**Curso Análisis y Teorías de Bases de Datos**

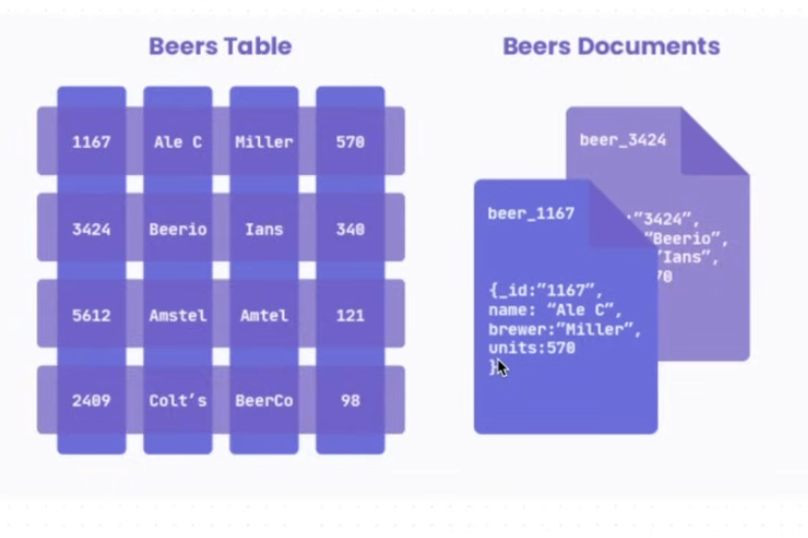
Las bases de datos han existido desde tiempos inmemoriales; fueron creadas para guardar datos. Sin embargo, en la actualidad es importante diferenciar los términos fundamentales:

* **Datos:** Cualquier cosa puede considerarse un dato.
* **Información:** Es un conjunto de datos organizados de manera que tienen significado.
* **Base de datos:** Son los datos almacenados en cualquier lugar, ya sea en papel, libretas o dispositivos digitales.
* **Sistema Gestor de Bases de Datos (DBMS):** Son sistemas de administración como MySQL, MongoDB o cualquier otra aplicación. Estos gestores, conocidos también como SGBD (Sistemas de Gestión de Bases de Datos) o DBMS (Database Management System), se encargan de procesar las búsquedas y gestionar los datos.

**Tipos de Sistemas de Gestión de Bases de Datos**

Existen dos tipos principales:

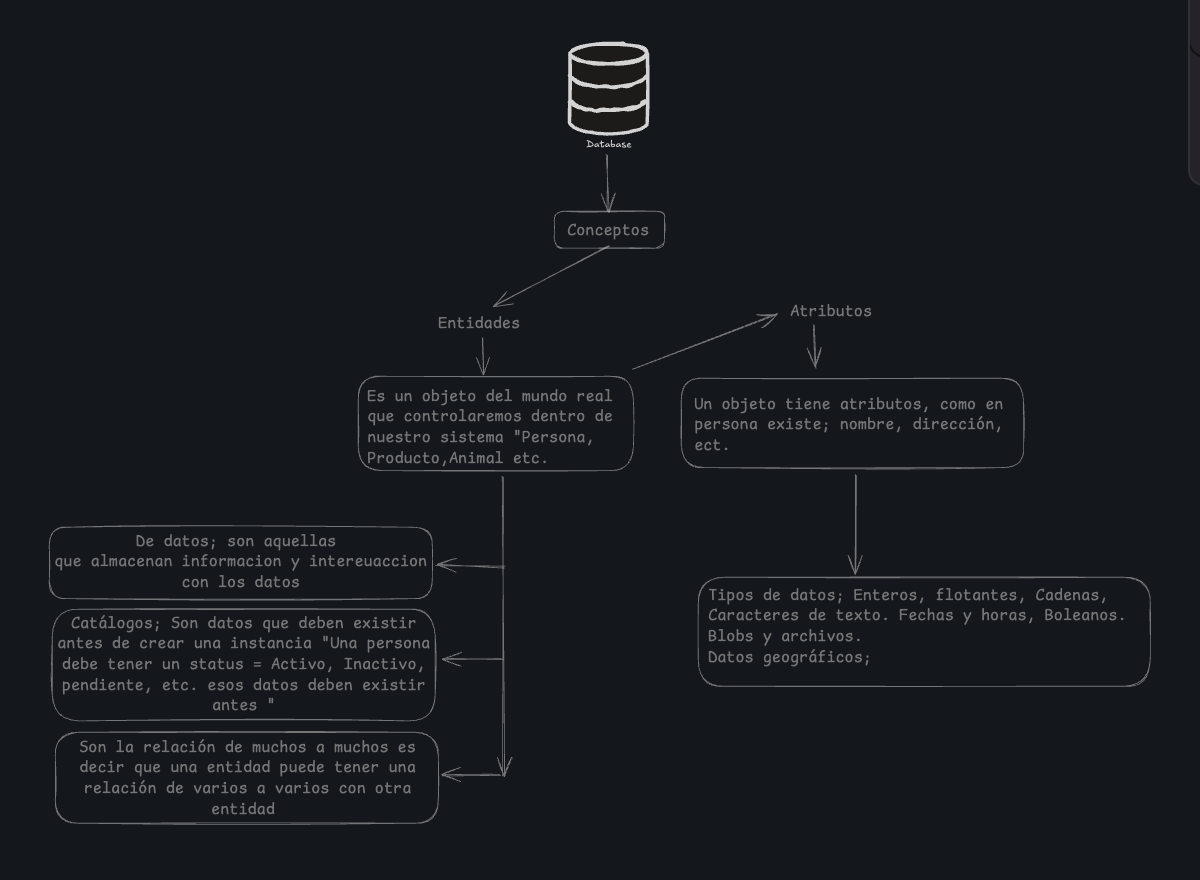
1. **Sistemas Relacionales (SQL):**
   * Se caracterizan por organizar la información en tablas formadas por filas y columnas.
   * Para consultar datos, se utiliza el lenguaje SQL (Structured Query Language), que permite realizar relaciones teniendo en cuenta la estructura de los datos según sus identificadores.
   * Evitan la duplicidad de informacion
2. **Sistemas No Relacionales (NoSQL):**
   * Son más simples y cada entidad funciona de manera independiente.
   * En estas bases de datos, puede existir redundancia de datos, ya que priorizan la velocidad de acceso y procesamiento.
   * Los formatos más comunes son los basados en documentos, como el diccionario clave-valor o el formato JSON.



**¿Cuándo usar bases de datos relacionales o no relacionales?**

* **Bases de datos relacionales (SQL):**  
  Son ideales cuando desarrollamos aplicaciones que requieren un alto nivel de organización, validaciones estrictas y relaciones bien definidas entre los datos. Por ejemplo:
  + Una aplicación de contabilidad.
  + Un sistema de gestión de clientes (CRM).  
    Estas bases de datos son excelentes para casos en los que los datos mantienen una estructura fija y deben respetar reglas claras, como tipos de datos específicos o integridad referencial.
* **Bases de datos no relacionales (NoSQL):**  
  Son más adecuadas cuando los datos no tienen una estructura fija o pueden variar considerablemente. Por ejemplo:
  + Guardar estadísticas de un sitio web.
  + Una galería de fotos, donde los usuarios pueden subir un número indeterminado de imágenes.
  + Se utiliza mucho en big data ya que son muhos datos.
  + Aplicaciones en tiempo real como la ubicación de un objeto por la velocidad de consulta.
  + Progreso de un jugador en un videojuego, ya que esta información generalmente no se relaciona con otros datos y solo se almacena para consultas futuras.

En casos hipotéticos donde sea necesario almacenar información con validaciones estrictas y datos que rara vez cambiarán, lo más recomendable es optar por bases de datos SQL. Estas están diseñadas para garantizar consistencia, integridad y facilidad en la gestión de datos estructurados.



**CRUD: Operaciones fundamentales en bases de datos**

CRUD hace referencia a las operaciones básicas que se realizan en una base de datos:

* **Create (Crear):** Se encarga de insertar nuevos registros en la base de datos.
* **Read (Leer):** Permite consultar y obtener información almacenada en la base de datos.
* **Update (Actualizar):** Modifica registros existentes en la base de datos.
* **Delete (Eliminar):** Borra registros de la base de datos.

Estas operaciones son fundamentales en el manejo de bases de datos y son la base para cualquier aplicación que interactúe con información almacenada.

Es importante incorporar **lógica de negocio** antes de realizar estas operaciones. Esto asegura que los datos cumplen con las reglas y validaciones necesarias antes de ser almacenados o manipulados en la base de datos, garantizando consistencia y calidad en la información.

**Claves en bases de datos**

Las claves son elementos esenciales en bases de datos relacionales que permiten identificar y organizar los datos correctamente.

**1. Clave primaria (Primary Key):**

* Es un identificador único para cada registro dentro de una tabla.
* **Características:**
  + No puede repetirse (valores únicos).
  + No puede ser nula.
  + Se utiliza para identificar cada fila de manera inequívoca.
* **Ejemplo práctico:** En una tabla de clientes, el campo que almacena el número de identificación de cada cliente sería la clave primaria, ya que cada cliente debe tener un identificador único.

**2. Clave foránea (Foreign Key):**

* Es un campo en una tabla que se utiliza para establecer relaciones con otra tabla.
* **Características:**
  + Hace referencia a la clave primaria de otra tabla.
  + Garantiza que los valores registrados en el campo foráneo existan en la tabla relacionada, manteniendo la integridad de los datos.
* **Ejemplo práctico:** En una tabla de órdenes o pedidos, el campo que indica qué cliente realizó el pedido sería una clave foránea, vinculando el pedido con la tabla de clientes.

**3. Clave única (Unique Key):**

* Se utiliza para asegurar que los valores en una columna (o conjunto de columnas) sean únicos en toda la tabla.
* **Características:**
  + No permite valores duplicados, pero sí permite un único valor nulo.
  + Es útil cuando necesitas evitar que un dato se repita, pero no es la clave principal de la tabla.
* **Ejemplo práctico:** En una tabla de clientes, el campo de correo electrónico podría tener una restricción de clave única para evitar que dos clientes utilicen el mismo correo.

**Tipos de relaciones en bases de datos**

**1. Relación 1 a 1 (Uno a Uno):**

* Cada registro de una tabla está relacionado con un único registro en otra tabla, y viceversa.
* **Ejemplo:**
  + Una persona y su pasaporte. Cada persona tiene un único pasaporte, y cada pasaporte pertenece a una sola persona.

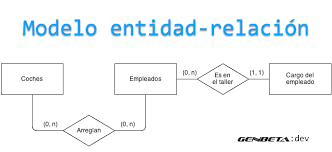
**2. Relación 1 a M (Uno a Muchos):**

* Un registro de una tabla está relacionado con múltiples registros en otra tabla, pero cada registro en la segunda tabla está relacionado con un único registro en la primera.
* **Ejemplo:**
  + Un cliente puede realizar múltiples pedidos, pero cada pedido pertenece a un único cliente.

**3. Relación M a M (Muchos a Muchos):**

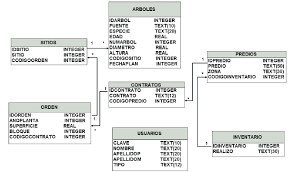
* Varios registros de una tabla pueden estar relacionados con varios registros en otra tabla.
* **Ejemplo:**
  + Los estudiantes pueden inscribirse en varios cursos, y cada curso puede tener múltiples estudiantes.

**Diagrama Entidad-Relación (ER):**

* Es una representación gráfica que utiliza figuras y símbolos para modelar la lógica del negocio.
* Sirve para visualizar las entidades (tablas), sus atributos y las relaciones entre ellas, lo que permite comprender mejor las acciones y operaciones que se llevarán a cabo en la base de datos.
* **Ejemplo:** Una entidad podría ser "Clientes" y otra "Pedidos", conectadas por una relación que indica que un cliente puede realizar múltiples pedidos. 

**Modelo Relacional de la Base de Datos:**

* Es un modelo lógico que organiza los datos en tablas (o relaciones) compuestas por filas (tuplas) y columnas (atributos).
* Este modelo puede aplicarse tanto a sistemas de bases de datos relacionales (SQL) como a bases de datos no relacionales (NoSQL) cuando los datos pueden estructurarse de esta forma.
* En sistemas relacionales, el modelo sigue reglas estrictas para garantizar la consistencia e integridad de los datos.



**Normalización de bases de datos**

La normalización es el proceso de organizar los datos en una base de datos para reducir la redundancia y mejorar la integridad de los datos. Existen varias formas normales (NF), cada una con reglas específicas que aseguran una mejor estructura de las tablas.

**Primera Forma Normal (1NF):**

* Los valores en las columnas deben ser **atómicos**, es decir, indivisibles.
* **Ejemplo:**  
  Si tienes una columna para "Dirección" que incluye calle, número y código postal juntos, deberías dividirla en varias columnas: Calle, Número y Código Postal.

**Segunda Forma Normal (2NF):**

* Cumple con la 1NF.
* No debe haber dependencias parciales, es decir, ningún atributo no clave debe depender solo de una parte de la clave primaria.
* **Ejemplo:**  
  Si tienes un campo llamado "Rol" (por ejemplo, Profesor, Estudiante, Administrador), este debe moverse a una tabla separada llamada Roles. Allí, cada rol se relaciona con los usuarios a través de una clave foránea.

**Tercera Forma Normal (3NF):**

* Cumple con la 2NF.
* No debe haber **dependencias funcionales transitivas**, es decir, un atributo no clave no debe depender de otro atributo no clave.
* **Ejemplo:**  
  Si tienes una tabla que incluye datos de cliente, producto y venta juntos, deberías dividirla en tres tablas:
  + Clientes: Datos del cliente.
  + Productos: Información sobre los productos.
  + Ventas: Relación entre cliente, producto y detalles de la venta.

**Cuarta Forma Normal (4NF):**

* Cumple con la 3NF.
* No debe haber dependencias multivaluadas, es decir, una tabla no debe tener atributos que dependan de múltiples valores independientes.
* **Ejemplo:**  
  Una tabla que almacena información de empleados con sus idiomas y proyectos podría tener redundancia si cada empleado tiene varios idiomas y trabaja en varios proyectos. En este caso, se dividiría en dos tablas:
  + Una para relacionar empleados con idiomas.
  + Otra para relacionar empleados con proyectos.

**Quinta Forma Normal (5NF):**

* Cumple con la 4NF.
* Descompone las relaciones para eliminar dependencias de unión, garantizando que la información pueda ser reconstruida sin pérdida.
* **Ejemplo:**  
  Supongamos que tienes una tabla que relaciona "Proveedores", "Productos" y "Tiendas". Para evitar redundancia, puedes dividirla en tres tablas:
  + Proveedores-Productos: Relaciona proveedores con los productos que suministran.
  + Productos-Tiendas: Relaciona productos con las tiendas que los venden.
  + Proveedores-Tiendas: Relaciona proveedores con las tiendas a las que abastecen

**Pasos para el modelado de una base de datos:**

