

# Tipos de bases de datos y cómo trabajar con ellas

Backend



# Tipos de bases de datos y cómo trabajar con ellas

Datos, datos, datos... Nos movemos en un mundo en el que los datos aparecen por todos los lados, pero esos datos habrá que guardarlos en algún sitio, claro está. Pues bien, de esos sitios vamos a hablar en este tema, de las bases de datos.

Nuestro objetivo de aquí en páginas sucesivas es que entiendas **qué son las bases de datos y para qué sirven**, que sepas también distinguir los diferentes **tipos** de bases de datos con los que te puedes encontrar y, ¡cómo no!, también queremos que aprendas unas nociones, al menos, básicas de su manejo.

Eso sí, no esperes de este fastbook una guía teórica, para eso ya está la documentación oficial de los distintos motores de bases de datos; lo que buscamos con este fastbook es que aprendas a manejarte con las BD desde un punto de vista práctico, para que tengas una visión más global.

¡Vamos allá!

Autor: Antonio Blanco Oliva

Introducción a las bases de datos

Las bases de datos relacionales

MySQL & PostgreSQL

Las bases de datos NoSQL

MongoDB

Otras bases de datos: las vectoriales

# Introducción a las bases de datos



Una base de datos se define como un conjunto de datos estructurados que pertenecen al mismo contexto.

Gestionar esta información y estructurarla adecuadamente es una labor que debe realizar un software, al que llamaremos *sistema gestor de base de datos*, cuyas siglas en inglés son DBMS.

La propia definición deja abierta la forma en la que los datos serán estructurados, la cual ha evolucionado a lo largo del tiempo dando lugar a distintas maneras de estructuración o de familias de sistemas gestores de bases de datos: **SQL, NOSQL y vectoriales**, entre otros. Pero todos cumplen una serie de características por las que fueron diseñados:

**EL ESCALADO**

**LA INTEGRIDAD**

**LA SEGURIDAD**

La cantidad de datos cada vez es mayor y, por tanto, resulta imposible gestionarla de forma manual; de ahí que haya tomado sentido un gestor de base de datos.

**EL ESCALADO****LA INTEGRIDAD****LA SEGURIDAD**

La coherencia de los datos debe mantenerse a lo largo del tiempo, y se debe velar por ella.

**EL ESCALADO****LA INTEGRIDAD****LA SEGURIDAD**

No todo el mundo debe poder acceder a todos los datos. Un sistema de control por capacidades o roles permitirá llevar un control sobre los accesos.

Las bases de datos, y más concretamente sus motores de gestión, han ido evolucionando con el tiempo, según los requerimientos del mercado, aunque podemos organizarlos en 3 tipos fundamentales:

1

Las bases de datos relacionales: SQL

2

Las bases de datos no relacionales: NoSQL

3

Y otras, entre las que me gustaría destacar las vectoriales.

# Las bases de datos relacionales



Las bases de datos relacionales son aquellas que usan como estructura para almacenar los datos, las tablas, las columnas y las filas, así como las relaciones entre ellas.

La estructura de datos se refiere a cómo se organizan y almacenan los datos. Los tipos de datos son las categorías que especifican qué tipo de información se puede almacenar en cada columna de una tabla.

A continuación, vamos a describir los conceptos clave vinculados con la estructura de datos y los tipos de datos en las bases de datos relacionales.

## Estructura de datos en bases de datos relacionales

- Tablas (o relaciones).** Las tablas son la estructura principal en una base de datos relacional. Cada tabla almacena datos relacionados y se organiza en filas y columnas.

- Columnas (o atributos).** Cada tabla consta de columnas que representan atributos específicos de los datos. Cada columna tiene un nombre y un tipo de datos que define la clase de información que puede almacenar.
- Filas (o registros).** Cada fila en una tabla representa un conjunto de datos relacionados. Cada fila contiene valores para cada una de las columnas de la tabla.
- Clave primaria.** La clave primaria es un atributo o conjunto de atributos que identifican de manera única cada fila en una tabla. Asegura la integridad y la unicidad de los datos en la tabla.
- Clave externa (o foránea).** Una clave externa es un atributo en una tabla que establece una relación con la clave primaria de otra tabla. Se utiliza para definir relaciones entre tablas.

Trabajemos estos conceptos sobre un ejemplo real.

Imaginemos que tenemos nuestro currículum con cierta información personal, nuestras *skills* y la formación académica.

Dicha información se podría dividir en 3 tipos de entidades:

- **Candidato:** contendrá la información personal del candidato.
- **Formación:** será la formación que tienen dicho candidato.
- **Skill:** representa la habilidad o conocimiento que puede tener un candidato.

Aquí podemos ver cómo sería la estructura de la tabla, las columnas y las filas:

Candidato

ID	Nombre	Edad	...
1	Antonio	43	...
2	Maria	38	...
...	...	...	...

Skill

ID	Título	Nivel
1	Bases de datos	80
2	Docker	90
...	...	...

Formación

ID	Título	Fecha Inicio	Fecha Fin
1	Back end Developer	01/09/2023	28/02/2024
2	Front end Developer	01/09/2023	28/02/2024
...	...	...	

## Tipos de datos en bases de datos relacionales

Los tipos de datos en una base de datos relacional definen el formato de los valores que se pueden almacenar en una columna. Veamos cuáles son esos tipos y qué incluyen.

1

**Enteros.** Estos tipos de datos almacenan números enteros, por ejemplo:

INT, BIGINT, SMALLINT...

2

**Decimales o reales.** Se utilizan para almacenar números con decimales e incluyen:

DECIMAL, NUMERIC, FLOAT, REAL, DOUBLE PRECISION...

3

**Texto y cadenas de caracteres.** Almacenan texto o caracteres, por ejemplo:

CHAR, VARCHAR, TEXT...

4

**Fechas y tiempos.** Estos tipos almacenan fechas y horas, por ejemplo:

DATE, TIME, TIMESTAMP...

5

**Booleanos.** Almacenan valores verdaderos o falsos. En algunos sistemas de gestión de bases de datos se utilizan tipos como:

BOOLEAN o BIT.

6

**Binarios.** Almacenan datos binarios, como imágenes o archivos, por ejemplo:

BLOB (Binary Large Object) o VARBINARY.

7

**Enumeraciones y tipos definidos por el usuario:** algunas bases de datos permiten definir tipos de datos personalizados, como las enumeraciones.

8

**Otros tipos específicos del sistema:** hay bases de datos que pueden tener tipos de datos específicos del sistema, como arrays o tipos de datos geoespaciales.

---

**Los tipos de datos utilizados en una base de datos relacional dependen del sistema de gestión de bases de datos (DBMS) específico que estés utilizando, pero, en general, los mencionados anteriormente son comunes en la mayoría de los DBMS. La elección del tipo de datos adecuado para una columna es importante para garantizar la integridad de los datos y la eficiencia de las consultas.**

## Relaciones

En las bases de datos relacionales, las relaciones entre tablas son fundamentales para organizar y relacionar los datos. Aquí tienes una descripción de los **tipos de relaciones más comunes** entre tablas en una base de datos relacional:

1

### Relación uno a uno (1:1).

- En esta relación, un registro en una tabla está relacionado con un solo registro en otra tabla y viceversa.
- Es útil cuando tienes información que se divide en dos tablas por razones de organización, pero cada fila en una tabla tiene una correspondencia única con una fila en la otra.

2

**Relación uno a muchos (1:N).**

En esta relación, un registro en una tabla se relaciona con varios registros en otra tabla, pero un registro en la segunda tabla se relaciona con un solo registro en la primera tabla.



Es la relación más común en las bases de datos relacionales y se utiliza para representar relaciones de uno a muchos, como un cliente que tiene múltiples pedidos.

3

**Relación muchos a uno (N:1).**

Es básicamente lo contrario de la relación uno a muchos. Aquí, varios registros en una tabla se relacionan con un solo registro en otra tabla.

4

**Relación muchos a muchos (N:N).**

En esta relación, varios registros en una tabla pueden relacionarse con varios registros en otra tabla y viceversa.



Se utiliza cuando necesitas representar una relación compleja entre entidades, como estudiantes inscritos en múltiples cursos y cursos que tienen múltiples estudiantes.

Para implementar estas relaciones, se utilizan **claves primarias** y **claves externas** (o foráneas) en las tablas.

---

La clave primaria es un campo único en una tabla que se utiliza para identificar de manera única cada entrada, mientras que la clave externa es un campo en una tabla que se relaciona con la clave primaria de otra tabla. Esto establece la conexión entre las tablas.

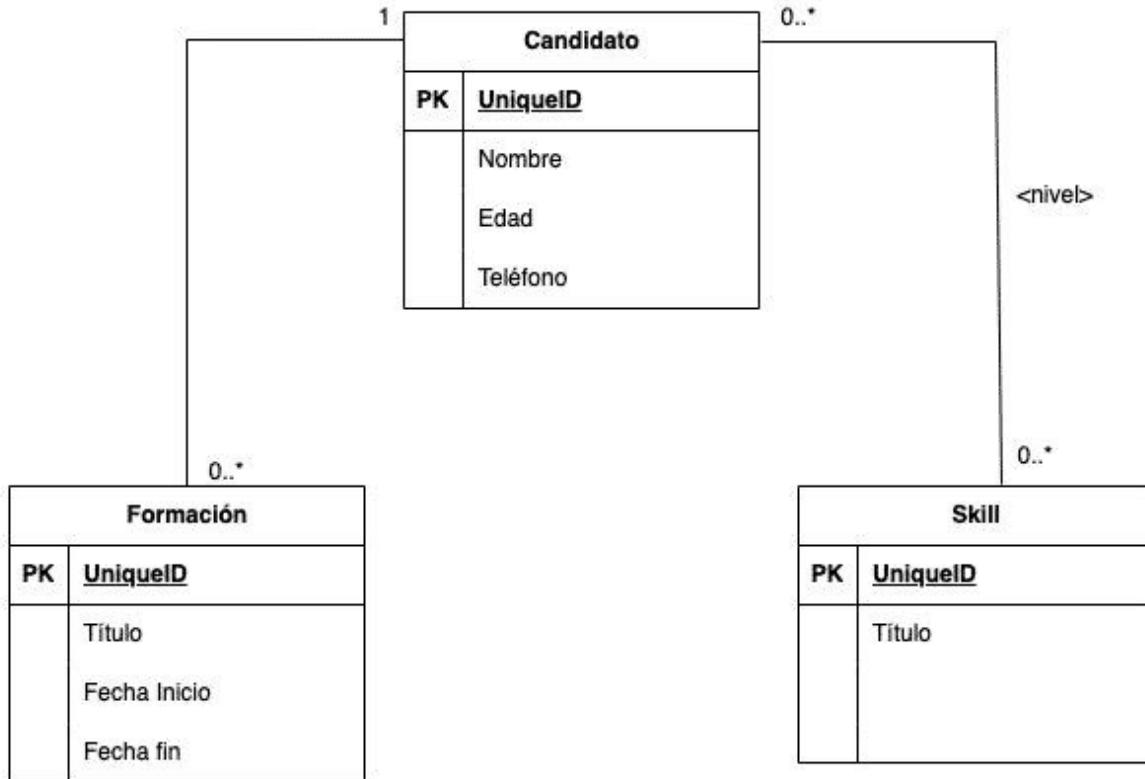
Encontrar el tipo de relación entre las tablas suele costar más inicialmente, pero la práctica te dará soltura para saber de qué tipo es cada relación.

Una forma práctica que suele ayudar a localizarla es usar el verbo '**tener**' o '**pertenecer**'.

Por ejemplo:

- "Un candidato tiene 0 o más recomendaciones"
  
- "Una recomendación pertenece a 1 candidato"

Lo que nos daría una relación de 1 a muchos.



Estas relaciones entre tablas van a permitir al motor gestor **añadir consistencia a los datos**, por ejemplo, controlando que una clave externa no pueda apuntar a un elemento que no existe o que las claves primarias sean únicas en las tablas.

## Vistas

---

Las vistas en una base de datos relacional son objetos virtuales que representan una consulta almacenada como una tabla lógica.

Las vistas permiten a los usuarios y aplicaciones acceder y manipular los datos de una base de datos de una manera más conveniente y segura. Aquí compartimos algunos aspectos clave sobre las vistas:

1

### **Definición de vistas.**

- Una vista es una consulta SQL almacenada en la base de datos como un objeto lógico.
- La consulta SQL utilizada para crear la vista puede contener selecciones, proyecciones, uniones, filtros y otros operadores SQL para recuperar los datos de una o varias tablas.

2

### **Características de las vistas.**

- Las vistas no almacenan datos físicamente, sino que muestran los datos de una o varias tablas subyacentes en tiempo real.
- Las vistas son útiles para ocultar la complejidad de las consultas a los usuarios y proporcionar una representación simplificada de los datos.
- Las vistas pueden restringir el acceso a datos sensibles al permitir a los usuarios acceder solo a las columnas o filas específicas que se definen en la vista.

3

### **Ventajas de las vistas.**

- Simplifican el acceso a los datos al proporcionar una capa de abstracción sobre las tablas subyacentes.

- Ayudan a garantizar la seguridad y la integridad de los datos al limitar lo que los usuarios pueden ver y manipular.
  
- Facilitan la gestión de cambios en la estructura de la base de datos, ya que los cambios en las vistas no afectan a las aplicaciones que las utilizan.

## Procedimientos o funciones

Cabe mencionar que los procedimientos o funciones son un tipo de objeto en bases de datos relacionales que contienen una serie de instrucciones SQL. Estos procedimientos se almacenan en la base de datos y se pueden invocar y ejecutar mediante una llamada desde una aplicación o directamente desde el sistema de gestión de bases de datos (DBMS). Los procedimientos almacenados son especialmente útiles para **ejecutar tareas específicas o lógica de negocios** en la base de datos de una manera controlada y reutilizable.

# MySQL & PostgreSQL



En este apartado, vamos a ver dos de los **principales motores de bases de datos relacionales** que se usan en el mercado, aunque hay bastantes más. Además, es mejor tener afianzados los conocimientos base que cómo lo implementa cada uno.

## MySQL

MySQL es un sistema de gestión de bases de datos relativos (RDBMS) de código abierto ampliamente utilizado. Fue desarrollado originalmente por la empresa sueca **MySQL AB** y posteriormente adquirido por **Sun Microsystems**, que a su vez fue adquirida por Oracle. MySQL es uno de los sistemas de gestión de bases de datos más populares del mundo y es ampliamente utilizado en aplicaciones web y empresariales.

Estos son los **aspectos clave** de MySQL que debes conocer:



### Código abierto

MySQL es de código abierto, lo que significa que **su código fuente es accesible y modificable** por la comunidad de desarrolladores. Esto ha llevado a la creación de muchas variantes y proyectos relacionados, como **MariaDB**.



### **Soporte para múltiples plataformas**

Está disponible en una variedad de plataformas, incluyendo Linux, Windows y macOS, lo que lo hace **versátil y ampliamente compatible**, y cuenta con conectores para multitud de lenguajes y frameworks.



### **Escalabilidad**

También es conocido por su capacidad de **escalabilidad**, es decir, puede manejar bases de datos pequeñas y simples, así como bases de datos grandes y complejas con millones de registros.



### **Rendimiento**

Goza además de **buen rendimiento** en aplicaciones web y sistemas de bases de datos, lo que lo hace adecuado para una amplia gama de aplicaciones.



### **Características avanzadas**

Destaca por funcionalidades como la **replicación** (para redundancia y alta disponibilidad), la **partición** (para dividir grandes tablas en partes más manejables) y la **gestión de transacciones** (para garantizar la integridad de los datos en operaciones críticas).



## Amplia comunidad y ecosistema

MySQL cuenta con una **gran comunidad** de usuarios y desarrolladores, lo que supone una abundancia de recursos, documentación y soporte disponible en línea. Esto nos ayudará a solventar futuros problemas o atascos que tengamos.

Veamos a continuación algunos de los **comandos** más comunes en este caso para MySQL, pero prácticamente extensibles como SQL general.

- **Conexión**

```mysql`: este comando se utiliza para conectarse a una base de datos MySQL desde la línea de comandos. Puedes proporcionar detalles de conexión como el nombre del servidor, el nombre de usuario y la base de datos a la que deseas acceder. Por ejemplo:

```
mysql -u usuario -p -h host -D nombre_base_de_datos
```

- **Comandos SQL**

**SELECT**: se utiliza para recuperar datos de una o varias tablas en la base de datos. Es uno de los comandos más comunes y se utiliza para realizar consultas. Por ejemplo:

```
SELECT columnal, columna2 FROM tabla WHERE condición;
```

*INSERT INTO:* se utiliza para agregar nuevos registros a una tabla. Por ejemplo:

```
INSERT INTO tabla (columnal, columna2) VALUES (valor1, valor2);
```

*UPDATE:* se utiliza para modificar registros existentes en una tabla. Por ejemplo:

```
UPDATE tabla SET columnal = nuevo_valor WHERE condición;
```

*DELETE:* se utiliza para eliminar registros de una tabla que cumplan con una condición especificada. Por ejemplo:

```
DELETE FROM tabla WHERE condición;
```

*CREATE TABLE*: se utiliza para crear una nueva tabla en la base de datos. Por ejemplo:

```
CREATE TABLE nueva_tabla (
    columna1 tipo_de_dato,
    columna2 tipo_de_dato,
    ...
);
```

*ALTER TABLE*: se utiliza para modificar la estructura de una tabla existente, como agregar, modificar o eliminar columnas. Por ejemplo:

```
ALTER TABLE tabla
ADD COLUMN nueva_columna tipo_de_dato;
```

*DROP TABLE*: se utiliza para eliminar una tabla y todos sus datos de la base de datos. Por ejemplo:

```
DROP TABLE tabla;
```

*CREATE DATABASE*: se utiliza para crear una nueva base de datos en el servidor de MySQL. Por ejemplo:

```
CREATE DATABASE nueva_base_de_datos;
```

**SHOW DATABASES:** se utiliza para mostrar la lista de bases de datos disponibles en el servidor MySQL. Por ejemplo:

```
SHOW DATABASES;
```

Una vez que conocemos los comandos entremos a descubrir los **procedimientos**.

---

**En MySQL, los procedimientos almacenados se pueden crear utilizando el lenguaje de programación SQL/PSM (Persistent Stored Modules). Esto permite definir procedimientos almacenados que incluyen declaraciones condicionales, bucles y otras estructuras de control de flujo.**

---

Puedes utilizar la sentencia CREATE PROCEDURE para crear un procedimiento almacenado y luego ejecutarlo con CALL.

Un ejemplo de creación de un procedimiento almacenado en MySQL sería el siguiente:

```
DELIMITER //
CREATE PROCEDURE sp_example(IN input_param INT)
BEGIN
    -- Lógica del procedimiento aquí
    SELECT * FROM tabla_ejemplo WHERE columna = input_param;
END //
DELIMITER ;
```

En cuanto a **las vistas**, volvamos a nuestro ejemplo del CV de los candidatos. Imaginemos que queremos que ciertos usuarios no puedan ver más allá del nombre y apellidos de nuestros candidatos, por tema de protección de datos u otro motivo. ¿Cómo lo solucionamos?

Pues podemos crear una vista que solo muestre el nombre y apellidos. Así si a un usuario o aplicación no le damos acceso a la tabla original, pero sí a la vista, solo tendrá disponible esos datos básicos.

```
CREATE VIEW vista_candidato AS
SELECT nombre, apellidos
FROM cv_candidato;
```

## PostgreSQL

PostgreSQL, conocido como *Postgres*, es un sistema de gestión de bases de datos relacional de código abierto altamente extensible y compatible con estándares. Ofrece un amplio soporte de SQL, garantiza la integridad de los datos a través de propiedades ACID y permite la creación de funciones y extensiones personalizadas. PostgreSQL es escalable, proporciona soluciones de replicación y alta disponibilidad, y es ampliamente utilizado en aplicaciones críticas y científicas debido a su enfoque en la precisión y la flexibilidad.

1

### Características destacables

- Extensibilidad y personalización.
- Cumplimiento de estándares SQL.
- Propiedades ACID para la integridad de los datos.
- Escalabilidad y capacidad para manejar grandes volúmenes de datos.
- Replicación y alta disponibilidad.
- Mantenimiento avanzado de transacciones.
- Licencia de código abierto.
- Ampliamente utilizado en aplicaciones críticas y científicas.

2

## URLs y herramientas que necesitas conocer

- [PostgreSQL](#)
- [pgAdmin](#)

3

## Comandos

En cuanto a los **comandos** en PostgreSQL, estos son muy similares a los vistos en MySQL, ya que ambos usan el lenguaje SQL con pequeñas diferencias. Vamos a estudiarlos.



### Conexión

**psql:** este comando se utiliza para conectarse a una base de datos PostgreSQL desde la línea de comandos. Puedes proporcionar detalles de conexión como el nombre del servidor, el nombre de usuario y la base de datos a la que deseas acceder. Por ejemplo:

```
psql -h hostname -U username -d database
```

**Aunque MySQL y PostgreSQL son sistemas de gestión de bases de datos relacionales (RDBMS) que siguen el estándar SQL en gran medida, existen algunas diferencias en los comandos comunes entre ambas plataformas.**

A continuación, nos presentamos algunas de las diferencias.

- **Sintaxis de AUTO\_INCREMENT e IDENTITY**

MySQL utiliza la sintaxis *AUTO\_INCREMENT* para definir columnas que se autogeneran. Por ejemplo:

```
CREATE TABLE tabla (id INT AUTO_INCREMENT, nombre VARCHAR(255));
```

PostgreSQL utiliza *SERIAL* o *IDENTITY* para el mismo propósito. Por ejemplo:

```
CREATE TABLE tabla (id SERIAL, nombre VARCHAR(255));
```

- **Tipos de datos y columnas BOOLEAN**

En MySQL, el tipo de datos booleano se representa generalmente como *TINYINT* con valores 0 y 1 para falso y verdadero, respectivamente.

PostgreSQL tiene un tipo de datos específico llamado *BOOLEAN* que utiliza los valores *TRUE* y *FALSE*.

- **Comandos de renombrado de columnas**

En MySQL para renombrar una columna, se utiliza la sentencia *ALTER TABLE* con el comando *CHANGE COLUMN*.

```
ALTER TABLE tabla CHANGE COLUMN vieja_columna nueva_columna tipo_de_dato;
```

En PostgreSQL para renombrar una columna, se utiliza *ALTER TABLE* con el comando *RENAME COLUMN*.

```
ALTER TABLE tabla RENAME COLUMN vieja_columna TO nueva_columna;
```

- **Sintaxis de comentarios en columnas**

MySQL utiliza la sintaxis *COMMENT* para agregar comentarios a columnas de una tabla.

```
ALTER TABLE tabla MODIFY COLUMN columna tipo_de_dato COMMENT 'Este es un co
```

PostgreSQL utiliza `COMMENT ON COLUMN` para agregar comentarios a columnas.

```
COMMENT ON COLUMN tabla.columna IS 'Este es un comentario.';
```

Como hicimos con MySQL, también estudiaremos los **procedimientos** en PostgreSQL.

Los procedimientos almacenados se pueden crear utilizando el lenguaje de programación PL/pgSQL, que es una extensión de SQL con características adicionales para la programación de procedimientos almacenados.

Puedes utilizar la sentencia `CREATE FUNCTION` para definir funciones almacenadas, que son equivalentes a procedimientos almacenados en otros sistemas de gestión de bases de datos.

Ejemplo de creación de una función almacenada en PostgreSQL:

```
CREATE OR REPLACE FUNCTION fn_example(input_param INT)
RETURNS TABLE (column1 INT, column2 TEXT) AS $$  
BEGIN
    -- Lógica de la función aquí
    RETURN QUERY SELECT column1, column2 FROM tabla_ejemplo WHERE columna
END;
$$ LANGUAGE plpgsql;
```

---

**Estas son algunas de las diferencias notables en los comandos comunes entre MySQL y PostgreSQL. Que como puedes ver son modificaciones de sintaxis, pero a grandes rasgos son bastante similares.**

---

## SQLite

Aunque no se ha comentado en el apartado dedicado a las bases de datos relacionales para centrarnos en MySQL y PostgreSQL, es interesante conocer un sistema muy liviano que se utiliza en pequeñas aplicaciones normalmente, el llamado SQLite.

SQLite es especialmente adecuado para aplicaciones que necesitan una base de datos local simple y rápida, como **aplicaciones móviles** que almacenan datos en el dispositivo del usuario. También es comúnmente utilizado para **aplicaciones de prueba y desarrollo** debido a su facilidad de uso y portabilidad. Sin embargo, no es la elección adecuada para las aplicaciones que requieren escalabilidad y acceso concurrente masivo, ya que está diseñado para un acceso de una sola conexión a la vez.

# Las bases de datos NoSQL



NoSQL, que significa *Not Only SQL* o *No Solo SQL*, es un término genérico que se utiliza para describir sistemas de bases de datos que no se basan en el modelo relacional tradicional utilizado en bases de datos SQL (*Structured Query Language*). En lugar de utilizar tablas con filas y columnas, como lo hace SQL, las bases de datos NoSQL **emplean modelos de datos flexibles** que permiten la gestión y el almacenamiento de datos no estructurados o semiestructurados.

Las bases de datos NoSQL son adecuadas para una variedad de aplicaciones donde la escalabilidad, la flexibilidad y la velocidad son fundamentales. Estas son algunas de sus **características** clave:

- Modelos de datos flexibles.** Las bases de datos NoSQL pueden manejar datos en diferentes formatos, como documentos, gráficos, columnas o llaves-valor. Esto permite adaptar la estructura de los datos según las necesidades de la aplicación.
- Escalabilidad horizontal.** Muchas bases de datos NoSQL están diseñadas para escalar horizontalmente, lo que significa que pueden distribuir datos en múltiples servidores para manejar grandes volúmenes de información y cargas de trabajo intensivas.
- Velocidad y rendimiento.** Estas bases de datos a menudo están optimizadas para consultas rápidas y tiempos de respuesta bajos, lo que las hace adecuadas para las aplicaciones en tiempo real y de alto rendimiento.

- Consistencia eventual.** Algunas bases de datos NoSQL ofrecen un modelo de consistencia eventual en lugar del enfoque ACID (atomicidad, consistencia, aislamiento y durabilidad) de las bases de datos relacionales. Esto permite una mayor escalabilidad a costa de una posible consistencia diferida.
- Simplificación de la administración.** En comparación con las bases de datos SQL, las NoSQL a menudo simplifican la administración al eliminar la necesidad de definir esquemas rígidos y realizar consultas complejas.
- Amplio espectro de uso.** Las bases de datos NoSQL se utilizan en una variedad de aplicaciones, incluyendo aplicaciones web, redes sociales, internet de las cosas (IoT), análisis de big data y más.

Existen varios **tipos** de bases de datos NoSQL, aquí te los presentamos:

### **Bases de datos de documentos**

Almacenan datos en documentos, como JSON o XML  
Por ejemplo: MongoDB y CouchDB.

### **Bases de datos de gráficos**

Se utilizan para modelar y almacenar datos relacionales complejos, como redes sociales.  
Por ejemplo: Neo4j y Amazon Neptune.

## Bases de datos de columnas

Almacenan datos en columnas en lugar de filas, lo que las hace adecuadas para análisis.  
Por ejemplo: Apache Cassandra y HBase.

## Bases de datos de llaves-valor

Almacenan datos en pares de llave-valor simples y son adecuadas para almacenar y recuperar datos rápidamente.  
Por ejemplo: Redis y Amazon DynamoDB.

La elección entre una base de datos SQL y una NoSQL depende de las necesidades específicas de tu proyecto, como el tipo de datos que manejarás, el rendimiento requerido y la escalabilidad deseada.

Una regla sencilla que se suele usar es: si vas a tener muchas lecturas en función del número de escrituras, con cruces de datos entre tablas, **las No Relacionales suelen ser más rápidas.**

Por ejemplo, mostrar el perfil de un usuario en una red social suele llevar mucha información cruzada de distintas tablas, que en NoSQL podemos modelar en un documento.

# MongoDB



MongoDB es un sistema de gestión de bases de datos NoSQL que se destaca por su capacidad para almacenar datos en un formato flexible y semiestructurado.

A diferencia de las bases de datos relacionales tradicionales, MongoDB permite a los desarrolladores almacenar datos en documentos que pueden variar en estructura, lo que facilita la adaptación a necesidades cambiantes. Además, es altamente escalable y distribuido, y por lo tanto adecuado para aplicaciones web y móviles de alto rendimiento. Su modelo de datos y lenguaje de consulta nos permiten trabajar con datos de manera eficiente, y es ampliamente utilizado en una variedad de aplicaciones que requieren flexibilidad y escalabilidad en el manejo de datos.

Veamos a continuación algunos **comandos básicos y cómo trabajar** con este sistema.

1

## Iniciar el Servidor de MongoDB.

Una vez instalado, tenemos que iniciar el servidor de MongoDB. En sistemas Unix, puedes usar el siguiente comando:

```
mongod
```

En sistemas Windows, puedes iniciar el servidor de MongoDB ejecutando:

```
mongod.exe
```

Eso sí, asegúrate de que el servidor esté en ejecución antes de continuar.

2

## Conexión.

Abre una nueva ventana de terminal o consola y conecta a MongoDB utilizando el cliente de línea de comandos *mongo*. Por defecto, se conectará al servidor en el puerto 27017.

```
mongo
```

3

**Para crear una base de datos.**

Puedes crear una nueva con el comando `use`. MongoDB creará la base de datos si no existe.

```
use mi_basededatos
```

4

**Para crear una colección.**

En MongoDB, los datos se almacenan en colecciones. Puedes crear una nueva colección utilizando el método `db.createCollection()` o simplemente insertando un documento en una nueva colección.

```
db.mi_coleccion.insert({ campo1: "valor1", campo2: "valor2" })
```

5

**A la hora de trabajar con documentos.**

- Si quieras insertar un documento:

```
db.mi_coleccion.insert({ campo1: "nuevo_valor" })
```

- Si quieres consultar documentos:

```
db.mi_coleccion.find()
```

- Para actualizar documentos:

```
db.mi_coleccion.update({ campo1: "valor_original" }, { $set: { campo1: "nue
```

- Para eliminar documentos:

```
db.mi_coleccion.remove({ campo1: "valor_a_eliminar" })
```

**(i) Aquí te dejo los enlaces para trabajar con MongoDB.**

Página [documentación](#) MongoDB.

Para [instalarlo](#).

Como [utilidad gráfica](#), puedes utilizar Compass.

# Otras bases de datos: las vectoriales



Qualentum Lab

Las bases de datos vectoriales desempeñan un papel importante en el campo de la **inteligencia artificial (IA)** y el **aprendizaje automático (ML de Machine Learning)** en diversas formas.

A continuación, te describimos algunos casos en los que se utilizan habitualmente las bases de datos vectoriales.

## Procesamiento de imágenes

En aplicaciones de visión por ordenador, las imágenes suelen representarse como matrices de píxeles. Estas matrices pueden convertirse en vectores y almacenarse en bases de datos vectoriales para su procesamiento. Las bases de datos vectoriales ayudan a organizar y recuperar imágenes para tareas como reconocimiento de objetos, detección de rostros, análisis de imágenes médicas y más.

## Análisis de texto y NLP (procesamiento del lenguaje natural)

En NLP, las palabras y las frases se pueden representar como vectores en espacios vectoriales semánticos. Esto permite realizar operaciones matemáticas en palabras y documentos para tareas como análisis de sentimientos, traducción automática y recuperación de información.

## Análisis geoespacial en ML

Para aplicaciones de ML que involucran datos geoespaciales, como la predicción de precios de bienes raíces o la detección de anomalías en rutas de transporte, las bases de datos vectoriales almacenan y facilitan la recuperación de datos geoespaciales.

## Aprendizaje automático no supervisado

Las bases de datos vectoriales pueden utilizarse para entrenar algoritmos de aprendizaje automático no supervisado, como *clustering* y reducción de dimensionalidad. Los datos vectoriales pueden representar características de entrada de modelos de ML.

## Recomendación personalizada

Las bases de datos vectoriales se utilizan en sistemas de recomendación para representar perfiles de usuarios y elementos recomendados. Esto permite realizar cálculos de similitud y proporcionar recomendaciones personalizadas.

## Aprendizaje profundo

En redes neuronales profundas, las representaciones vectoriales, como las incrustaciones (*embeddings*), se utilizan para codificar características en vectores que se pueden alimentar a la red neuronal. Las bases de datos vectoriales pueden ser útiles para almacenar y gestionar estas representaciones vectoriales.

## Procesamiento de series temporales

Para aplicaciones de ML que involucran series temporales, como pronóstico de demanda y detección de anomalías, las bases de datos vectoriales pueden utilizarse para almacenar registros de series temporales y sus características.

En resumen, podemos destacar lo siguiente:

- Las bases de datos vectoriales desempeñan un papel fundamental en la preparación y gestión de datos que se utilizan en aplicaciones de IA y ML.
- Almacenar datos en formato vectorial facilita la manipulación, la consulta y el procesamiento de datos para entrenar modelos y realizar análisis avanzados.
- Las bases de datos geoespaciales, en particular, son esenciales para aplicaciones de IA y ML relacionadas con datos geográficos y de ubicación.

¡Enhорabuena! Fastbook superado



[Qualentum.com](http://Qualentum.com)