

**TECNOLÓGICO NACIONAL DE MÉXICO**

Instituto tecnológico de Mexicali



**Ingeniería en sistemas computacionales**

Fundamentos de Base de datos

TAREA #1 UNIDAD 2

**JOSE RAMON BOGARIN VALENZUELA**

Morales Ruiz Jose Luis

23490385

Derek Sainz Martínez

23490623

Torres Enríquez Braulio Adán

22490366

Mexicali Baja California a 24 de febrero de 2025

## Plataforma de Comercio Electrónico

### Requerimientos:

- Los clientes pueden realizar pedidos y agregar múltiples productos.
- Se debe contar con un historial de compras y facturación.
- Gestión de stock de productos.

### Pasos a realizar:

1. **Identificar las entidades del sistema:** Usuarios, Pedidos, Productos, DetallePedido.
  2. **Definir atributos clave para cada entidad.**
  3. **Establecer relaciones entre entidades.**
  4. **Elegir claves primarias para identificación única.**
  5. **Refinar el diseño para optimizar la estructura.**
  6. **Crear los diagramas de Venn y los diagramas modelo E-R.**
- 

## Entidades Clave:

### 1. Clientes

- . Atributos principales:
  - . ID\_Cliente (PK)
  - . Nombre
  - . Correo electrónico
  - . Dirección

### 2. Productos

- . Atributos principales:
  - . ID\_Producto (PK)
  - . Nombre
  - . Descripción
  - . Precio

### 3. Pedidos

- . Atributos principales:
  - . ID\_Pedido (PK)
  - . ID\_Cliente (FK)
  - . Fecha
  - . Estado

### 4. Facturación

- . Atributos principales:
  - . ID\_Factura (PK)

- ID\_Pedido (FK)
- Monto
- Método de pago
- Fecha de emisión

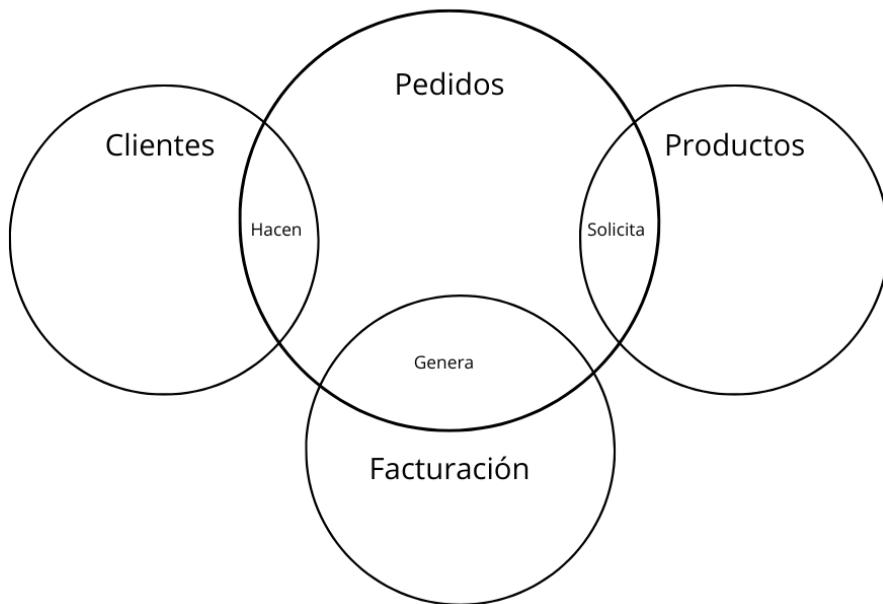
## Relaciones entre Entidades:

- Un cliente puede hacer varios pedidos (**1 a muchos**).
- Un pedido solicita uno o más productos (**muchos a muchos**, requiere una tabla intermedia Pedido\_Producto).
- Un pedido genera una factura (**1 a 1**).

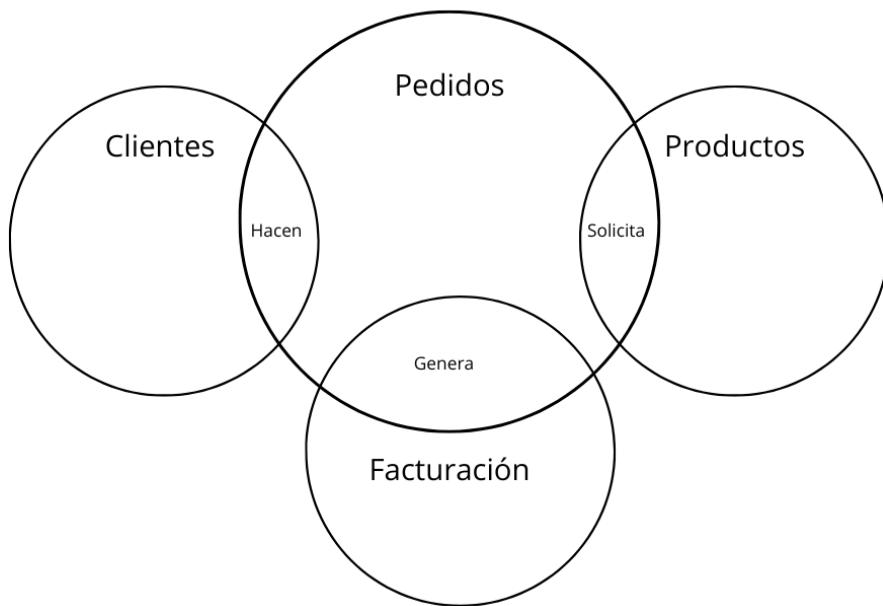
## Claves Primarias y Foráneas:

- PK (Clave Primaria):
  - ID\_Cliente en **Ciudades**
  - ID\_Producto en **Productos**
  - ID\_Pedido en **Pedidos**
  - ID\_Factura en **Facturación**
- FK (Clave Foránea):
  - ID\_Cliente (FK) en **Pedidos** (vinculado a Clientes)
  - ID\_Pedido (FK) en **Facturación** (vinculado a Pedidos)
  - ID\_Pedido (FK), ID\_Producto (FK) en **Pedido\_Producto** (para la relación muchos a muchos entre Pedidos y Productos)

### Comercio electrónico



### Comercio electrónico



## SQL

```
CREATE TABLE Facturacion ( ID_Facturacion INT PRIMARY KEY AUTO_INCREMENT,  
ID_Alumno INT, Monto DOUBLE, Fecha DATE, FOREIGN KEY (ID_Alumno)  
REFERENCES Alumnos(ID_Alumnos) );
```

```
CREATE TABLE Alumnos ( ID_Alumnos INT PRIMARY KEY AUTO_INCREMENT, Nombre  
VARCHAR(100), Email VARCHAR(100), Telefono VARCHAR(15) );
```

```
CREATE TABLE Profesores ( ID_Profesores INT PRIMARY KEY AUTO_INCREMENT,  
Nombre VARCHAR(100), Email VARCHAR(100), Telefono VARCHAR(15) );
```

```
CREATE TABLE Cursos ( ID_Cursos INT PRIMARY KEY AUTO_INCREMENT,  
ID_Profesores INT, Materia VARCHAR(100), Semestre INT, FOREIGN KEY  
(ID_Profesores) REFERENCES Profesores(ID_Profesores) );
```

```
CREATE TABLE Inscripciones ( ID_Inscripciones INT PRIMARY KEY  
AUTO_INCREMENT, ID_Alumnos INT, ID_Cursos INT, ID_Facturacion INT, Costo_Total  
DOUBLE, Fecha DATE, FOREIGN KEY (ID_Alumnos) REFERENCES  
Alumnos(ID_Alumnos), FOREIGN KEY (ID_Cursos) REFERENCES Cursos(ID_Cursos),  
FOREIGN KEY (ID_Facturacion) REFERENCES Facturacion(ID_Facturacion) );
```

```
CREATE TABLE Calificaciones ( ID_Calificaciones INT PRIMARY KEY  
AUTO_INCREMENT, ID_Profesores INT, Calificacion DOUBLE, FOREIGN KEY  
(ID_Profesores) REFERENCES Profesores(ID_Profesores) );
```

## Sistema de Gestión Escolar

### Requerimientos:

- Los alumnos pueden inscribirse en cursos.
- Los profesores pueden impartir múltiples materias.
- Registrar las calificaciones de los alumnos.

### Pasos a realizar:

1. **Identificar las entidades del sistema:** Alumnos, Cursos, Profesores, Inscripciones, Calificaciones.
2. **Definir atributos clave para cada entidad.**
3. **Establecer relaciones entre entidades.**
4. **Elegir claves primarias para identificación única.**
5. **Refinar el diseño para optimizar la estructura.**
6. **Crear los diagramas de Venn y los diagramas modelo E-R.**

#### 1. Alumnos

##### . Atributos principales:

- . ID\_Alumnos (PK)
- . Nombre
- . E-mail
- . Teléfono

#### 2. Profesores

##### . Atributos principales:

- . ID\_Profesores (PK)
- . Nombre
- . E-mail
- . Teléfono

#### 3. Cursos

##### . Atributos principales:

- . ID\_Cursos (PK)
- . ID\_Profesores (FK)
- . ID\_Calificaciones (FK)
- . Materia
- . Semestre

#### 4. Inscripciones

##### . Atributos principales:

- . ID\_Inscripciones (PK)
- . ID\_Alumnos (FK)
- . ID\_Cursos (FK)
- . ID\_Facturacion (FK)
- . Costo\_total
- . Fecha

#### 5. Calificaciones

##### . Atributos principales:

- . ID\_Calificaciones (PK)
- . ID\_Profesores (FK)
- . Calificación

## 6. Relaciones entre Entidades:

7. **Un alumno solicita** una inscripción a uno o más cursos (**1 a muchos**).
8. **Un curso es solicitado** por uno o más alumnos (**muchos a muchos**, a través de Inscripciones).
9. **Un profesor solicita** dar un curso (**1 a muchos**).
10. **Un curso dan** calificaciones (**1 a 1**).
11. **Un profesor verifica** calificaciones (**1 a muchos**).

## 12. Claves Primarias y Foráneas:

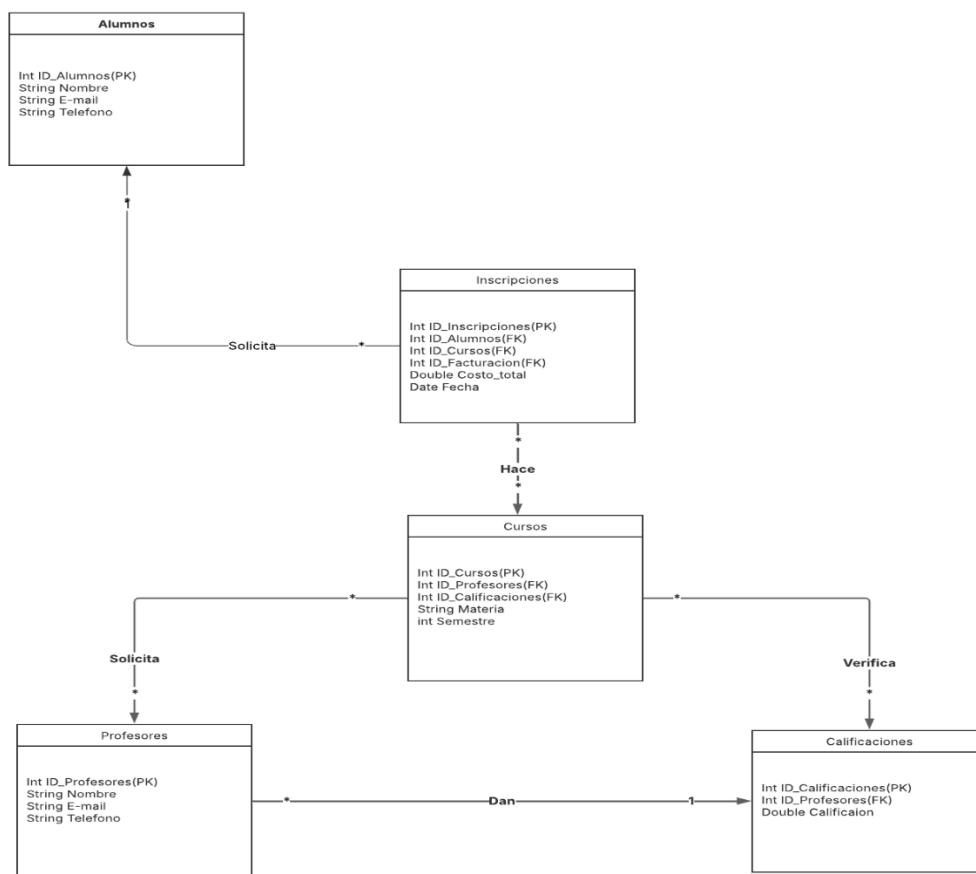
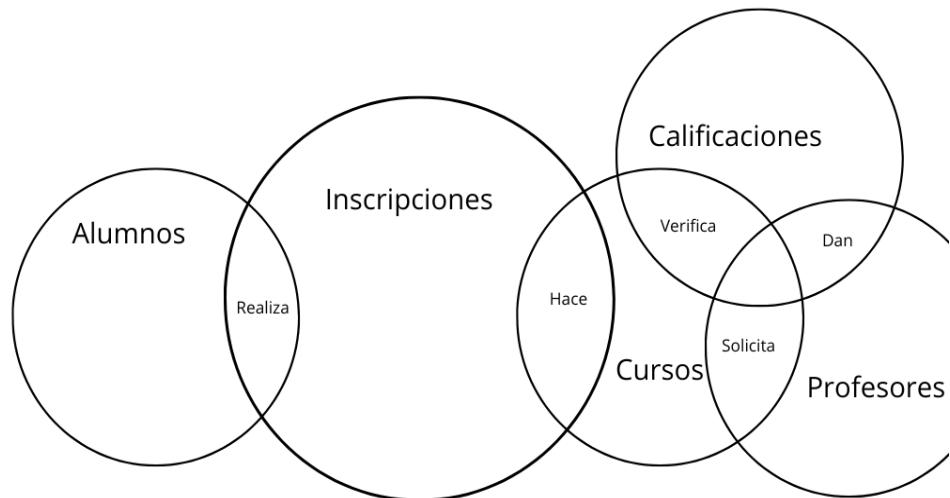
### 13. PK (Clave Primaria):

- . ID\_Alumnos en **Alumnos**
- . ID\_Profesores en **Profesores**
- . ID\_Cursos en **Cursos**
- . ID\_Inscripciones en **Inscripciones**
- . ID\_Calificaciones en **Calificaciones**

### 14. FK (Clave Foránea):

- . ID\_Alumnos (FK) en **Inscripciones** (vinculado a Alumnos)
- . ID\_Cursos (FK) en **Inscripciones** (vinculado a Cursos)
- . ID\_Facturacion (FK) en **Inscripciones**
- . ID\_Profesores (FK) en **Cursos** (vinculado a Profesores)
- . ID\_Calificaciones (FK) en **Cursos** (vinculado a Calificaciones)
- . ID\_Profesores (FK) en **Calificaciones** (vinculado a Profesores)

## Gestión escolar



## SQL

```
CREATE TABLE Alumnos ( ID_Alumnos INT PRIMARY KEY AUTO_INCREMENT, Nombre  
VARCHAR(100), Email VARCHAR(100), Telefono VARCHAR(15) );
```

```
CREATE TABLE Profesores ( ID_Profesores INT PRIMARY KEY AUTO_INCREMENT,  
Nombre VARCHAR(100), Email VARCHAR(100), Telefono VARCHAR(15) );
```

```
CREATE TABLE Cursos ( ID_Cursos INT PRIMARY KEY AUTO_INCREMENT,  
ID_Profesores INT, ID_Calificaciones INT, Materia VARCHAR(100), Semestre INT,  
FOREIGN KEY (ID_Profesores) REFERENCES Profesores(ID_Profesores), FOREIGN  
KEY (ID_Calificaciones) REFERENCES Calificaciones(ID_Calificaciones) );
```

```
CREATE TABLE Inscripciones ( ID_Inscripciones INT PRIMARY KEY  
AUTO_INCREMENT, ID_Alumnos INT, ID_Cursos INT, Fecha DATE, FOREIGN KEY  
(ID_Alumnos) REFERENCES Alumnos(ID_Alumnos), FOREIGN KEY (ID_Cursos)  
REFERENCES Cursos(ID_Cursos) );
```

```
CREATE TABLE Calificaciones ( ID_Calificaciones INT PRIMARY KEY  
AUTO_INCREMENT, ID_Profesores INT, Calificacion DOUBLE, FOREIGN KEY  
(ID_Profesores) REFERENCES Profesores(ID_Profesores) );
```

### 3 Aplicación de Mensajería

#### Requerimientos:

- Los usuarios pueden enviarse mensajes entre sí.
- Los mensajes pueden incluir archivos adjuntos.
- Se debe almacenar el historial de conversaciones.

#### Pasos a realizar:

1. **Identificar las entidades del sistema:** Usuarios, Mensajes, Adjuntos.
2. **Definir atributos clave para cada entidad.**
3. **Establecer relaciones entre entidades.**
4. **Elegir claves primarias para identificación única.**
5. **Refinar el diseño para optimizar la estructura.**
6. **Crear los diagramas de Venn y los diagramas modelo E-R.**

## Entidades Clave:

### 1. Usuarios

- . Atributos principales:
  - . ID\_Usuario (PK)
  - . Nombre
  - . E-mail
  - . Teléfono

### 2. Mensajes

- . Atributos principales:
  - . ID\_Mensaje (PK)
  - . ID\_Usuario (FK) (*Referencia a la entidad Usuarios, quien envía el mensaje*)
  - . ID\_Archivo (FK) (*Referencia a la entidad Archivos, si el mensaje tiene un archivo adjunto*)
  - . Mensaje (*Contenido del mensaje de texto*)
  - . Fecha (*Fecha en que se envió el mensaje*)

### 3. Archivos

- . Atributos principales:
  - . ID\_Archivo (PK)
  - . Tipo (*Ejemplo: imagen, documento, video, etc.*)
  - . Nombre (*Nombre del archivo almacenado*)

## Relaciones entre Entidades:

- Un usuario puede enviar muchos mensajes (1 a muchos entre Usuarios y Mensajes).
- Un mensaje puede contener un archivo adjunto, pero no es obligatorio (1 a 1 opcional entre Mensajes y Archivos).
- Un archivo pertenece a un solo mensaje, pero un usuario puede enviar múltiples archivos en distintos mensajes (1 a 1 entre Mensajes y Archivos).

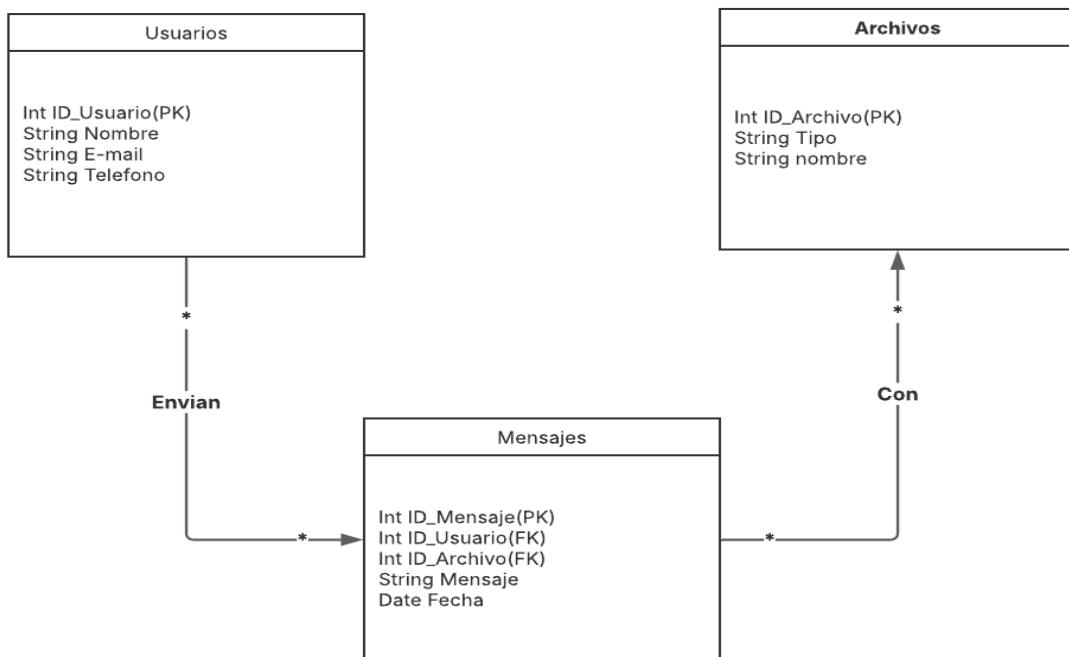
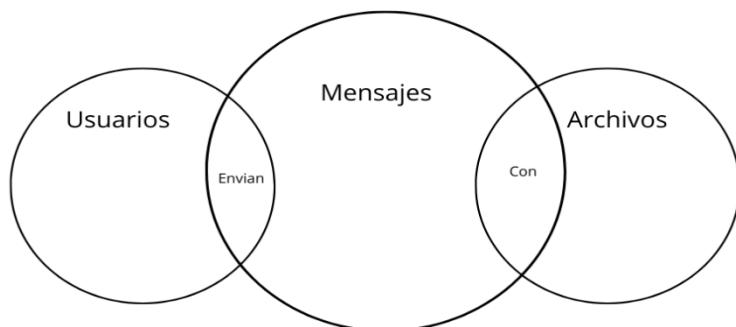
## Claves Primarias y Foráneas:

- PK (Clave Primaria):
  - ID\_Usuario en Usuarios
  - ID\_Mensaje en Mensajes
  - ID\_Archivo en Archivos
- FK (Clave Foránea):
  - ID\_Usuario (FK) en Mensajes (para identificar qué usuario envió el mensaje)
  - ID\_Archivo (FK) en Mensajes (para asociar un archivo con un mensaje, si lo tiene)

## Explicación del Flujo:

1. Un usuario registrado en el sistema puede enviar mensajes.
2. Cada mensaje se almacena en la tabla Mensajes, con un identificador único y la referencia al usuario que lo envió.
3. Un mensaje puede incluir un archivo adjunto, el cual se almacena en la tabla Archivos, y su referencia se guarda en la tabla Mensajes.

## Aplicacion de mensajeria



## SQL

```
CREATE TABLE Usuarios ( ID_Usuario INT PRIMARY KEY AUTO_INCREMENT, Nombre  
VARCHAR(100), Email VARCHAR(100), Telefono VARCHAR(15) );
```

```
CREATE TABLE Archivos ( ID_Archivo INT PRIMARY KEY AUTO_INCREMENT, Tipo  
VARCHAR(50), Nombre VARCHAR(100) );
```

```
CREATE TABLE Mensajes ( ID_Mensaje INT PRIMARY KEY AUTO_INCREMENT,  
ID_Usuario INT, ID_Archivo INT NULL, Mensaje TEXT, Fecha DATETIME, FOREIGN KEY  
(ID_Usuario) REFERENCES Usuarios(ID_Usuario), FOREIGN KEY (ID_Archivo)  
REFERENCES Archivos(ID_Archivo) ON DELETE SET NULL );
```

#### 4- Plataforma de Streaming de Música ♪

##### 📌 Requerimientos:

Los usuarios pueden crear playlists con canciones.

Los artistas pueden subir canciones.

Se debe almacenar el historial de reproducción.

##### ✓ Pasos a realizar:

1. Identificar las entidades del sistema: Usuarios, Canciones, Playlists, HistorialReproducción.
2. Definir atributos clave para cada entidad.
3. Establecer relaciones entre entidades.
4. Elegir claves primarias para identificación única.
5. Refinar el diseño para optimizar la estructura.
6. Crear los diagramas de Venn y los diagramas modelo E-R.

## **Entidades del sistema**

**Usuario:** Representa a la persona que reproduce las canciones.

**Canciones:** Representa lo que reproduce el usuario.

**Playlist:** Representa un almacén de canciones variadas.

**Historial de reproducción:** Almacena las canciones reproducidas.

## **Entidades clave**

- **Usuario:** ID\_Usuario (PK), Nombre, Correo, Teléfono.
- **Canciones:** ID\_Cancion (PK), NombreCancion, Artista, Álbum, Fecha, Genero, Duración.
- **Playlist:** ID\_Playlist (PK), ID\_Usuario (FK), ID\_Cancion (FK), NombrePlaylist, CantCanciones, DuraciónPlaylist.
- **Historial de reproducción:** ID\_Historial (PK), ID\_Usuario (FK), ID\_Cancion (FK).

## **Relación de entidades**

- Un **Usuario** puede escuchar varias **Canciones**
- Un **Usuario** puede crear varias **Playlist**
- Varias **Canciones** puede estar en varias **Playlist**
- El **Hisotorial de reproducción** guarda lo reproducido en **Canciones**

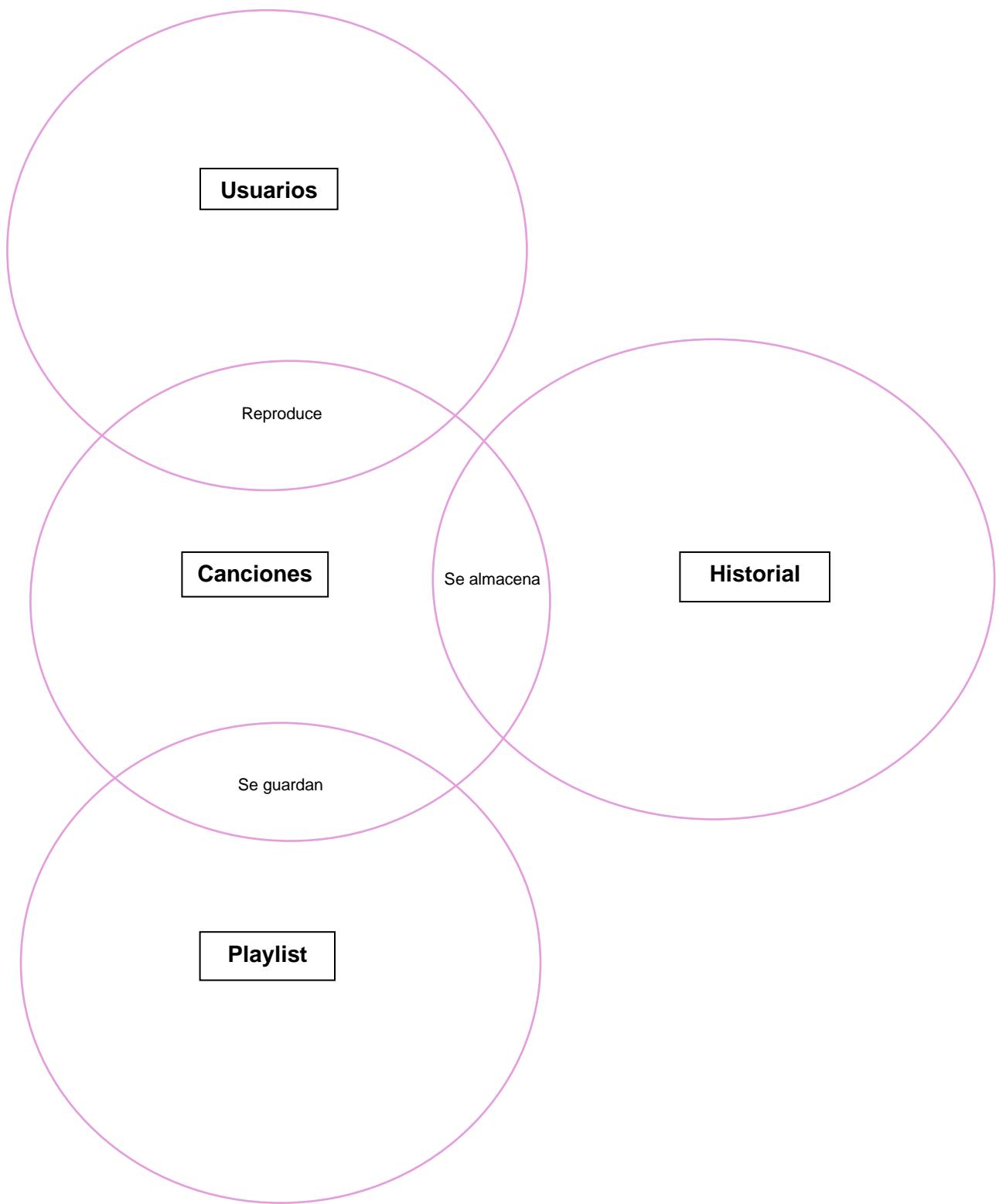
## **Elegir claves primarias para identificación única**

- **ID\_Usuario** como clave primaria para **Usuario**
- **ID\_Cancion** como clave primaria para **Canciones**
- **ID\_Playlist** como clave primaria para **Playlist**
- **ID\_Historial** como clave primaria para **Hisotorial de reproducción**

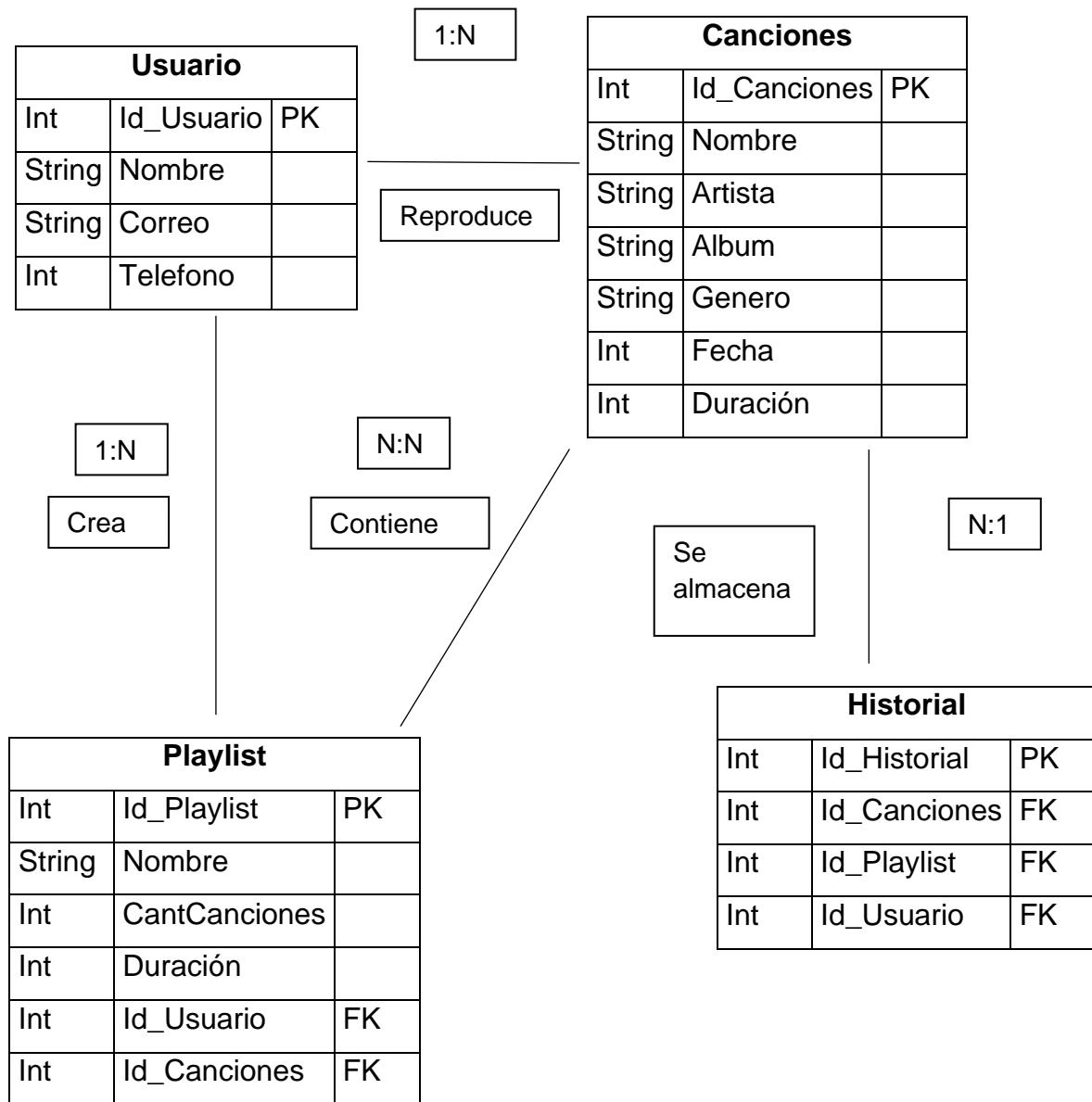
## **Refinar el diseño**

**Normalización:** Los datos se separan en entidades bien definidas para no tener redundancias, evitando que un usuario no tenga más de un historial de reproducción y que no pueda escuchar varias canciones al mismo tiempo.

## Diagrama de VENN



## Diagrama Modelo E-R



## **Creación de base de datos en PostgreSQL**

```
CREATE TABLE Usuario (
    ID_Usuario SERIAL PRIMARY KEY,
    Nombre VARCHAR(255) NOT NULL,
    Correo VARCHAR(255) NOT NULL,
    Telefono VARCHAR(20)
);

CREATE TABLE Canciones (
    ID_Cancion SERIAL PRIMARY KEY,
    NombreCancion VARCHAR(255) NOT NULL,
    Artista VARCHAR(255) NOT NULL,
    Album VARCHAR(255),
    Fecha DATE,
    Genero VARCHAR(50),
    Duracion INTERVAL
);

CREATE TABLE Playlist (
    ID_Playlist SERIAL PRIMARY KEY,
    ID_Usuario INT REFERENCES Usuario(ID_Usuario),
    ID_Cancion INT REFERENCES Canciones(ID_Cancion),
    NombrePlaylist VARCHAR(255) NOT NULL,
    CantCanciones INT,
    DuracionPlaylist INTERVAL
);
```

```
CREATE TABLE Historial_Reproduccion (
    ID_Historial SERIAL PRIMARY KEY,
    ID_Usuario INT REFERENCES Usuario(ID_Usuario),
    ID_Cancion INT REFERENCES Canciones(ID_Cancion)
);
```

## Sistema de Gestión de Citas Médicas

### ❖ Requerimientos:

Los pacientes pueden agendar citas con médicos.

Cada cita debe incluir fecha, hora y estado (pendiente, atendida, cancelada).

Los médicos deben contar con horarios específicos.

### ✓ Pasos a realizar:

Identificar las entidades del sistema: Pacientes, Médicos, Citas, Especialidades.

Definir atributos clave para cada entidad.

Establecer relaciones entre entidades

Elegir claves primarias para identificación única

Refinar el diseño para optimizar la estructura

Crear los diagramas de Venn y los diagramas modelo E-R

## **Entidades del sistema**

**Paciente:** Son las personas que acudirán a las citas medicas.

**Cita:** Son el intermediario entre el paciente y el médico.

**Médico:** Son las persona que atenderá al paciente.

**Especialidad:** Es en lo que el médico se especializa.

## **Entidades clave**

**Paciente:** ID\_Paciente (PK), Nombre, Correo, Teléfono, Edad, NSS.

**Médico:** ID\_Medico (PK), Nombre, Correo, Teléfono, Horario, Especialidad.

**Citas:** ID\_Cita (PK), ID\_Paciente (FK), ID\_Medico (FK), Fecha Cita, Hora Cita.

## **Relación de entidades**

El **Paciente** acude al hospital para ser atendido por un **Médico**

El **Médico** atiende al **Paciente** mediante una **Cita**

La **Cita** organiza el encuentro entre el **Paciente** y el **Médico**.

## **Claves primarias**

**ID\_Paciente** para **Paciente**

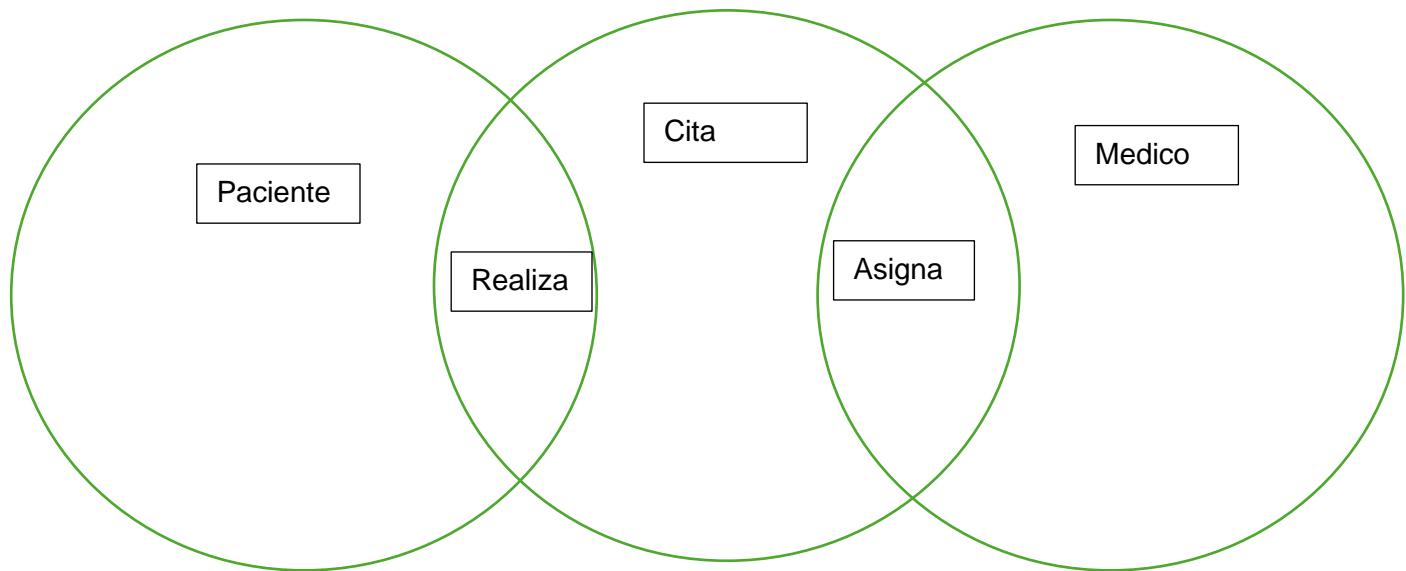
**ID\_Medico** para **Médico**

**ID\_Cita** para **Cita**

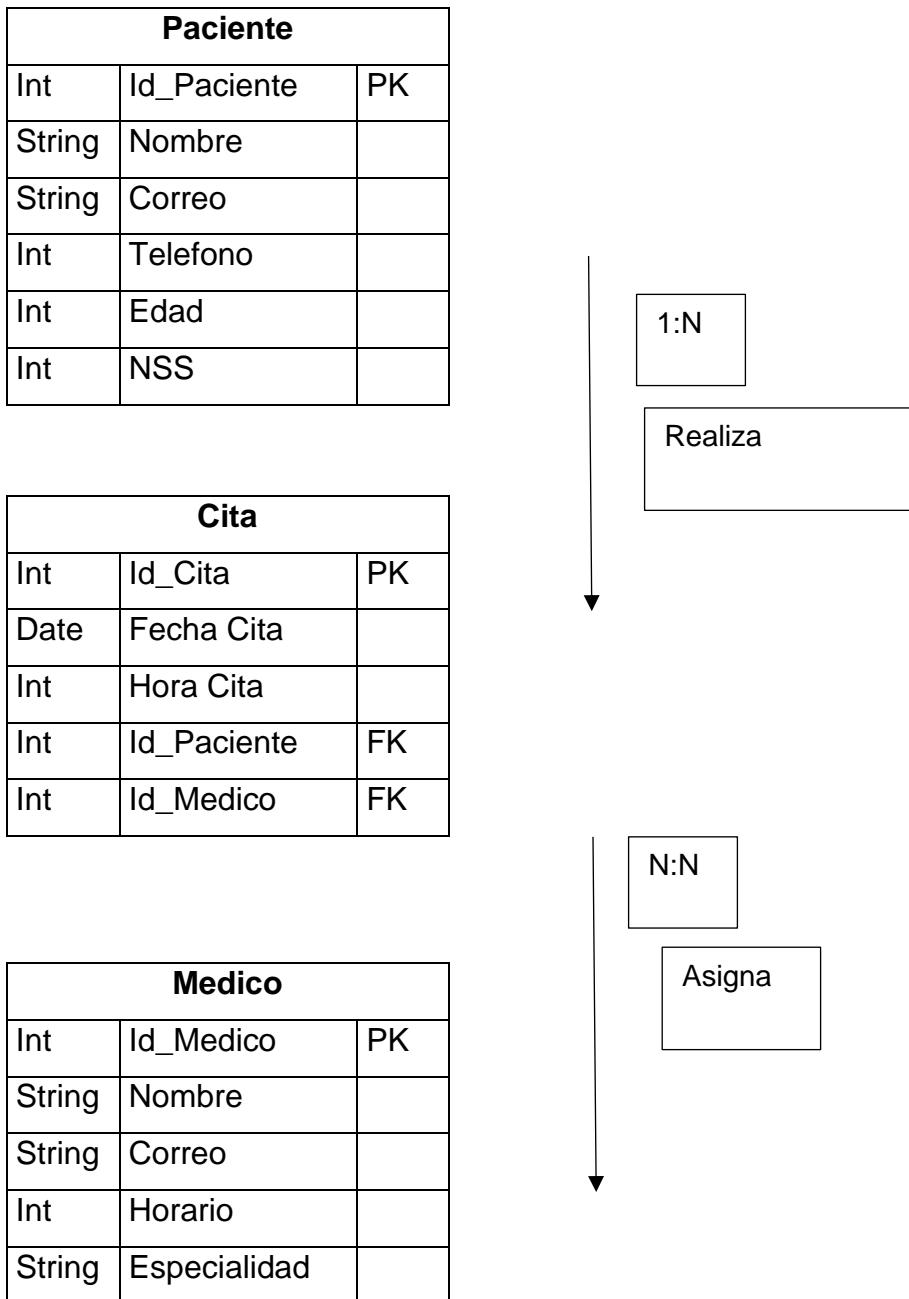
## **Refinar el diseño**

El Paciente no puede ser atendido por varios médicos a la vez y el médico no puede atender a varios pacientes a la vez, tampoco pueden estar varios pacientes y varios médicos en la misma cita.

## Diagrama de Venn



## Diagrama de modelo E-R



## **Creación de base de datos en PostgreSQL**

```
CREATE TABLE Paciente (
    ID_Paciente SERIAL PRIMARY KEY,
    Nombre VARCHAR(255) NOT NULL,
    Correo VARCHAR(255) NOT NULL,
    Telefono VARCHAR(20),
    Edad INT,
    Nss VARCHAR(20)
);
```

```
CREATE TABLE Especialidad (
    ID_Especialidad SERIAL PRIMARY KEY,
    NombreEspecialidad VARCHAR(255) NOT NULL
);
```

```
CREATE TABLE Medico (
    ID_Medico SERIAL PRIMARY KEY,
    Nombre VARCHAR(255) NOT NULL,
    Correo VARCHAR(255) NOT NULL,
    Telefono VARCHAR(20),
    Horario VARCHAR(50),
    ID_Especialidad INT REFERENCES Especialidad(ID_Especialidad)
);
```

```
CREATE TABLE Citas (
    ID_Cita SERIAL PRIMARY KEY,
    ID_Paciente INT REFERENCES Paciente(ID_Paciente),
    ID_Medico INT REFERENCES Medico(ID_Medico),
    Fecha_Cita DATE NOT NULL,
    Hora_Cita TIME NOT NULL
);
```

## **Biblioteca Digital**

Requerimientos:

Los usuarios pueden tomar libros en préstamo.

Se debe registrar la fecha de devolución esperada.

Solo ciertos usuarios pueden administrar los libros.

Pasos a realizar:

Identificar las entidades del sistema: Usuario, libro, Préstamo.

Definir atributos clave para cada entidad

Establecer relaciones entre entidades

Elegir claves primarias para identificación única

Refinar el diseño para optimizar la estructura

Crear los diagramas de Venn y los diagramas modelo E-R.

## **Entidades del sistema**

**Usuario:** Es quien leerá los libros.

**Libro:** Es el producto de interés del usuario.

**Préstamo:** Es el medio para que el usuario pueda llevarse el libro.

## **Entidades clave**

**Usuario:** ID\_Usuario (PK), Nombre, teléfono, correo.

**Libro:** ID\_Libro(PK), Nombre libro, autor, año, género, folio.

**Préstamo:** ID\_Prestamo(PK), ID\_Usuario(PK), ID\_Libro(PK), Fecha préstamo, fecha devolución.

## **Relación entre entidades**

El **Usuario** pide un **libro**

El **Libro** se le otorga al usuario mediante un **Préstamo**

El **Préstamo** contiene los datos del **Usuario** y el **Libro** que se le asignó.

## **Claves primarias**

**ID\_Usuario** para el **Usuario**

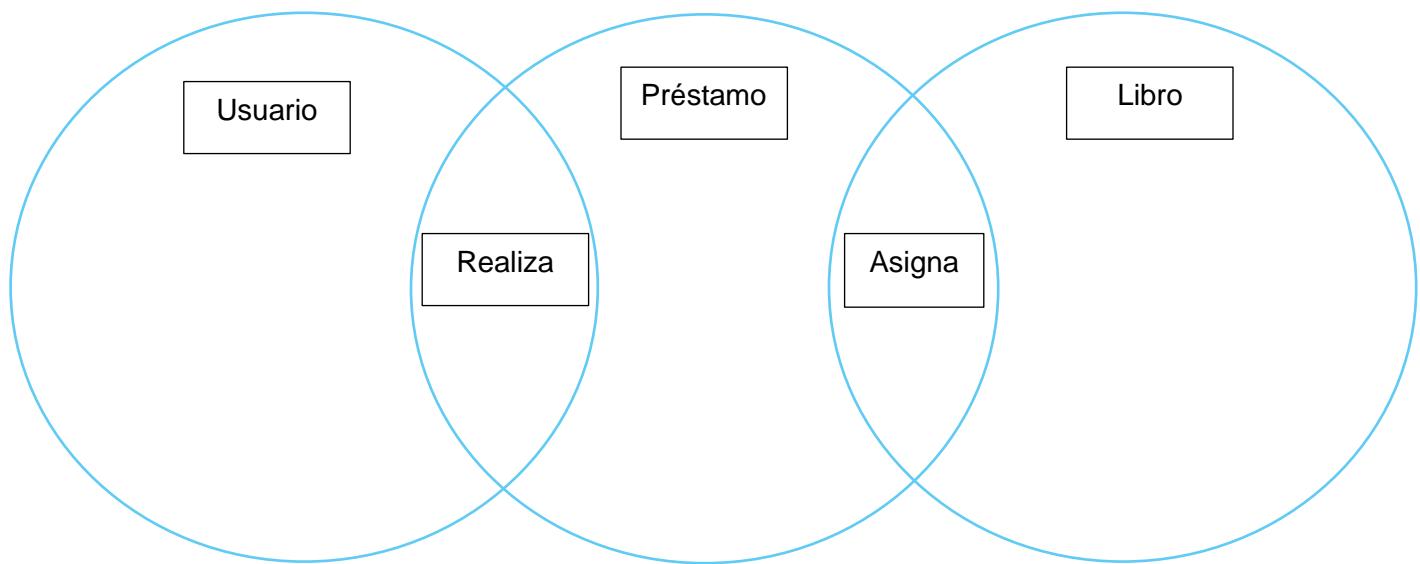
**ID\_Libro** para el **Libro**

**ID\_Prestamo** para el **Préstamo**

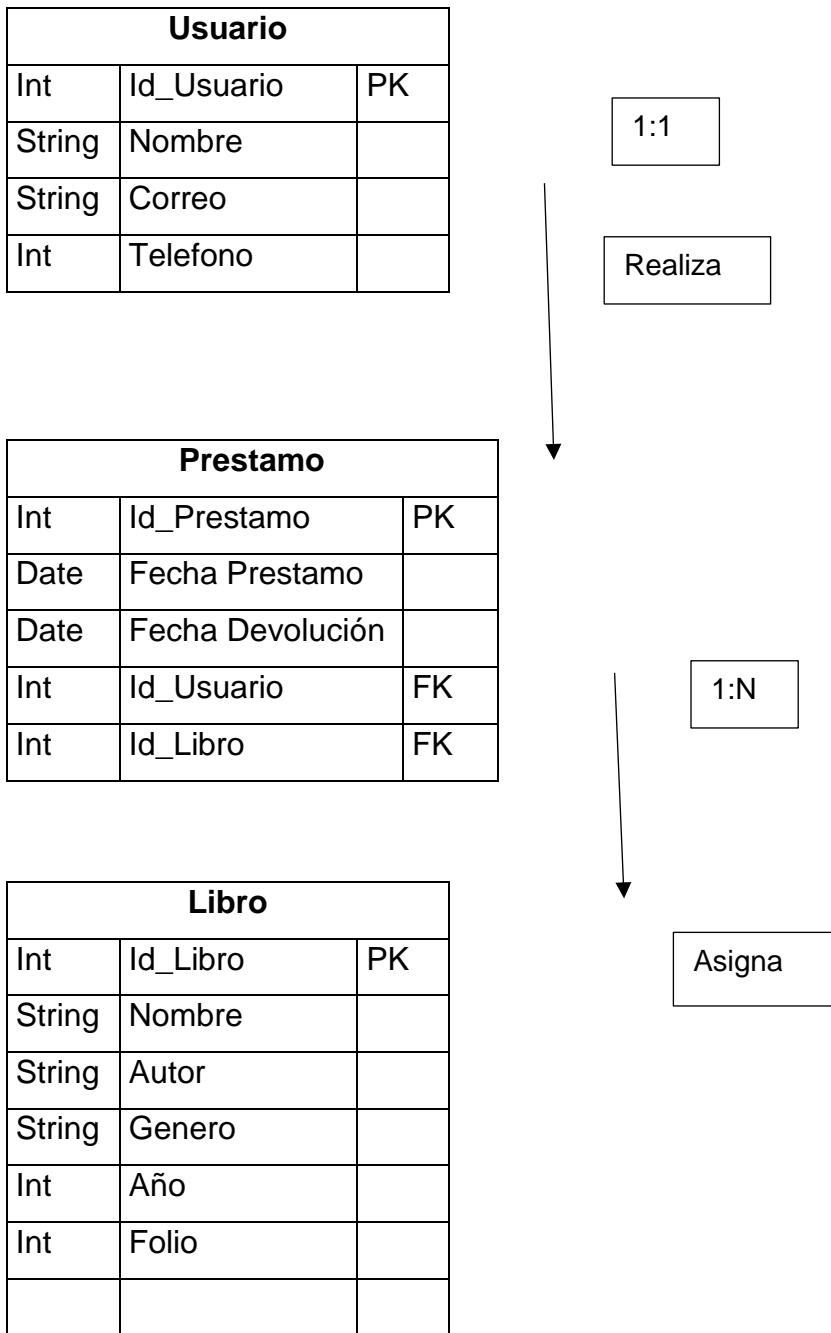
## **Refinar el diseño**

El usuario puede pedir varios libros de un solo préstamo, pero no puede pedir el mismo libro ni varios préstamos a la vez.

## Diagrama de Venn



## Diagrama de modelo E-R



## **Creación de base de datos en PostgreSQL**

```
CREATE TABLE Usuario (
    ID_Usuario SERIAL PRIMARY KEY,
    Nombre VARCHAR(255) NOT NULL,
    Telefono VARCHAR(20),
    Correo VARCHAR(255) NOT NULL
);
```

```
CREATE TABLE Libro (
    ID_Libro SERIAL PRIMARY KEY,
    Nombre_Libro VARCHAR(255) NOT NULL,
    Autor VARCHAR(255) NOT NULL,
    Año INT,
    Genero VARCHAR(50),
    Folio VARCHAR(50)
);
```

```
CREATE TABLE Prestamo (
    ID_Prestamo SERIAL PRIMARY KEY,
    ID_Usuario INT REFERENCES Usuario(ID_Usuario),
    ID_Libro INT REFERENCES Libro(ID_Libro),
    Fecha_Prestamo DATE NOT NULL,
    Fecha_Devolucion DATE
);
```

---

## Sistema de Gestión de Proyectos

### Requerimientos:

- Las empresas pueden registrar proyectos.
- Cada proyecto incluye múltiples tareas y responsables.
- Se deben registrar los avances de cada tarea.

### Pasos a realizar:

1. **Identificar las entidades del sistema:** Empresas, Proyectos, Tareas, Usuarios.
2. **Definir atributos clave para cada entidad.**
3. **Establecer relaciones entre entidades.**
4. **Elegir claves primarias para identificación única.**
5. **Refinar el diseño para optimizar la estructura.**
6. **Crear los diagramas de Venn y los diagramas modelo E-R.**

### **Identificar las entidades del sistema:**

- **Empresa:** Registra proyectos.
- **Proyecto:** Contiene múltiples tareas.
- **Tarea:** Pertenece a un proyecto y tiene responsables.
- **Usuario:** Puede ser responsable de una o varias tareas.

### **Definir atributos clave para cada entidad:**

- **Empresa:** ID\_Empresa, Nombre, Dirección.
- **Proyecto:** ID\_Proyecto, Nombre, Descripción, Fecha\_Inicio, Fecha\_Fin, ID\_Empresa.
- **Tarea:** ID\_Tarea, Descripción, Estado, Fecha\_Límite, ID\_Proyecto.
- **Usuario:** ID\_Usuario, Nombre, Correo, Rol.

### **Establecer relaciones entre entidades:**

- **Empresa** → Puede tener **muchos proyectos**.
- **Proyecto** → Puede tener **muchas tareas**.
- **Tarea** → Puede ser asignada a **uno o varios usuarios**.

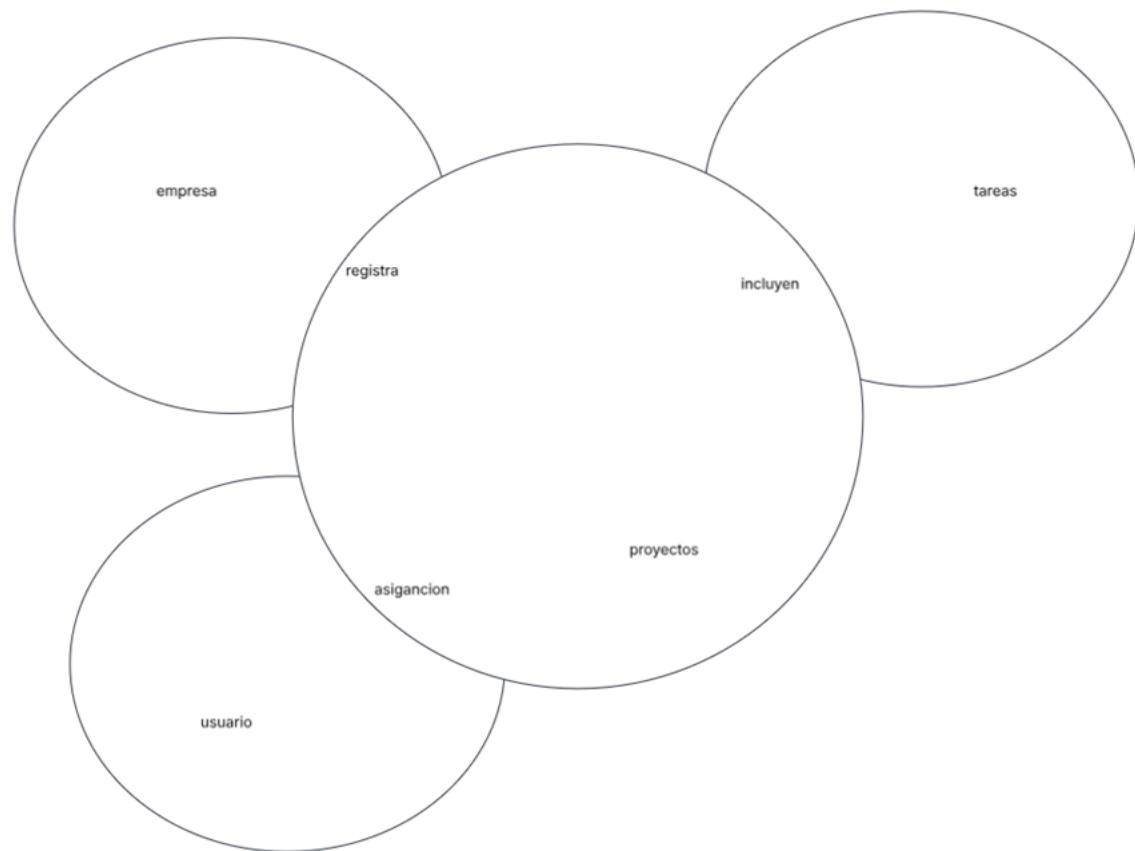
### **Elegir claves primarias para identificación única:**

- **ID\_Empresa** en Empresa.
- **ID\_Proyecto** en Proyecto.
- **ID\_Tarea** en Tarea.
- **ID\_Usuario** en Usuario.

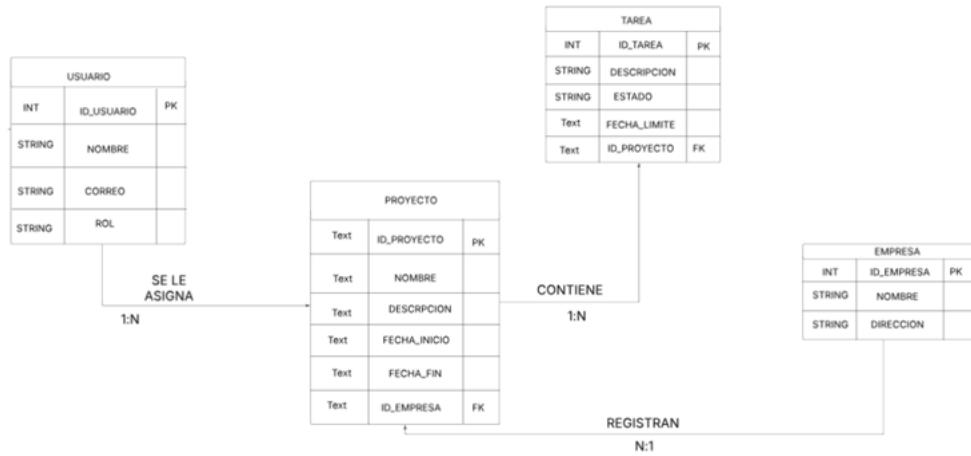
## Refinar el diseño para optimizar la estructura:

- Crear una tabla intermedia **Asignación\_Tareas** (**ID\_Tarea**, **ID\_Usuario**) para manejar relaciones **muchos a muchos** entre Tareas y Usuarios.

Diagrama de ven



## DIAGRAMAS E-R



### CODIGO SQL

```
CREATE DATABASE GestionProyectos;
```

```
USE GestionProyectos;
```

-- Tabla Empresa

```
CREATE TABLE Empresa (
    ID_Empresa INT PRIMARY KEY AUTO_INCREMENT,
    Nombre VARCHAR(255) NOT NULL,
    Direccion TEXT
);
```

-- Tabla Proyecto

```
CREATE TABLE Proyecto (
    ID_Proyecto INT PRIMARY KEY AUTO_INCREMENT,
    Nombre VARCHAR(255) NOT NULL,
    Descripcion TEXT,
    Fecha_Inicio DATE,
    Fecha_Fin DATE,
    ID_Empresa INT,
```

```
    FOREIGN KEY (ID_Empresa) REFERENCES Empresa(ID_Empresa) ON
DELETE CASCADE
);
```

-- Tabla Tarea

```
CREATE TABLE Tarea (
    ID_Tarea INT PRIMARY KEY AUTO_INCREMENT,
    Descripcion TEXT NOT NULL,
    Estado ENUM('Pendiente', 'En Progreso', 'Completada') NOT NULL,
    Fecha_Limite DATE,
    ID_Proyecto INT,
    FOREIGN KEY (ID_Proyecto) REFERENCES Proyecto(ID_Proyecto) ON
DELETE CASCADE
);
```

-- Tabla Usuario

```
CREATE TABLE Usuario (
    ID_Usuario INT PRIMARY KEY AUTO_INCREMENT,
    Nombre VARCHAR(255) NOT NULL,
    Correo VARCHAR(255) UNIQUE NOT NULL,
    Rol ENUM('Administrador', 'Gerente', 'Desarrollador', 'Tester') NOT NULL
);
```

-- Tabla intermedia para la relación muchos a muchos entre Tarea y Usuario

```
CREATE TABLE Usuario_Tarea (
    ID_Usuario INT,
    ID_Tarea INT,
    PRIMARY KEY (ID_Usuario, ID_Tarea),
    FOREIGN KEY (ID_Usuario) REFERENCES Usuario(ID_Usuario) ON
DELETE CASCADE,
```

```
FOREIGN KEY (ID_Tarea) REFERENCES Tarea(ID_Tarea) ON DELETE  
CASCADE  
);
```

 Red Social

 Requerimientos:

- Los usuarios pueden hacer publicaciones y reaccionar a ellas.
- Se debe almacenar el historial de interacciones.
- Los usuarios pueden seguir a otros usuarios.

 Pasos a realizar:

1. **Identificar las entidades del sistema:** Usuarios, Publicaciones, Comentarios, Reacciones.
2. **Definir atributos clave para cada entidad.**
3. **Establecer relaciones entre entidades.**
4. **Elegir claves primarias para identificación única.**
5. **Refinar el diseño para optimizar la estructura.**
6. **Crear los diagramas de Venn y los diagramas modelo E-R.**

## Usuario

- ID\_Usuario (PK, INT, AUTO\_INCREMENT)
- Nombre (VARCHAR)
- Correo (VARCHAR, UNIQUE)

## Publicación

- ID\_Publicacion (PK, INT, AUTO\_INCREMENT)
- ID\_Usuario (FK, INT) → Referencia a **Usuario**
- Fecha\_Creacion (TIMESTAMP)

## Comentario

- ID\_Comentario (PK, INT, AUTO\_INCREMENT)
- ID\_Publicacion (FK, INT) → Referencia a **Publicación**
- ID\_Usuario (FK, INT) → Referencia a **Usuario**
- Fecha\_Creacion (TIMESTAMP)

## Reacción

- ID\_Reaccion (PK, INT, AUTO\_INCREMENT)
- ID\_Usuario (FK, INT) → Referencia a **Usuario**
- ID\_Publicacion (FK, INT) → Referencia a **Publicación** (opcionalmente, también a comentarios)
- Fecha\_Reaccion (TIMESTAMP)

### **Establecer relaciones entre entidades:**

- **Un usuario** puede hacer **muchas publicaciones**.
- **Un usuario** puede hacer **muchos comentarios** en distintas publicaciones.
- **Un usuario** puede dar **reacciones** a publicaciones y comentarios.
- **Un usuario** puede **seguir a otros usuarios**.

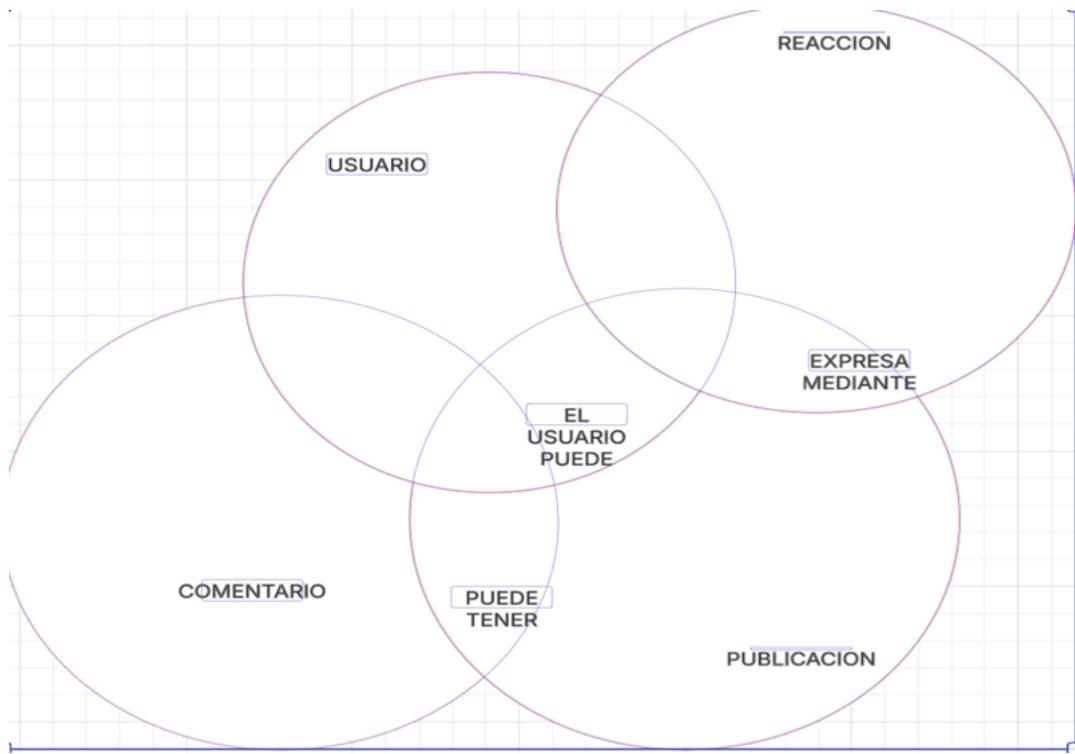
### **Elegir claves primarias para identificación única:**

Cada tabla tiene su ID como clave primaria (PK).

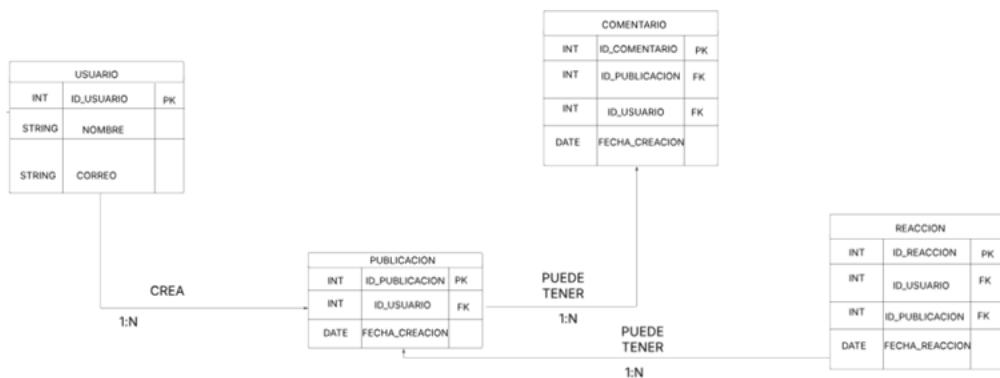
### **Refinar el diseño para optimizar la estructura:**

- Agregar índices en ID\_Usuario, ID\_Publicacion y ID\_Comentario para mejorar consultas.
- Normalizar la base de datos eliminando datos redundantes.
- Agregar **restricciones de integridad** (ON DELETE CASCADE) para mantener consistencia.

## DIAGRAMA VENN



## RELACION E-R



Código sql

```
CREATE DATABASE RedSocial;
```

```
USE RedSocial;
```

-- Tabla Usuario

```
CREATE TABLE Usuario (
    ID_Usuario INT PRIMARY KEY AUTO_INCREMENT,
    Nombre VARCHAR(255) NOT NULL,
    Correo VARCHAR(255) UNIQUE NOT NULL
);
```

-- Tabla Publicacion

```
CREATE TABLE Publicacion (
    ID_Publicacion INT PRIMARY KEY AUTO_INCREMENT,
    ID_Usuario INT,
    Fecha_Creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (ID_Usuario) REFERENCES Usuario(ID_Usuario) ON DELETE CASCADE
);
```

-- Tabla Comentario

```
CREATE TABLE Comentario (
    ID_Comentario INT PRIMARY KEY AUTO_INCREMENT,
    ID_Publicacion INT,
    ID_Usuario INT,
    Fecha_Creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (ID_Publicacion) REFERENCES Publicacion(ID_Publicacion) ON DELETE CASCADE,
    FOREIGN KEY (ID_Usuario) REFERENCES Usuario(ID_Usuario) ON DELETE CASCADE
);
```

```
-- Tabla Reaccion
CREATE TABLE Reaccion (
    ID_Reaccion INT PRIMARY KEY AUTO_INCREMENT,
    ID_Usuario INT,
    ID_Publicacion INT NULL,
    ID_Comentario INT NULL,
    Fecha_Reaccion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (ID_Usuario) REFERENCES Usuario(ID_Usuario) ON DELETE CASCADE,
    FOREIGN KEY (ID_Publicacion) REFERENCES Publicacion(ID_Publicacion)
    ON DELETE CASCADE,
    FOREIGN KEY (ID_Comentario) REFERENCES Comentario(ID_Comentario)
    ON DELETE CASCADE
);
```

```
-- Tabla Segidores (para la relación de seguir usuarios)
CREATE TABLE Segidores (
    ID_Seguido INT,
    ID_Seguidor INT,
    PRIMARY KEY (ID_Seguido, ID_Seguidor),
    FOREIGN KEY (ID_Seguido) REFERENCES Usuario(ID_Usuario) ON
    DELETE CASCADE,
    FOREIGN KEY (ID_Seguidor) REFERENCES Usuario(ID_Usuario) ON
    DELETE CASCADE
);
```

## Sistema de Facturación

### Requerimientos:

- Generar facturas para clientes por productos comprados.
- Registrar impuestos y descuentos.
- Asociar cada factura a un cliente.

### Pasos a realizar:

1. **Identificar las entidades del sistema:** Clientes, Facturas, DetalleFactura, Productos.
2. Definir atributos clave para cada entidad.
3. Establecer relaciones entre entidades.
4. Elegir claves primarias para identificación única.
5. Refinar el diseño para optimizar la estructura.
6. Crear los diagramas de Venn y los diagramas modelo E-R.

### **Identificar las entidades del sistema:**

- **Cliente:** Persona o empresa que compra productos.
- **Factura:** Documento que registra una compra.
- **DetalleFactura:** Relación entre facturas y productos.
- **Producto:** Artículo disponible para la venta.

### **Definir atributos clave para cada entidad:**

#### **Cliente**

- ID\_Cliente (PK, INT, AUTO\_INCREMENT)
- Nombre (VARCHAR)
- Correo (VARCHAR, UNIQUE)
- Teléfono (VARCHAR)
- Dirección (TEXT)

#### **Factura**

- ID\_Factura (PK, INT, AUTO\_INCREMENT)
- ID\_Cliente (FK, INT) → Referencia a **Cliente**
- Fecha\_Emisión (TIMESTAMP)
- Total (DECIMAL)
- Impuestos (DECIMAL)
- Descuento (DECIMAL)

#### **Producto**

- ID\_Producto (PK, INT, AUTO\_INCREMENT)
- Nombre (VARCHAR)

- Descripción (TEXT)
- Precio (DECIMAL)
- Stock (INT)

#### **DetalleFactura (Tabla intermedia para Facturas y Productos)**

- ID\_Detalle (PK, INT, AUTO\_INCREMENT)
- ID\_Factura (FK, INT) → Referencia a **Factura**
- ID\_Producto (FK, INT) → Referencia a **Producto**
- Cantidad (INT)
- Precio\_Unitario (DECIMAL)
- Subtotal (DECIMAL)

#### **Establecer relaciones entre entidades:**

- **Un cliente** puede tener **muchas facturas**.
- **Una factura** puede contener **muchos productos** (relación muchos a muchos).
- **Un producto** puede estar en **muchas facturas** (por eso se usa la tabla intermedia DetalleFactura).

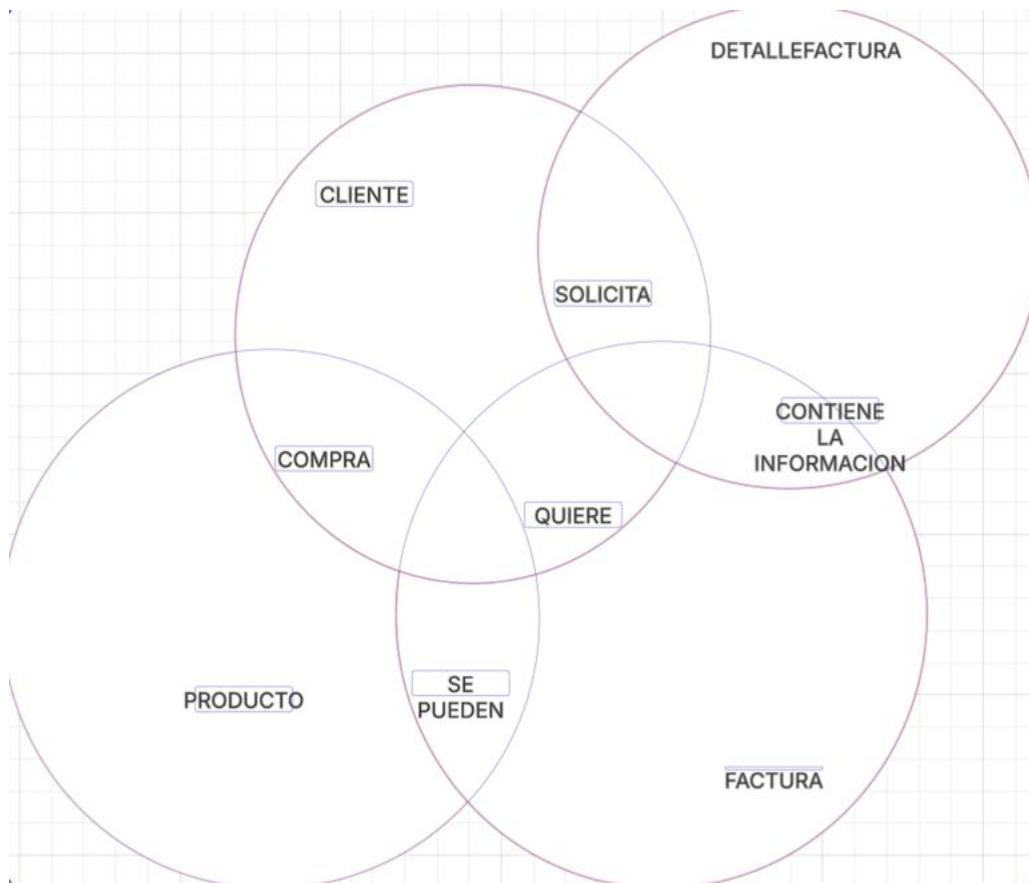
#### **Elegir claves primarias para identificación única:**

Cada tabla tiene su ID como clave primaria (PK).

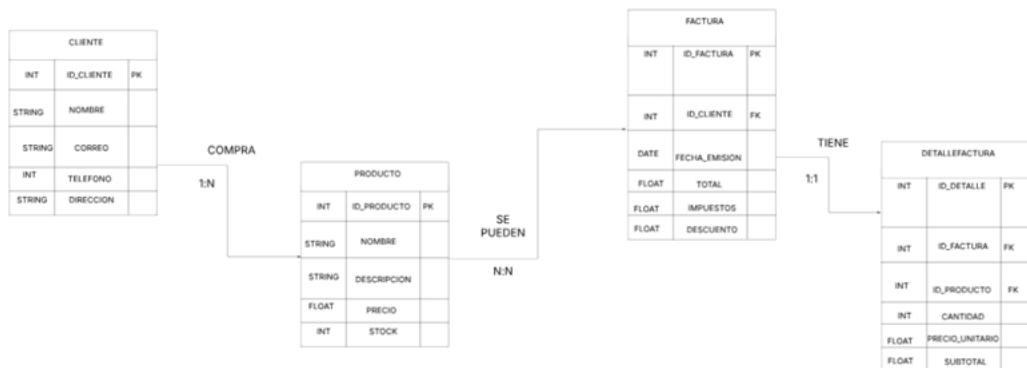
#### **Refinar el diseño para optimizar la estructura:**

- Agregar índices en ID\_Cliente, ID\_Factura y ID\_Producto para mejorar rendimiento.
- Aplicar **restricciones de integridad** (ON DELETE CASCADE) en las claves foráneas.

## DIAGRAMA DE VENN



## RELACION E-R



Código sql

```
CREATE DATABASE GestionFacturas;
```

```
USE GestionFacturas;
```

-- Tabla Cliente

```
CREATE TABLE Cliente (
```

```
    ID_Cliente INT PRIMARY KEY AUTO_INCREMENT,
```

```
    Nombre VARCHAR(255) NOT NULL,
```

```
    Correo VARCHAR(255) UNIQUE NOT NULL,
```

```
    Telefono VARCHAR(20),
```

```
    Direccion TEXT
```

```
);
```

-- Tabla Factura

```
CREATE TABLE Factura (
```

```
    ID_Factura INT PRIMARY KEY AUTO_INCREMENT,
```

```
    ID_Cliente INT,
```

```
    Fecha_Emision TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
    Total DECIMAL(10,2),
```

```
    Impuestos DECIMAL(10,2),
```

```
    Descuento DECIMAL(10,2),
```

```
    FOREIGN KEY (ID_Cliente) REFERENCES Cliente(ID_Cliente) ON DELETE
```

```
CASCADE
```

```
);
```

-- Tabla Producto

```
CREATE TABLE Producto (
```

```
    ID_Producto INT PRIMARY KEY AUTO_INCREMENT,
```

```
    Nombre VARCHAR(255) NOT NULL,
```

```
    Descripcion TEXT,
```

```
    Precio DECIMAL(10,2) NOT NULL,
```

```
    Stock INT NOT NULL  
);
```

-- Tabla DetalleFactura (relación muchos a muchos entre Factura y Producto)

```
CREATE TABLE DetalleFactura (  
    ID_Detalle INT PRIMARY KEY AUTO_INCREMENT,  
    ID_Factura INT,  
    ID_Producto INT,  
    Cantidad INT NOT NULL,  
    Precio_Unitario DECIMAL(10,2) NOT NULL,  
    Subtotal DECIMAL(10,2) NOT NULL,  
    FOREIGN KEY (ID_Factura) REFERENCES Factura(ID_Factura) ON DELETE  
    CASCADE,  
    FOREIGN KEY (ID_Producto) REFERENCES Producto(ID_Producto) ON  
    DELETE CASCADE  
);
```