo, de Amazon, surgen las bases de datos NoSQL.

Ventajas:

- Productividad del desarrollo de aplicaciones; Se simplifica la codificación, la depuración y la evolución de las aplicaciones.
- Datos a gran escala: El uso de grandes cantidades de datos procesados rápidamente. La mayoría de las bases de datos NoSQL se ejecutan en clúster.

Características

A modo de reforzamiento, repasemos las principales características de las bases de datos relacionales:

- No usan SQL como lenguaje de consultas (CQL en Cassandra)
- Proyectos de código abierto
- Necesidad de ejecutarse en clúster. (Hay excepciones)
- No tiene un esquema fijo (libertad y flexibilidad)

Modelos sobre agregados y modelos de distribución

Los siguientes son modelos de base de datos NoSQL orientados sobre agregados:

- Las tres primeras son orientadas a agregados, donde se utilizan listas y otras estructuras de registros que están anidados.
- Los agregados van a representar a un conjunto de datos relacionados que son tratados como una unidad.

Esta estructura, al ser una unidad indivisible, facilita la implementación de clústeres, ya que permite la replicación y distribución eficiente de los datos.

Los siguientes son modelos de distribución de datos de las bases de datos NoSQL:

Existen dos modelos de distribución, los cuales pueden ser complementarios: **Replicación:** Técnica que implica copiar los mismos datos en múltiples servidores. **Sharding:** Técnica que distribuye diferentes datos a múltiples servidores, de manera que cada servidor actúe como una única fuente para un subconjunto de datos.

La consistencia de datos en las bases de datos NOSQL

Cuando dos usuarios tratan de actualizar el mismo dato se produce un conflicto de escritura. Cuando las escrituras llegan al servidor, las serializa y entonces decide aplicar una y, a continuación, la otra. De esta forma, uno de los usuarios sufre una pérdida de actualización, por lo cual se aplica una serie de bloqueos para determinar el orden de ejecución.

a) **Consistencia de lectura**: Una inconsistencia de lectura se produce cuando un usuario realiza una lectura en mitad de la escritura de otro sobre los mismos datos que se están leyendo.

b) **Consistencia de replicación:** Las bases de datos NoSQL tienen consistencia de replicación, lo cual consiste en asegurar que los mismos datos tengan el mismo valor cuando son leídos desde diferentes réplicas. Por el tipo de BD, se establece que el tiempo de respuesta es de milisegundos para las inconsistencias.

c) **Consistencia de sesión**: Las ventanas de inconsistencia, que ocurren cuando los datos no se actualizan de forma instantánea en todos los nodos de un clúster, pueden generar confusiones para el usuario. **Un ejemplo claro es cuando se publica un contenido en una web y al actualizar la página, el cambio no aparece porque la actualización se ha dirigido a un nodo que aún no la ha recibido. Para evitar estas situaciones, se requiere una consistencia del tipo "leer lo que tú escribes", la cual garantiza que una vez realizada una actualización, el usuario la verá en las siguientes consultas dentro de la misma sesión.**

MongoDB

MongoDB es una base de datos NoSQL **orientada a documentos.** Fue creada por la compañía 10gen, en el año 2007. Se caracteriza porque almacena datos en documentos de tipo JSON con un esquema dinámico denominado BSON.

JSON	BSON
Es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las	BSON (Binary JSON) es un formato de datos que, si bien no es directamente legible como texto plano, a diferencia

máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript.	de JSON, ofrece a las tecnologías como bases de datos un acceso más rápido y directo a la información.

Documentos

Los documentos son la unidad básica de organización de información en MongoDB. Se trata de un conjunto ordenado de claves que tiene asociados determinados valores. Su estructura es clave-valor como, por ejemplo: **{"Nombre":"Juan", "Pais":"Chile"}**

Las claves son cadenas, con algunas excepciones:

- No puede tener el valor nulo <<\0>>
- Tampoco puede tener el punto << . >> y << $ >>
- MongoDB es sensitivo a mayúsculas/minúsculas y a los tipos de datos; {"Edad":3}, {"Edad":"3"}, {"edad":3}, {"edad":"3"} son distintos.
- Los documentos no pueden tener claves duplicadas, por ejemplo {"edad":20}, {"edad":30}
- Los pares clave-valor están ordenados en los documentos: {"x":3, "y":5} es distinto que {"y":5, "x":3}
- Los valores de los documentos pueden ser de diferentes tipos.

Tipos de datos	Característica	Ejemplo
Nulo	Representa el valor nulo o un campo que no existe	{“x”: null}
Booleanos	Puede tomar los valores true o false	{“x”: true}
Números	Pueden ser números reales o números enteros	{“x”:3.14} {“x”:3}
Cadenas	Cualquier cadena de caracteres	{“x”: ”Hola Mundo”}
Fecha	Almacena fecha en milisegundos, pero no la zona horaria	{“x”: new Date()}
Expresiones regulares	Se pueden usar para realizar consultas	

Arrays	Se representa como un conjunto o lista de valores	{“x”: [“a”, ”b”, ”c”]}
Documentos embebidos	Los documentos pueden contener documentos embebidos como valores de un documento padre	{“x”: {“y”:45}}
Identificadores de objetos	Es un identificador de 12 bytes para un documento	{“x”: ObjectId()}
Datos Binarios	Es una cadena de bytes arbitraria que no puede ser manipulada directamente desde la Shell y que sirve para representar caracteres no UT8	
Código JavaScript	Los documentos y las consultas pueden	{“x”: function(){….}}

	contener código JavaScript	

Algunas consideraciones a tomar en cuenta:

a) **Fechas:** Para crear un objeto de tipo fecha se usa el comando new Date() y se llama la fecha sin el new; también se puede usar ISODate.

b) **Arrays:** Pueden contener diferentes tipos de datos y se puede navegar dentro del array.

a. {“Cosas”: [“edad”:45]}

c) **Documentos embebidos:** Se usan para anidar la información y así su estructura sea natural y así poder navegar, crear índices o realizar actualizaciones.

{
"nombre": "Juan”,
"dirección": {
"calle":"Mayor 3",
"ciudad": "Madrid",
"País": "España”
}}

d) **Identificador de objetos:** Cada documento tiene una clave _id:

- Puede ser de cualquier tipo, por defecto ObjectId

- En una colección cada documento tiene un id único, por lo que cada documento puede ser identificado de manera única.

- El ObjectId está compuesto de 12bytes, lo que permite representar una cadena de 24 dígitos hexadecimales.

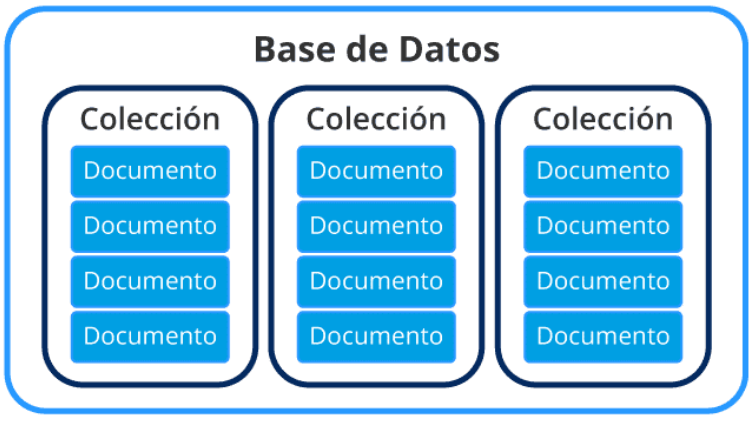
Figura 5: ObjectId

Colecciones

Es un grupo de documentos y desempeña un papel análogo a las tablas de las bases de datos relacionales. Tiene esquemas dinámicos, puede haber documentos con diferentes estructuras.

{“edad”:34},{“x”:”casa”}

Figura 6: Colecciones en Base de datos



Fuente: Elaboración propia

¿Por qué es necesario tener más de una colección?

- Para que las búsquedas sean más rápidas.
- Para que las estructuras de las colecciones en cuanto a los documentos sean similares.

Para poder indexar los documentos de forma óptima.

Al ser una estructura dinámica, no es obligatorio tener más de una colección en la base de datos.

Una colección se identifica por su nombre. Este aspecto presenta ciertas restricciones:

- La cadena vacía no es válida como nombre.
- Los nombres de las colecciones no pueden tener el carácter nulo \0
- No se puede crear una colección que comience con system, ya que es una palabra reservada interna de la base de datos.
- Las colecciones no pueden llevar el carácter $

Bases de datos

Las colecciones se agrupan en bases de datos, de manera que una instancia de MongoDB puede gestionar varias bases de datos cada una agrupando cero o más colecciones. Cada base de datos se almacena en archivos separados y con sus propios permisos. Los datos de una aplicación deben estar en una base de datos.

Estas son algunas características de la base de datos:

a) **Precisiones acerca de los nombres:**

La cadena vacía no es válida como nombre de la base de datos.

El nombre no puede tener ninguno de los siguientes caracteres: \, /, . , ” , * , < , >, : , ? , $, espacio o \0

Los nombres de las bases de datos son sensitivos a mayúsculas/minúsculas.

Los nombres están limitados a 64bit

b) **Nombres reservados:** Hay nombres reservados que no pueden ser usados como nombres de base de datos:

admin: es el nombre de la base de datos root en términos de autenticación}

local: base de datos que no será replicada

configura: se usa para base de datos de tipo fragmento.

En síntesis, en este recurso de la asignatura aprendiste sobre las características y diferencias que existen entre los dos tipos de bases de datos (NSQL y NoSQL). Ahora sabes que las bases de datos estructuradas o relacionales se utilizan como almacenamiento para sistemas transaccionales que necesitan una fuerte confiabilidad en sus procesos.

Por otro lado, las bases de datos no estructuradas o no relacionales se usan en el almacenamiento de datos de sistemas informáticos que necesitan procesamiento de grandes volúmenes de información de forma rápida.

También aprendiste que MongoDB es una base de datos de tipo documental sustentada, estructuralmente, en la creación colecciones y documentos, y que posee diferentes formas de representación de la información y tiene operadores (comandos) para todas sus operaciones como: crear, insertar, eliminar, actualizar y consultar datos.

Bibliografia

- **Iglesias, A., Cuadra, D., & Castro, E.** (2014). *Desarrollo de bases de datos*. RA-MA Editorial. Inacap - Desarrollo de bases de datos: casos prácticos desde el análisis a la implementación. https://elibro.net/es/lc/inacap/titulos/58524
- **Sarasa, A. (2026).** *Introducción a las bases de datos NoSQL usando MongoDB*. Editorial UOC. https://elibro.net/es/lc/inacap/titulos/58524
- **Silberschatz, A., Korth, H., & Sudarshan, S**. (2014). *Fundamentos de base de datos* (6ta ed.). McGraw-Hill.