Concepto de Bases de Datos

Objetivos:

- Entender qué es una base de datos
- Identificar ejemplos reales del uso de bases de datos
- Comprender cómo se organizan los datos en una base de datos

Todos utilizamos datos y bases de datos en nuestra vida diaria. Por ejemplo, nuestro sistema de archivos en una carpeta en nuestro computador, subir fotos a nuestras redes sociales, descargar archivos en el trabajo y jugar en línea son ejemplos de uso de la base de datos.

¿Qué son exactamente los datos y cómo interactúan con la base de datos?

Volvamos a la primera de nuestras preguntas**. ¿Qué son los datos?** En términos básicos, los datos son hechos y cifras sobre cualquier cosa. Por ejemplo, si se recopilaran datos de una persona, esos datos podrían incluir su nombre, edad, correo electrónico y fecha de nacimiento.

Podría ser el número de pedido, la descripción, la cantidad del pedido y la fecha; incluso, el correo electrónico del cliente. Los datos son cruciales para los individuos y para las organizaciones.

¿Dónde se almacenan todos estos datos?

En nuestro mundo digital, los datos ya no se almacenan en archivos manuales. En su lugar, los desarrolladores utilizan algo llamado bases de datos. Una base de datos es una forma de almacenamiento electrónico en el que se organizan los datos de forma sistemática. Almacena y manipula datos electrónicamente para que sean más manejables, eficaces y seguros. Existen muchos ejemplos reales del uso de las bases de datos. Por ejemplo:

- Un negocio puede utilizar una base de datos para almacenar los datos de sus clientes, cuentas y transacciones bancarias.
- Una universidad utiliza una base de datos para almacenar los datos de los estudiantes, datos de los docentes, datos de los programas y mucho más.

¿cómo es realmente una base de datos?

Bueno, una base de datos se asemeja a datos organizados sistemáticamente. Por lo general esta organización se parece a una hoja de cálculo o tabla.

¿Qué significa exactamente el término sistemático?

Todos los datos contienen elementos o características y atributos a partir de los cuales se pueden identificar. Por ejemplo, una persona puede identificarse por atributos, tales como su edad, altura o color de cabello. Estos datos están separados y se almacenan en entidades que representan esos elementos.

Una entidad es como una tabla. Contiene filas y columnas que almacenan datos relacionados con un elemento específico.

Order ID	Customer ID	Order Date	Delivery Date
01	C1	02-03-2022	07-03-2022
02	C2	02-03-2022	10-03-2022
03	C3	02-03-2022	14-03-2022
04	C4	02-03-2022	08-03-2022
05	C5	02-03-2022	22-03-2022

Ejemplo de Entidad

Un **atributo** es una característica o propiedad que describe un objeto o entidad dentro de la base de datos. Cada atributo corresponde a una columna en una tabla de la base de datos.

Por ejemplo, en una tabla de clientes, los atributos podrían incluir el nombre del cliente, la dirección, el número de teléfono, etc. Los atributos son esenciales para almacenar y organizar los datos de manera estructurada.

En otras palabras, son elementos relacionales. Están relacionados entre sí. Estas entidades pueden ser representaciones físicas, como un empleado, un cliente o un producto. O podrían ser conceptuales, como un pedido, una factura o un presupuesto. Las entidades almacenan los datos en formato de tabla contra los atributos o las características relacionadas con el elemento.

Por ejemplo, una tienda online podría tener los datos de los clientes en una entidad cliente que contiene atributos específicos relativos al cliente. Estos atributos podrían incluir

nombre, apellido, fecha de nacimiento y correo electrónico. También podrían tener datos de productos almacenados en una entidad de producto con atributos como el código de producto, la descripción, el precio y la disponibilidad.

En el mundo de bases de datos relacionales, estas entidades se conocen como relaciones o tablas. Los atributos se convierten en las columnas de la tabla.

Cada fila de la tabla representa una instancia de esa entidad.

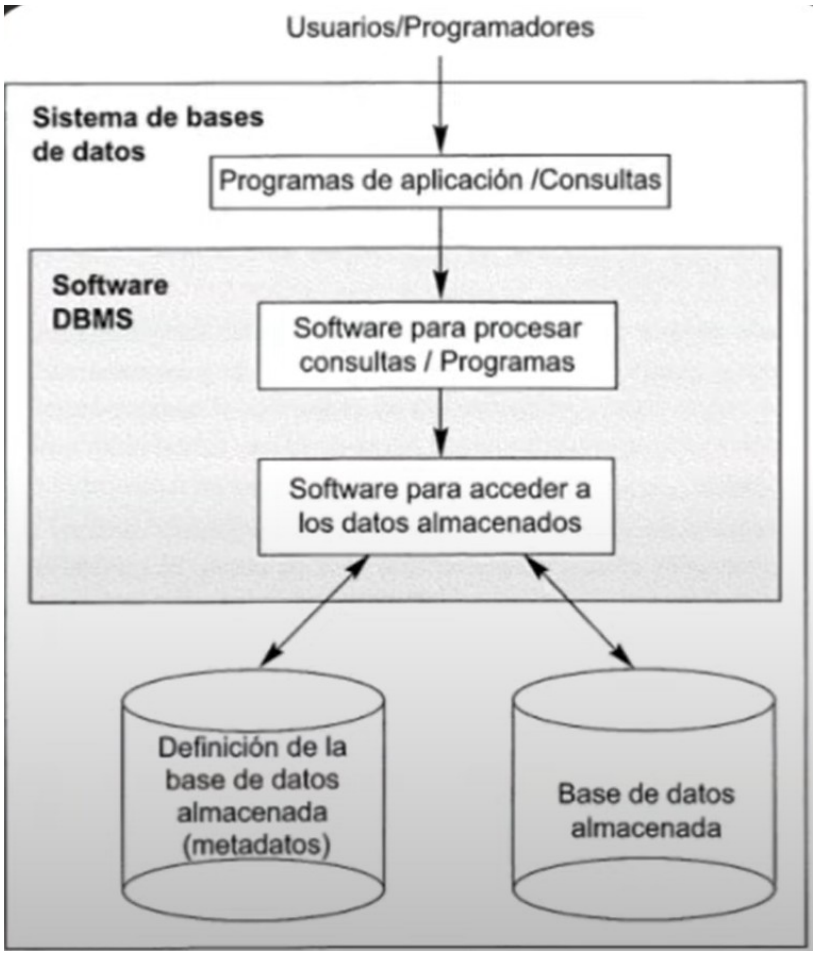
Por ejemplo, veamos las entidades de la tienda en línea que acaba de analizar. Estos dos ejemplos podrían combinarse en una lista de pedidos que la tienda recibía de sus clientes.

Dentro de una base de datos, estos datos podrían representarse como una tabla de pedidos o entidad. Los datos pueden estar organizados en filas con un número de pedido único, el nombre del cliente que realizó el pedido, el producto que pidió y el precio de ese producto.

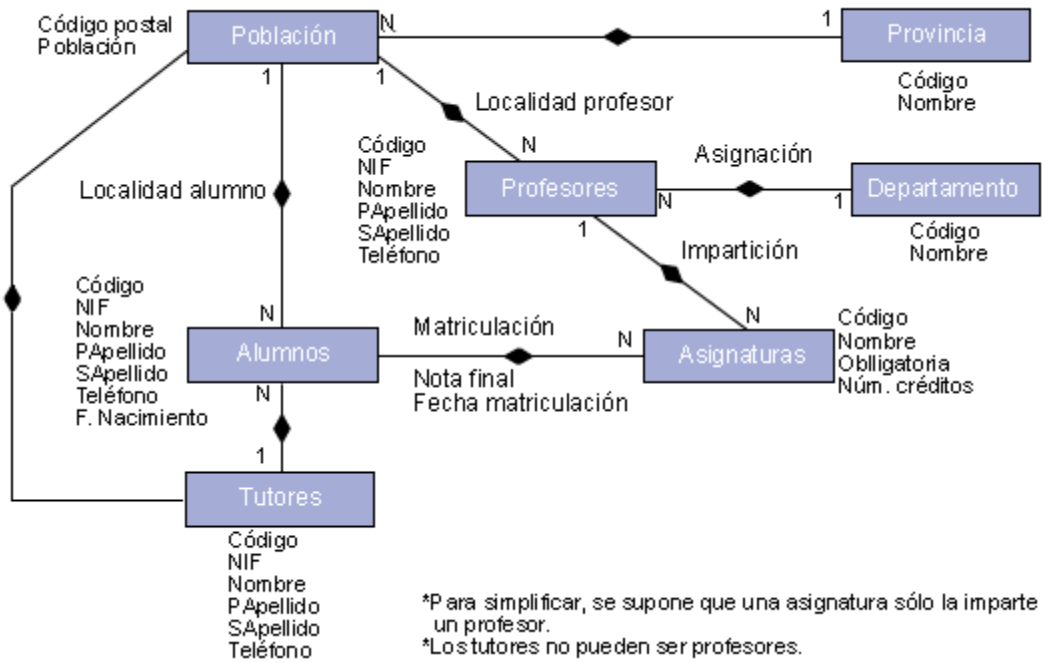
Numero de Pedido	Nombre Cliente	Producto	Precio
1	Marcela Hernandez	Leche en bolsa 1 litro	$12,000.00
2	Adriana Jimenez	Arroz kilo	$ 4,500.00
3	Samuel Molina	Panela	$ 2,300.00
4	Nidia Fernandez	Jabon lavaloza	$ 3,800.00
5	Ana Aguirre	Frijoles libra	$ 8,000.00

Flujo de una base de datos

La capa más externa son los usuarios o programadores que van a acceder a la DB y la ultima capa o ultimo nivel es en si la base de datos.



Ejemplo de abstracción de una base de datos



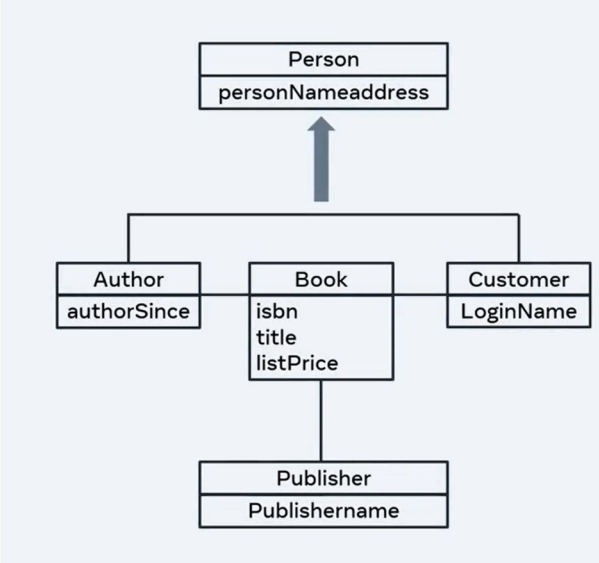
Formas de organizar los datos en una base de datos.

Las bases de datos relacionales no son las únicas de bases de datos que encontraremos. He aquí algunos ejemplos comunes de otros tipos de bases de datos.

Base de datos orientada a objetos es el lugar en donde se almacenan los datos en forma de objetos en lugar de tablas o relaciones.

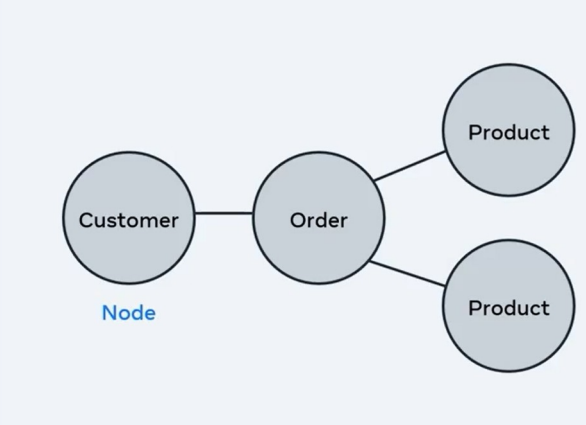
Un ejemplo de este tipo de base de datos podría ser una librería online. La base de datos de la tienda podría representar autores, clientes, libros y editores, como clases, como conjuntos o categorías. Los objetos o instancias de estas clases entonces tendrían los datos reales.

Object oriented databases
Data stored in the form of objects



Bases de datos gráficas almacenan datos en forma de nodos. En este caso, entidades como los clientes, pedidos y productos se representan como nodos. Las relaciones entre ellos se representan como aristas.

Graph databases
Data stored in the form of nodes



Por último, hay bases de datos de documentos donde se almacenan los **datos como objetos de JSON** o notación de objetos de JavaScript. Los datos se organizan en colecciones como tablas. Dentro de cada colección hay documentos escritos en JSON que registran datos.

En este ejemplo, los documentos del cliente se mantienen en una colección del cliente mientras se almacenan los documentos del pedido de productos en las colecciones del pedido de productos.

¿Dónde se almacenan las bases de datos?

Una base de datos puede alojarse en una máquina exclusiva dentro las instalaciones de una organización o empresa, o se podría alojar en la nube en servidores de Amazon, GCP, Azure entre otros. Actualmente, las bases de datos en la nube son una opción más popular. Esto se debe a que permiten almacenar, administrar y recuperar datos a una plataforma en la nube, y acceder a los datos a través de Internet.

Todos ofrecen una opción de costo de menor para la gestión de datos, y otras opciones similares.

Practica:

https://demo.phpmyadmin.net/master-config/public/

https://onecompiler.com/mysql

Motores de Bases de Datos en la Industria

Hasta ahora hemos visto que existen diferentes formas de organizar los datos: en tablas, en objetos, en grafos y en documentos. Pero, **¿qué herramientas reales utilizan las empresas para implementar estas organizaciones?**

En el mundo profesional, no se habla solo de "base de datos relacional" o "base de datos NoSQL" en abstracto. Se habla de **motores** o **sistemas de gestión** específicos, cada uno con sus fortalezas, licencias y casos de uso. A continuación exploramos los más importantes.

Filosofías

ACID — Transacciones Seguras

ACID es la regla de oro de las **bases de datos SQL**. Cada letra es una promesa:

Letra	**Significado**	**Qué garantiza**
A — Atomicidad - Atomicity	La transacción es indivisible	O se hace TODO, o no se hace NADA. Si falla a la mitad, se deshace todo (rollback).
C — Consistencia - Consistency	Los datos siempre cumplen las reglas	Después de cualquier transacción, la base de datos sigue siendo válida (ej: saldo nunca negativo si así lo definiste).
I — Aislamiento - Isolation	Las transacciones no se entrometen	Si Ana y Beto operan al mismo tiempo, ninguno ve los datos "a medias" del otro.
D — Durabilidad - Durability	Lo confirmado, permanece	Aunque se apague el servidor, lo que ya se guardó no se pierde.

Ejemplo real: Un banco. Si transfieres dinero y se cae el sistema a la mitad, no puedes permitir que tu cuenta se debitó pero el destinatario no recibió nada. ACID hace que todo vuelva a cero.

BASE — Disponibilidad Prioritaria

BASE es la filosofía de las **bases de datos NoSQL distribuidas**. Prioriza que el sistema **nunca se caiga**, aunque los datos no estén perfectamente sincronizados en todos los servidores al mismo instante.

Letra	Significado	Qué garantiza
B — Básicamente Disponible - Basically Available	Siempre responde	El sistema nunca dice "no puedo atenderte". Responde con lo que tenga, aunque no sea lo último.
A — Estado Suave - Soft state	El estado puede variar	Cada servidor puede tener un valor ligeramente diferente por unos segundos.
S — Consistencia Eventual - Eventually consistent	Al final, todos coinciden	Si dejas de hacer cambios, tarde o temprano todos los servidores muestran el mismo dato.

Ejemplo real: El contador de "me gusta" de Instagram. Cuando das like, Instagram no espera a actualizar millones de servidores en todo el mundo antes de mostrarte el corazón rojo. Te responde de inmediato y, en los próximos segundos, los demás servidores se sincronizan. Es aceptable que alguien en otro país vea 99 likes en vez de 100 por un momento.

¿Cuándo usar cada uno?

Situación	Elige	Por qué
Transferencia bancaria	**ACID**	No puede existir dinero "perdido". La consistencia es absoluta.
Contador de visitas web	**BASE**	Que diga 9,999 o 10,001 por segundos no importa. La app no puede caerse.
Inventario de almacén	**ACID**	No puedes vender algo que ya no tienes.

Situación	Elige	Por qué
Feed de noticias / timeline	**BASE**	Es aceptable que un post aparezca 2 segundos tarde. Lo crítico es que cargue siempre.

Bases de Datos Relacionales (SQL)

Las bases de datos relacionales siguen el modelo de tablas, filas y columnas que ya estudiamos. Todas utilizan el lenguaje **SQL** (Structured Query Language) para consultar y manipular datos, aunque cada motor tiene pequeñas variantes en su sintaxis.

MySQL

MySQL es el motor relacional de código abierto más popular del mundo. Fue creado en 1995 y actualmente es mantenido por Oracle. Es la opción predeterminada en la mayoría de servidores web y plataformas de hosting compartido.

Característica	Descripción
Licencia	GPL / Comercial
Lenguaje	SQL estándar
Fortalezas	Fácil de aprender, alta velocidad en lectura, comunidad enorme, compatible con PHP, Python y Java
Debilidades	Menos funciones avanzadas que PostgreSQL, gestión de concurrencia más limitada en versiones antiguas
Usado en	WordPress, Drupal, comercio electrónico, aplicaciones LAMP

https://onecompiler.com/mysql

PostgreSQL

PostgreSQL es considerado el motor relacional avanzado de código abierto. Destaca por su extrema conformidad con los estándares SQL y por permitir extensibilidad.

Característica	Descripción
Licencia	PostgreSQL License (tipo BSD, muy permisiva)
Lenguaje	SQL + PL/pgSQL
Fortalezas	Soporte nativo de JSON/JSONB, índices avanzados (GIN, GiST), replicación lógica, ideal para datos geoespaciales (PostGIS)
Debilidades	Curva de aprendizaje más pronunciada que MySQL, configuración inicial más compleja
Usado en	Sistemas financieros, plataformas SaaS, data warehousing, aplicaciones geoespaciales

https://onecompiler.com/postgresql

SQL Server

Desarrollado por Microsoft, SQL Server está profundamente integrado con el ecosistema .NET, Azure y las herramientas de Business Intelligence de Microsoft.

Característica	Descripción
Licencia	Propietaria (existe edición Express gratuita con limitaciones)
Lenguaje	T-SQL (Transact-SQL)
Fortalezas	Excelente integración con Excel y Power BI, análisis en memoria (OLAP), seguridad a nivel de fila
Debilidades	Costo elevado para ediciones Enterprise, dependencia del ecosistema Windows
Usado en	ERP, CRM, sistemas de reportes empresariales, aplicaciones .NET

https://onecompiler.com/sqlserver

Oracle Database

Es la base de datos relacional más potente y costosa del mercado. Es el estándar en grandes corporaciones que manejan millones de transacciones diarias.

Característica	**Descripción**
Licencia	Propietaria (muy costosa)
Lenguaje	SQL + PL/SQL
Fortalezas	Máxima escalabilidad, clusters activos (RAC), particionamiento sofisticado, recuperación ante desastres
Debilidades	Costo prohibitivo para pequeñas empresas, requiere administrador (DBA) especializado
Usado en	Banca, telecomunicaciones, gobierno, grandes corporaciones

https://onecompiler.com/oracle

MariaDB

MariaDB es un *fork* (derivado) de MySQL creado por el fundador original de MySQL tras la adquisición por Oracle. Mantiene compatibilidad con MySQL pero añade mejoras.

Característica	**Descripción**
Licencia	GPL v2
Lenguaje	SQL
Fortalezas	Totalmente abierto, rendimiento superior en ciertas cargas, cluster nativo (Galera)
Debilidades	Ecosistema de herramientas menor que MySQL
Usado en	Migraciones desde MySQL, aplicaciones web de alto tráfico

https://onecompiler.com/mariadb

Bases de Datos No Relacionales (NoSQL)

Estas bases de datos rompen con el modelo de tablas rígidas. Son ideales cuando los datos no tienen una estructura uniforme, cuando se necesita escalabilidad masiva o cuando las relaciones entre datos son complejas.

MongoDB

MongoDB es la base de datos de documentos más popular. Almacena la información en formato BSON (JSON binario), lo que permite esquemas flexibles.

Característica	**Descripción**
Licencia	SSPL
Lenguaje	MongoDB Query Language (MQL)
Fortalezas	Desarrollo ágil sin esquema rígido, escalabilidad horizontal nativa (sharding), ideal para datos jerárquicos
Debilidades	Transacciones multi-documento más lentas que SQL, consumo de memoria elevado
Usado en	Catálogos de productos, gestión de contenidos, IoT, perfiles de usuario

https://onecompiler.com/mongodb

Redis

Redis no es una base de datos tradicional: es un **almacén de estructuras de datos en memoria**. Funciona como base de datos, caché y broker de mensajes simultáneamente (es un sistema intermediario que recibe mensajes de una aplicación, los almacena temporalmente y los entrega a otra aplicación que los necesita. Funciona como un **correo electrónico programático**: alguien envía un mensaje, el broker lo guarda y se lo entrega al destinatario cuando esté listo para recibirlo.).

Característica	**Descripción**
Licencia	BSD
Lenguaje	Comandos Redis

Característica	Descripción
Fortalezas	Latencia sub-milisegundo, operaciones atómicas, soporta strings, hashes, listas, sets y geolocalización
Debilidades	Datos limitados por la RAM disponible, persistencia no es su función principal
Usado en	Caché de sesiones, contadores en tiempo real, colas de mensajes, leaderboards

https://www.youtube.com/watch?v=hRHM13uFpCE

https://onecompiler.com/redis

Apache Cassandra

Cassandra es una base de datos distribuida orientada a columnas. Fue creada en Facebook y diseñada para soportar escrituras masivas sin puntos únicos de fallo.

Característica	Descripción
Licencia	Apache 2.0
Lenguaje	CQL (Cassandra Query Language, similar a SQL)
Fortalezas	Escalabilidad lineal, sin maestro único, replicación entre datacenters, tolerancia a particiones
Debilidades	Modelado de datos restrictivo (diseño query-first), operación compleja
Usado en	Mensajería (Netflix, Instagram), series temporales, logging masivo

https://onecompiler.com/cassandra

Neo4j

Neo4j es la base de datos de grafos más conocida. En lugar de tablas, almacena nodos y relaciones, optimizando consultas sobre datos altamente conectados.

Característica	Descripción
Licencia	GPL v3 / Comercial
Lenguaje	Cypher
Fortalezas	Consultas de relaciones extremadamente rápidas, modelo intuitivo para redes, algoritmos de grafos integrados
Debilidades	No ideal para datos no conectados, escalabilidad horizontal más compleja
Usado en	Redes sociales, detección de fraude, motores de recomendación

¿Cómo elegir entre SQL y NoSQL?

La elección depende de qué se necesite priorizar:

Característica	Relacional (SQL)	No Relacional (NoSQL)
Modelo de datos	Tablas con esquema fijo	Documentos, grafos, clave-valor, columnas (esquema flexible)
Lenguaje	SQL estandarizado	Cada motor tiene su propia API
Escalabilidad	Vertical (más potencia en un servidor)	Horizontal (más servidores distribuidos)
Integridad	ACID (transacciones seguras)	BASE (disponibilidad prioritaria)
Casos típicos	Banca, ERP, inventario, finanzas	Redes sociales, IoT, análisis en tiempo real
Ejemplos	MySQL, PostgreSQL, SQL Server, Oracle, MariaDB	MongoDB, Redis, Cassandra, Neo4j, DynamoDB

> **Regla práctica:** Si tus datos tienen una estructura clara y estable, y las transacciones deben ser 100% confiables (como en un banco), elige **SQL**. Si tus

datos cambian de forma constante, necesitas crecer rápidamente o manejas volúmenes masivos de información no estructurada, considera **NoSQL**.