RESUMEN DE BASES DE DATOS RELACIONALES. CONCEPTOS BÁSICOS.

1. **Base de Datos Relacional (RDBMS)**

   - Un sistema de gestión de bases de datos que utiliza un modelo relacional para almacenar y organizar los datos en tablas.
   - Ejemplos: **MySQL**, **PostgreSQL**, **Oracle**, **SQL Server**.

2. **Tabla (Table)**

   - Estructura principal en una base de datos relacional donde se almacenan los datos.
   - Está organizada en filas (registros) y columnas (campos).
   - Ejemplo: tabla **`Clientes`** que almacena información de cada cliente de una tienda.

id_cliente	nombre	direccion	telefono
1	Ana	Calle A, 123	555-1234
2	Juan	Calle B, 456	555-5678

3. **Fila (Row)**

   - Cada fila de una tabla representa una **instancia** de datos o un **registro**.
   - Ejemplo:
     - Cada fila representa un **registro** o un cliente único en la tabla `Clientes`.
     - Ejemplo: La primera fila representa al cliente "Ana", que vive en "Calle A, 123" y tiene el teléfono "555-1234".

4. **Columna (Column)**

   - Cada columna define un campo, un tipo específico de dato para los registros de la tabla.
   - Ejemplo:
     - Las columnas en Clientes definen los tipos de información que se almacena sobre cada cliente.
     - Ejemplos de columnas: `id_cliente`, `nombre`, `direccion`, `telefono`.

5. **Clave Primaria (Primary Key)**

   - Un campo o conjunto de campos que identifica de forma unívoca cada fila en una tabla.
   - No puede contener valores nulos y debe ser único para cada fila o registro.
   - Ejemplo:
     - La columna id_cliente en Clientes es una clave primaria porque identifica de forma única a cada cliente. Por ejemplo: `id_cliente`=1 solo puede pertenecer a "Ana"; no puede repetirse en la tabla.

6. **Clave Ajena (Foreign Key)**

   - Un campo en una tabla que hace referencia a la **clave primaria** de otra tabla.
   - Se usa para establecer relaciones entre tablas, como la relación entre un pedido y un cliente.

- Ejemplo:
    - Supongamos que existe una tabla **`Pedidos`** con una columna `id_cliente` que hace referencia a `id_cliente` de la tabla `Clientes`.
    - `id_cliente` en la tabla `Pedidos` es una clave ajena; asegura que cada pedido corresponde a un cliente existente en `Clientes`.

id_pedido	id_cliente	fecha	total
101	1	2024-01-01	200.00

7. Relación (Relationship)

- La forma en que las tablas se vinculan entre sí mediante claves primarias y ajenas.
- Tipos de relaciones:
    - **Uno a uno (1:1)**: Cada registro en una tabla se relaciona con un único registro en otra tabla.
    - **Uno a muchos (1:M)**: Un registro de una tabla se puede relacionar con varios registros en otra tabla.
    - **Muchos a muchos (M:M)**: Varios registros de una tabla se pueden relacionar con varios registros en otra tabla. Normalmente, se resuelve mediante una tabla intermedia.
- Ejemplo:
    - Entre `Clientes` y `Pedidos` se puede establecer una relación **uno a muchos**: cada cliente puede tener varios pedidos. Por ejemplo: El cliente con `id_cliente` 1 ("Ana") puede tener varios registros en la tabla `Pedidos`.

8. Índice (Index)

- Estructura que mejora la velocidad de las operaciones de consulta (SELECT) sobre una tabla.
- Similar al índice de un libro, permite encontrar rápidamente los registros sin tener que escanear toda la tabla.
- Ejemplo: Un índice en `nombre` de la tabla `Clientes` aceleraría las búsquedas por nombre.

9. SQL (Structured Query Language)

- Lenguaje estándar utilizado para interactuar con bases de datos relacionales.
- Permite crear tablas, modificar, consultar y administrar los datos en una base de datos.
- Ejemplo de SQL: Crear la tabla `Clientes`

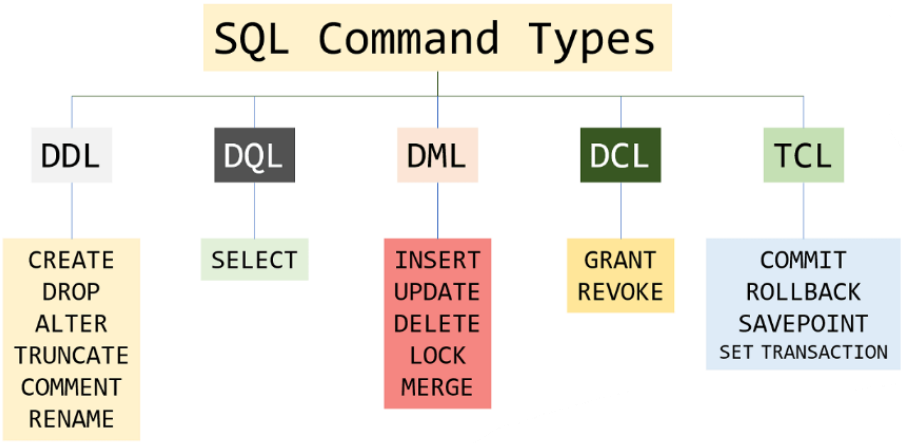
```
CREATE TABLE Clientes (
    id_cliente INT PRIMARY KEY,
    nombre VARCHAR(100),
    direccion VARCHAR(255),
    telefono VARCHAR(15)
);
```

10. Consulta (Query)

- Instrucción en **SQL** (Structured Query Language) para recuperar, insertar, actualizar o borrar datos de una base de datos.
- Ejemplos de consultas básicas:

```
SELECT * FROM Clientes;
SELECT nombre, direccion FROM clientes WHERE id_cliente = 1;
```

11. Clasificación de sentencias SQL



1. DDL (Data Definition Language)
2. DQL (Data Query Language)
3. DML (Data Manipulation Language)
4. DCL (Data Control Language)
5. TCL (Transaction Control Language)

12. Operaciones CRUD

- Las cuatro operaciones básicas de consulta y manipulación
  - **Create**: Insertar datos (INSERT)
  - **Read**: Leer datos (SELECT)
  - **Update**: Actualizar datos (UPDATE)
  - **Delete**: Borrar datos (DELETE)
- Ejemplo de cada operación en la tabla `Clientes`:
  - **Create**:

```
INSERT INTO Clientes (id_cliente, nombre, direccion, telefono)
VALUES (4, 'Laura', 'Calle D, 321', '555-8765');
```

  - **Read**: `SELECT * FROM Clientes;`
  - **Update**: `UPDATE Clientes SET direccion = 'Calle C, 789' WHERE id_cliente = 2;`
  - **Delete**: `DELETE FROM Clientes WHERE id_cliente = 1;`

13. Restricciones (Constraints)

- Reglas que se imponen sobre los datos para garantizar la **integridad** de la base de datos.
- Ejemplos de constraints:
  - **NOT NULL**: Garantiza que una columna no puede contener valores nulos.
  - **UNIQUE**: Asegura que los valores de una columna son únicos.
  - **CHECK**: Define una condición que los valores de una columna deben cumplir.
- Ejemplos de constraints en la tabla cliente:
  - **NOT NULL**: `telefono` en `Clientes` debe tener un valor.
  - **UNIQUE**: `email` en `Clientes` debe ser único.
  - **CHECK**: edad en `Clientes` debe ser mayor que 18.

14. Normalización (Normalization)

- Proceso de organizar los datos en una base de datos para reducir la redundancia y mejorar la integridad de los datos.
- Implica dividir grandes tablas en tablas más pequeñas y relacionarlas entre sí.
- Las **formas normales** son reglas que guían el proceso de normalización (1NF, 2NF, 3NF, etc.).
- Ejemplo: Dividir la tabla `Clientes` en dos tablas (`Clientes` y `Direcciones`) para evitar duplicar datos de dirección.
    - Tabla Clientes: id_cliente, nombre, telefono.
    - Tabla `Direcciones`: `id_cliente`, `direccion`.

15. Transacción (Transaction)

- Conjunto de operaciones que se ejecutan como una unidad indivisible.
- Propiedades de una transacción (ACID):
    - **Atomicidad** (**A**tomicity): Todas las operaciones de la transacción deben completarse correctamente o no realizarse.
    - **Consistencia** (**C**onsistency):La base de datos pasa de un estado válido a otro.
    - **Aislamiento** (**I**solation**)**: Las transacciones se ejecutan de forma independiente.
    - **Durabilidad (D**urability**)**: Los cambios realizados por una transacción persisten, incluso en caso de fallo del sistema.
- Ejemplo: Registrar un pedido en `Pedidos` y actualizar el stock en `Productos` dentro de una transacción:

```
BEGIN TRANSACTION;
INSERT INTO Pedidos (id_pedido, id_cliente, fecha, total) VALUES (101, 1,
'2024-01-01', 200.00);
UPDATE Productos SET stock = stock - 1 WHERE id_producto = 10;
COMMIT;
```

16. Vista (View)

- Una **tabla virtual** que resulta de una consulta SQL. No almacena datos, sino que muestra resultados basados en una consulta.
- Puede usarse para simplificar la consulta de datos complejos o para ocultar detalles de la base de datos.
- Ejemplo: Crear una vista `VistaClientesActivos` que muestra solo los clientes activos

```
CREATE VIEW VistaClientesActivos AS
SELECT id_cliente, nombre FROM Clientes WHERE estado = 'Activo';
```

17. Esquema (Schema)

- Conjunto de tablas, vistas, índices y otros objetos de base de datos que definen la estructura de una base de datos.
- En algunos sistemas de gestión de bases de datos, un esquema también puede hacer referencia a una **base de datos** específica.
- Ejemplo: esquema `Tienda` que agrupa las tablas `Clientes`, `Pedidos`, y `Productos`.

18. Catálogo (catalog)

- Un catálogo es el nivel más alto de la jerarquía de organización de una base de datos. Dentro de un catálogo se encuentran esquemas, y dentro de los esquemas, las tablas y demás objetos.

SGBD	Catalog	Schema
SQLite	No existe el concepto de catálogo.	Existe, pero en un sentido limitado. SQLite tiene un único esquema lógico por base de datos, pero usando ATTACH se pueden conectar múltiples bases de datos y tratarlas como esquemas.
MySQL	Existen catálogos implícitamente. MySQL organiza las tablas, vistas, y demás objetos dentro de lo que se denomina "base de datos". Según el estándar SQL, estas bases de datos también pueden considerarse como catálogos.	Existe, pero "schema" y "database" son términos intercambiables.
Oracle	No existe el concepto de catálogo.	Está vinculado directamente a un usuario. Cada usuario tiene un esquema propio, con el mismo nombre que el usuario.

19. Procedimientos Almacenados (Stored Procedures)

- Conjunto de instrucciones SQL que se almacenan en la base de datos y pueden ser ejecutadas de manera repetida.
- Permiten encapsular lógica compleja de negocio y mejorar el rendimiento.
- Ejemplo: procedimiento para registrar un nuevo cliente en `Clientes`.

```
CREATE PROCEDURE RegistrarCliente (IN nombre VARCHAR(100), IN direccion
                                   VARCHAR(255), IN telefono VARCHAR(15))
BEGIN
    INSERT INTO Clientes (nombre, direccion, telefono) VALUES (nombre,
                          direccion, telefono);
END;
```

20. Disparadores (Trigger)

- Un conjunto de instrucciones SQL que se ejecutan automáticamente en respuesta a un evento en la base de datos, como la inserción, actualización o eliminación de un registro.
- Ejemplo: Un trigger que actualiza la tabla `HistorialPedidos` cuando se inserta un nuevo pedido.

```
CREATE TRIGGER ActualizarHistorial AFTER INSERT ON Pedidos
FOR EACH ROW
BEGIN
    INSERT INTO HistorialPedidos (id_pedido, fecha) VALUES (NEW.id_pedido,
NEW.fecha);
END;
```

21. JOIN

- Operación que permite combinar registros de dos o más tablas basándose en una columna común.
- Tipos comunes:
    - **INNER JOIN**: Devuelve solo las filas que tienen coincidencias en ambas tablas.
    - **LEFT JOIN**: Devuelve todas las filas de la tabla izquierda, con las filas correspondientes de la tabla derecha (si las hay).
    - **RIGHT JOIN**: Devuelve todas las filas de la tabla derecha, con las filas correspondientes de la tabla izquierda (si las hay).
- Ejemplo: JOIN para obtener el nombre del cliente junto con cada pedido en `Pedidos`:

```
SELECT Clientes.nombre, Pedidos.id_pedido, Pedidos.fecha
FROM Clientes
INNER JOIN Pedidos ON Clientes.id_cliente = Pedidos.id_cliente;
```

    - SELECT `Clientes.nombre, Pedidos.id_pedido, Pedidos.fecha`:

      Define las columnas que queremos en el resultado: el nombre del cliente, el ID del pedido y la fecha del pedido.

    - FROM Clientes:

      Indica que la tabla principal para la consulta es `Clientes`.

    - INNER `JOIN Pedidos`:

      Une la tabla `Clientes` con `Pedidos` para combinar información de ambas.

    - ON Clientes`.id_cliente = Pedidos.id_cliente`:

      Especifica la condición de unión: las filas de ambas tablas se combinan solo si el valor de `id_cliente` coincide.