

# AE5 – Relaciones en Bases de Datos Relacionales

Bootcamp Full Stack Python

Módulo 5: Fundamentos de Bases de Datos Relacionales

**Aprendizaje Esperado:** Elaborar un modelo de datos para resolver un problema de baja complejidad.





# Objetivos de la Clase

- 1** Comprender relaciones entre tablas en bases de datos.
- 2** Diferenciar entre 1:1, 1:N y N:M.
- 3** Aplicar ejemplos prácticos en SQL y VS Code.
- 4** Resolver casos reales de modelado de datos.





## ¿Qué son las relaciones en bases de datos?

Las tablas no viven aisladas.

Se conectan entre sí para representar la vida real.

Una relación indica cómo los datos de una tabla se conectan con otra.

**Usuarios ↔ Direcciones**

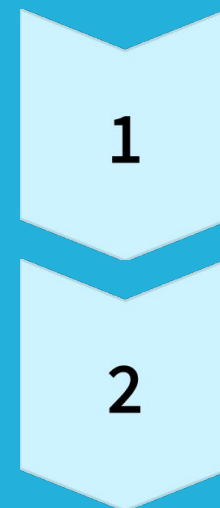
**Clientes ↔ Pedidos**

**Estudiantes ↔ Cursos**



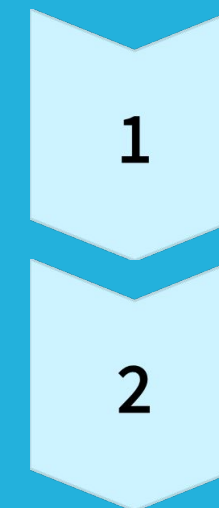
## Relación Uno a Uno (1:1)

Cada registro de una tabla se relaciona con un solo registro en otra tabla.



**Usuario**

**Perfil único**



**Vehículo**

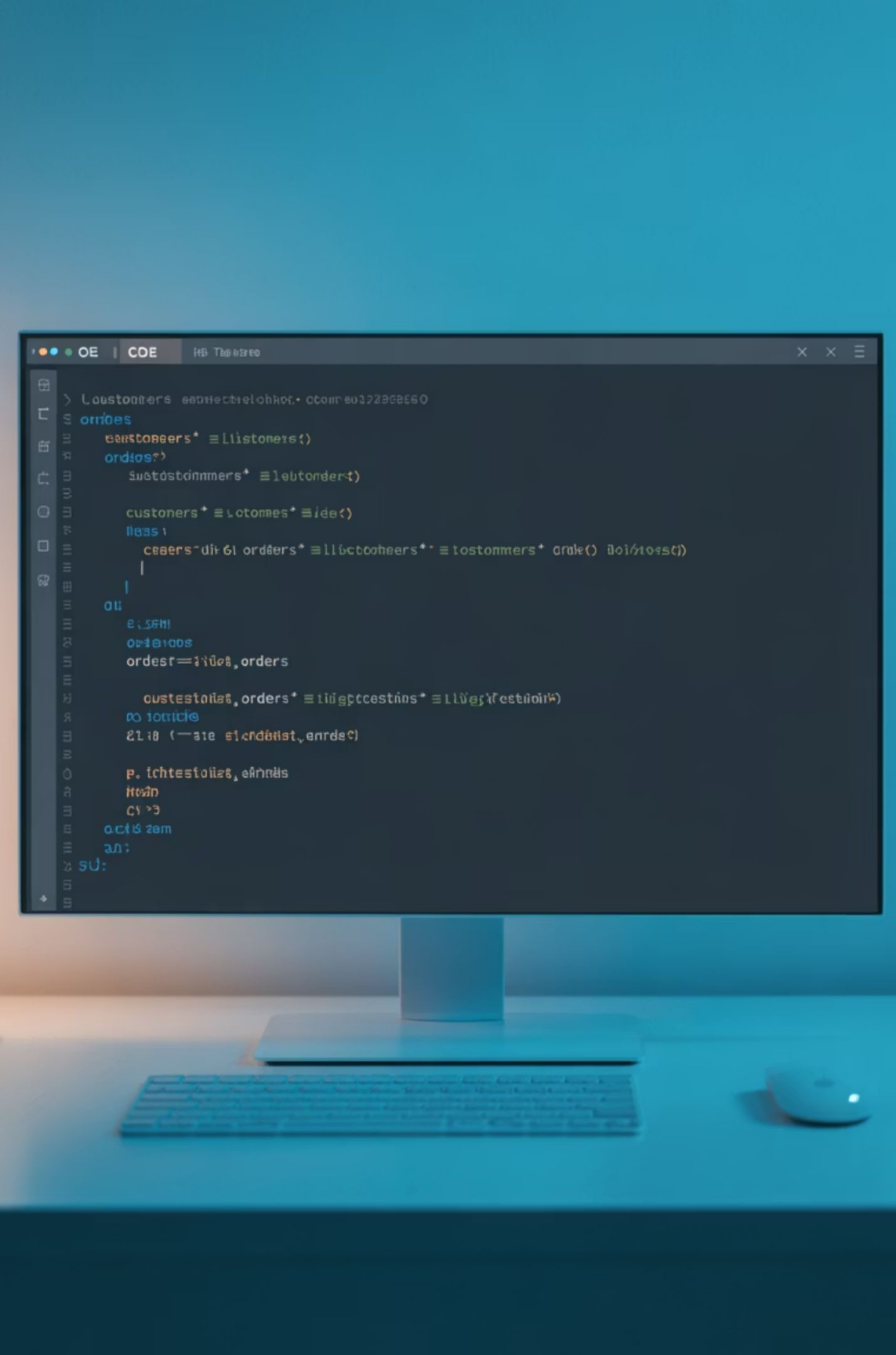
**Seguro vehicular**

# Ejemplo Práctico 1:1

## Tablas:


```
CREATE TABLE Usuario ( id INT PRIMARY KEY, nombre
VARCHAR(100));CREATE TABLE Perfil ( id INT PRIMARY KEY,
direccion VARCHAR(200), usuario_id INT UNIQUE, FOREIGN KEY
(usuario_id) REFERENCES Usuario(id));
```

👉 Cada usuario tiene un perfil único.





## Actividad en Clase 1:1

 **Mini reto: Modela una relación Empleado ↔ Expediente laboral.**

**¿Qué atributos tendría cada tabla?**

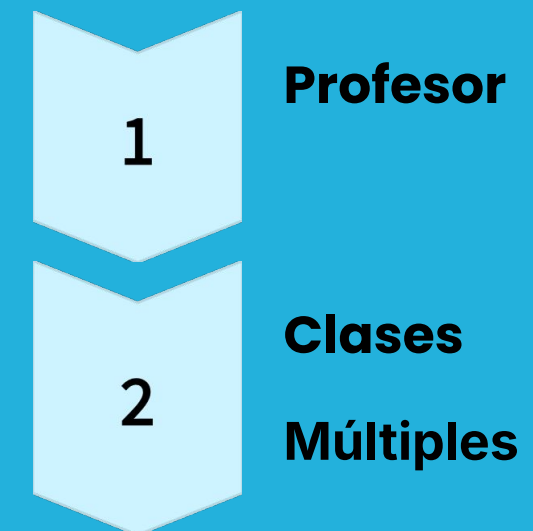
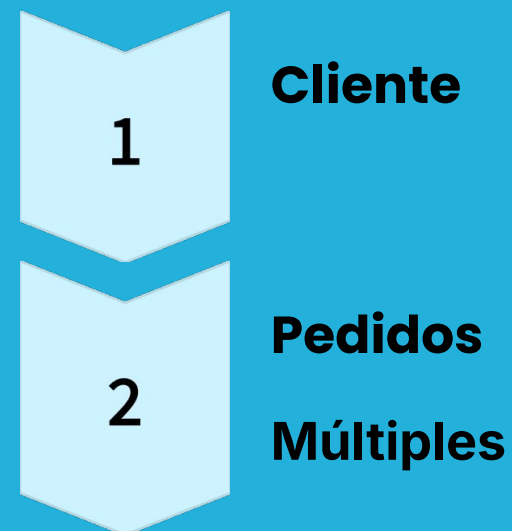
**Piensa en los campos necesarios para cada entidad.**

**¿Qué columna sería la clave foránea?**

**Identifica la columna que conectará ambas tablas.**

# Relación Uno a Muchos (1:N)

Un registro de una tabla se conecta con muchos registros en otra tabla.





## Ejemplo Práctico 1:N

```
CREATE TABLE Cliente ( id INT PRIMARY KEY, nombre  
VARCHAR(100));CREATE TABLE Pedido ( id INT PRIMARY KEY, fecha  
DATE, cliente_id INT, FOREIGN KEY (cliente_id) REFERENCES  
Cliente(id));
```

👉 Un cliente puede tener varios pedidos, pero cada pedido pertenece a un único cliente.





## Ejercicio en Clase 1:N

❓ **Situación: Un Hospital tiene muchos Pacientes, pero cada paciente solo está en un hospital.**

👉 **Diseña las tablas: Hospital y Paciente.**

1

**Hospital**

Piensa en los atributos principales

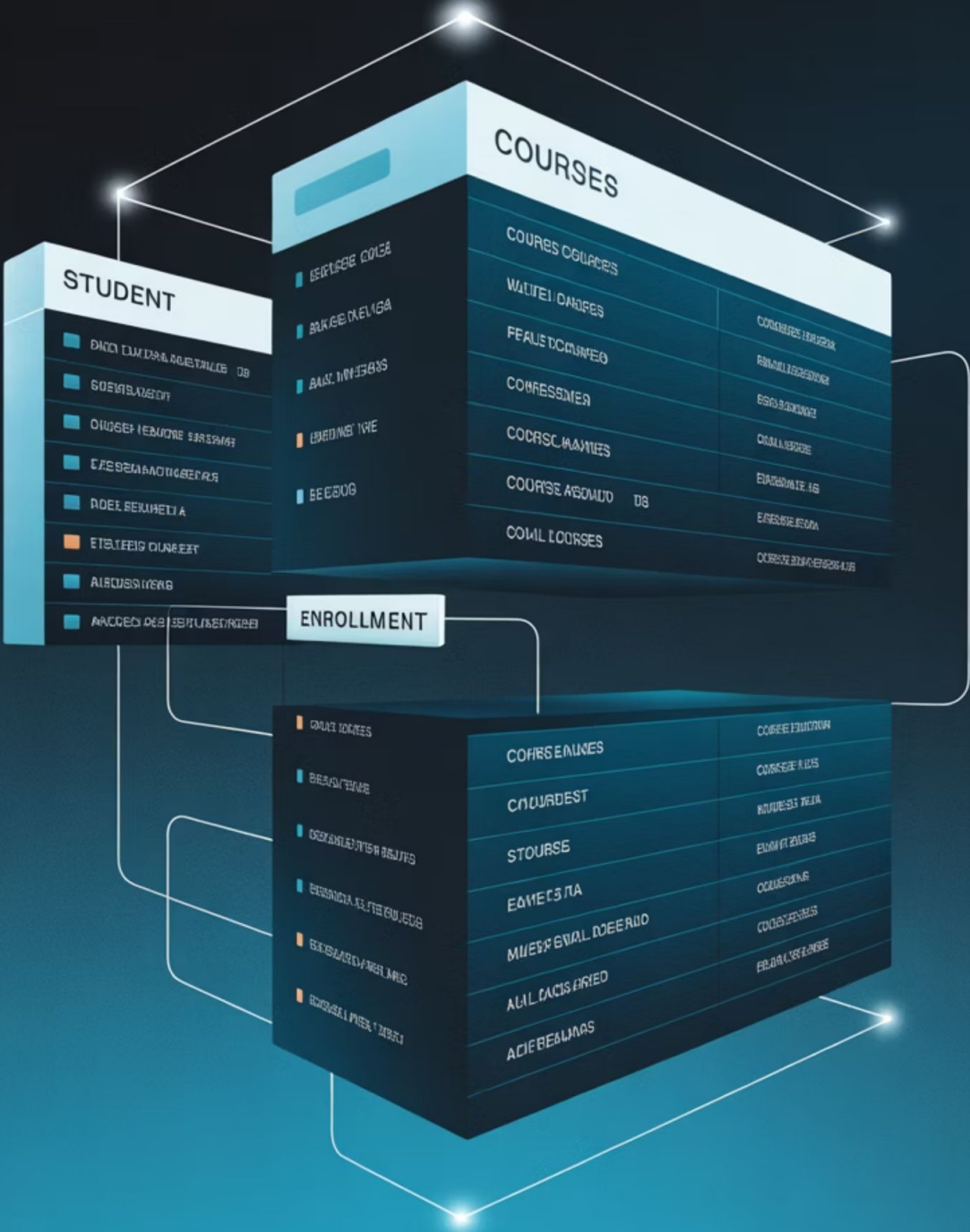
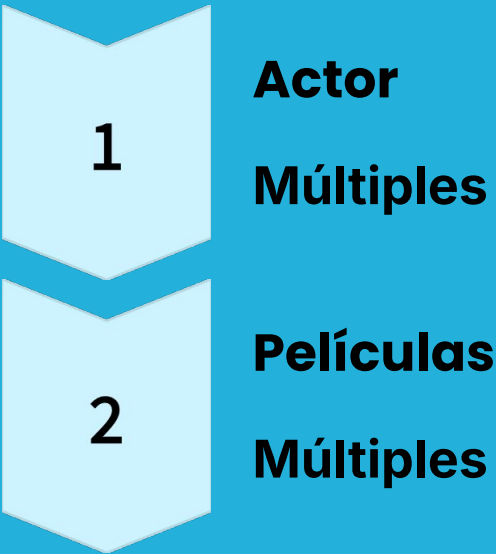
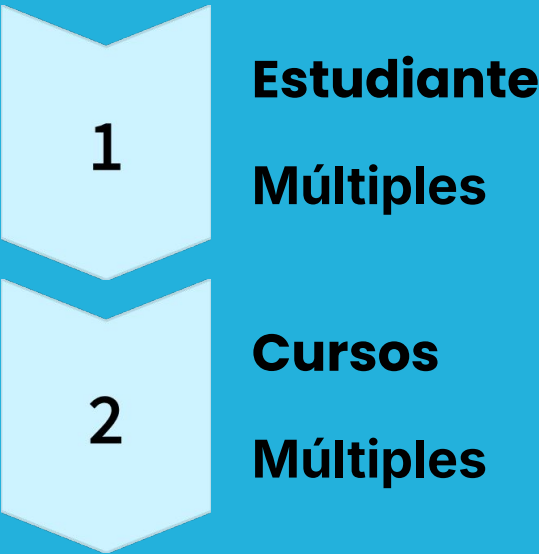
2

**Paciente**

¿Dónde iría la clave foránea?



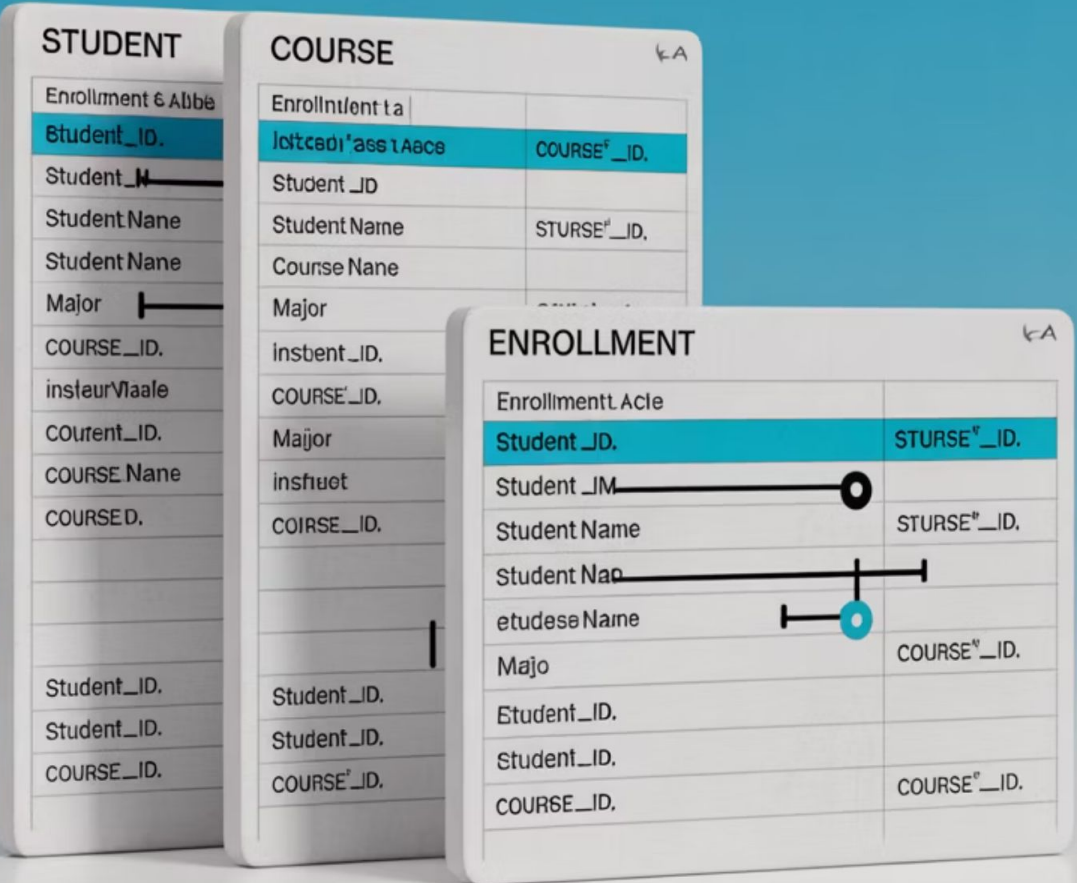
Un registro de una tabla se conecta con muchos registros en otra, y viceversa.





# Ejemplo Práctico N:M

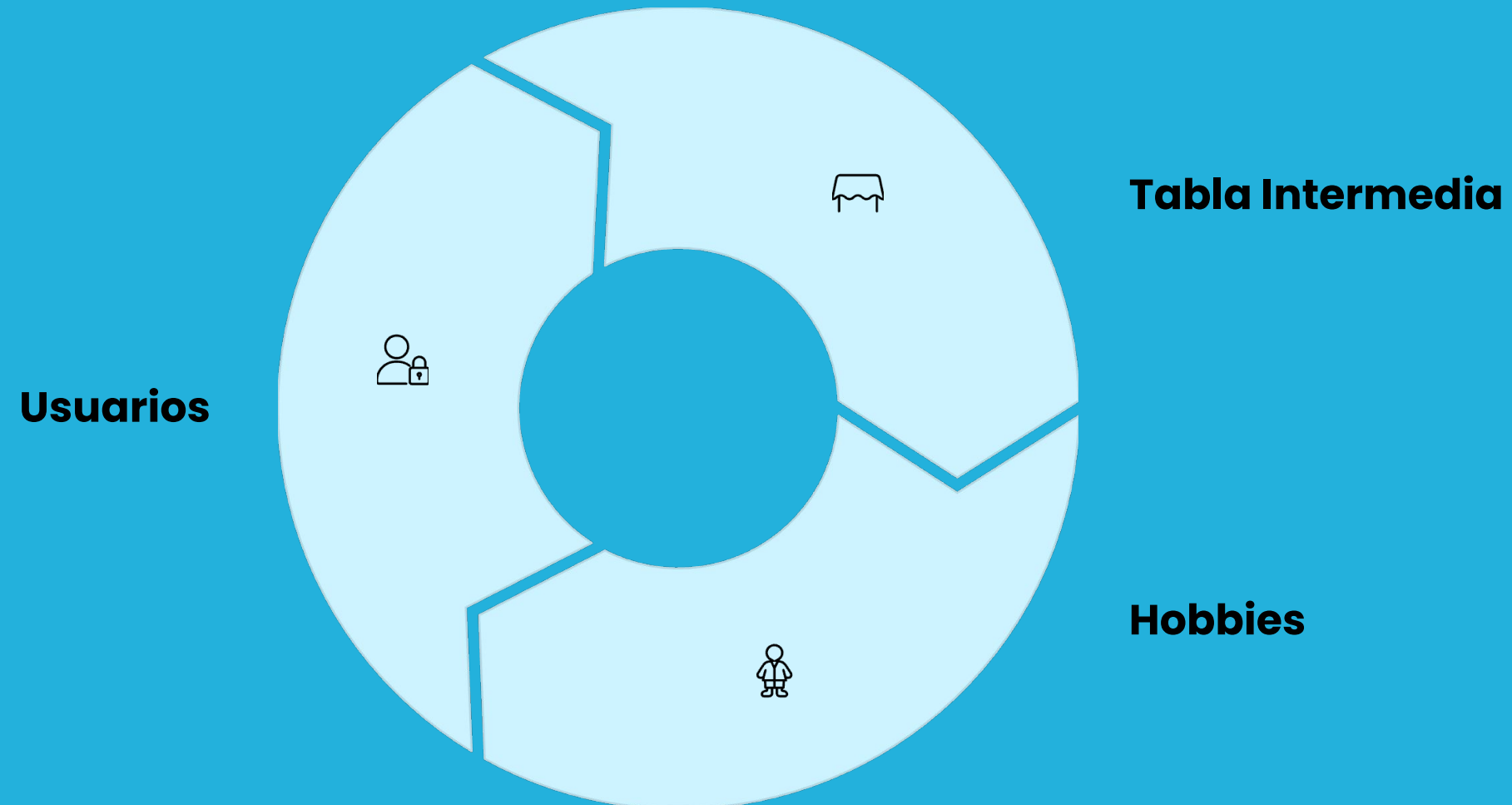
```
CREATE TABLE Estudiante ( id INT PRIMARY KEY, nombre VARCHAR(100));CREATE TABLE Curso ( id INT PRIMARY KEY, titulo VARCHAR(100));CREATE TABLE Incripcion ( estudiante_id INT, curso_id INT, PRIMARY KEY (estudiante_id, curso_id), FOREIGN KEY (estudiante_id) REFERENCES Estudiante(id), FOREIGN KEY (curso_id) REFERENCES Curso(id));
```



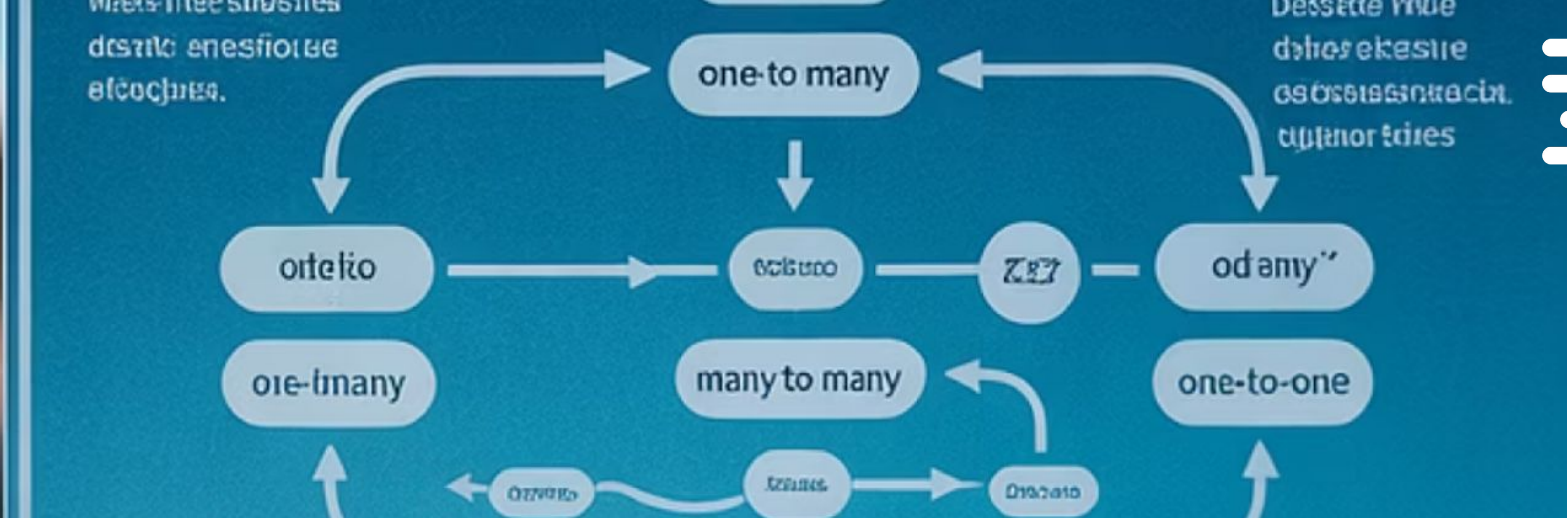
# Mini Reto N:M

 **Actividad: Modela un sistema de Usuarios ↔ Hobbies.**

 Cada usuario puede tener varios hobbies y cada hobby puede pertenecer a muchos usuarios.







# Cómo identificar el tipo de relación

Regla práctica:

**Si es único ↔ único → 1:1.**

**Si un registro tiene varios en otra tabla → 1:N.**

**Si ambos lados tienen varios → N:M.**

## ERD Completo de un E-commerce

Entidades:

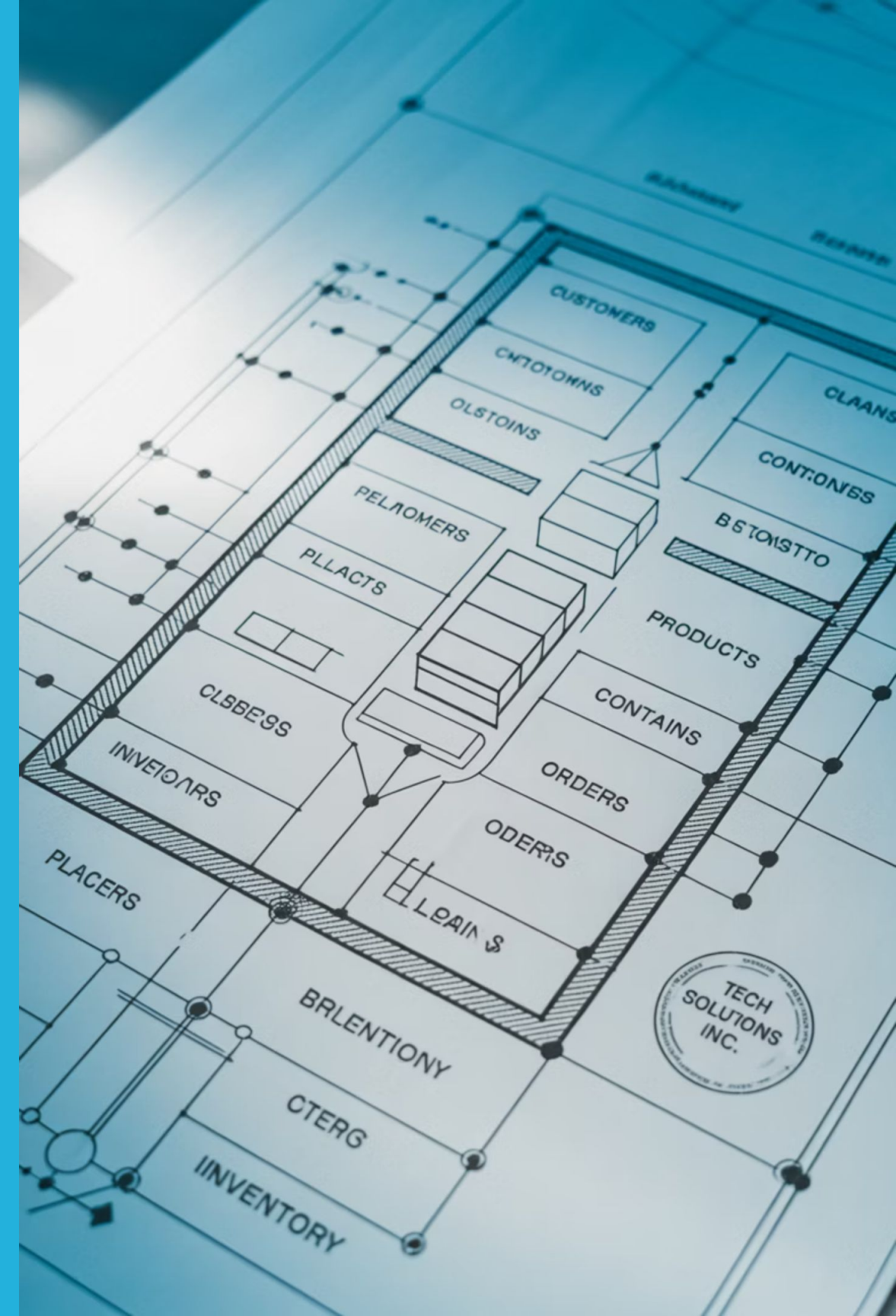
Cliente

Pedido

Producto

Pedido\_Producto

👉 Se usan las tres relaciones (1:1, 1:N, N:M).





# Ejemplo Práctico en VS Code

-- Crear tablas Cliente y Pedido con 1:N

-- Relación Pedido-Producto N:M

```
CREATE TABLE Producto (  
    id INT PRIMARY KEY,  
    nombre VARCHAR(100),  
    precio DECIMAL(10,2)  
);  
  
CREATE TABLE Pedido_Producto (  
    pedido_id INT,  
    producto_id INT,  
    PRIMARY KEY (pedido_id, producto_id),  
    FOREIGN KEY (pedido_id) REFERENCES Pedido(id),  
    FOREIGN KEY (producto_id) REFERENCES Producto(id));
```

# Actividad Colaborativa

**Caso: Diseñar la base de datos de Netflix.**



**Usuario**



**Película**



**Perfil**

**Relación:**

**Usuario ↔ Perfil (1:1)**

**Usuario ↔ Película vista (N:M)**

**👉 Haz el ERD en grupos y compártelo.**



## Discusión en Plenario

### Preguntas para debate:

**¿Qué ventajas tiene separar en varias tablas?**

Piensa en normalización y eficiencia

**¿Qué pasaría si todo estuviera en una sola tabla gigante?**

Considera redundancia y problemas de mantenimiento



# Conexión con SQL Queries

**Contenido:**

**Más adelante veremos cómo unir tablas con JOIN.**

**Las relaciones que diseñamos hoy serán la base para esas consultas.**



**Diseño de Relaciones**



**Consultas JOIN**



**Resultados**



## Cierre Reflexivo

### Ideas clave:

- 1** 1:1, 1:N, N:M representan relaciones del mundo real.
- 2** Diseñar bien evita redundancia y errores.
- 3** El modelado es el primer paso antes de programar.

❓ **Pregunta metacognitiva:**  
👉 ¿Qué relación usarías para un sistema de Spotify con Usuarios y Playlists?



# Gracias y Próximos Pasos



**Practicar más ERDs.**



**Implementar relaciones en MySQL desde VS Code.**



**Próxima sesión: Consultas SQL con JOIN.**