INSTITUTO TECNOLOGICO DE MEXICALI

Carrera:
ING. en Sistemas.

Materia:
Fundamentos de base de datos.

Alumno:
Marin Salazar Juan Sebastian 22490423
Hernandez Garcia Martin 22490354
Rangel Garcia Jaime Javier 21490880

Profesor:
Jose Ramón Bogarin Valenzuela.

Mexicali, Baja California a 24 de Febrero del 2025.

2.- Plataforma de Comercio Electrónico

Requerimientos:

Los clientes pueden hacer pedidos y agregar múltiples productos.
Se necesita historial de compras y facturación.
Gestión de stock de productos.

Prácticas a aplicar:

Crear entidades: Usuarios, Pedidos, Productos, DetallePedido.
Restricción: No permitir pedidos con stock insuficiente.
Normalizar para evitar repetir datos del cliente en cada pedido.

Identificar las entidades:

Usuarios: Representa a la persona que realiza el/los pedidos
Pedidos: es el intermediario entre los usuarios y el producto
Productos: son los elementos que compondrán los pedidos realizados
DetallePedido: tendrá los detalles correspondientes de cada pedido realizado.

Definitr atributos claves para cada entidad:

Usuarios: IdUsuario (PK), Nombre
Pedidos: disponibilidad
Productos: IdProducto (PK), cantidad
DetallePedido: HistorialCompra, facturación

Establecer relaciones entre entidades:

Los Usuarios hacen multiples pedidos
cada Pedido tiene uno o multiples Productos
los Pedido realizados genera un Detalle de pedido

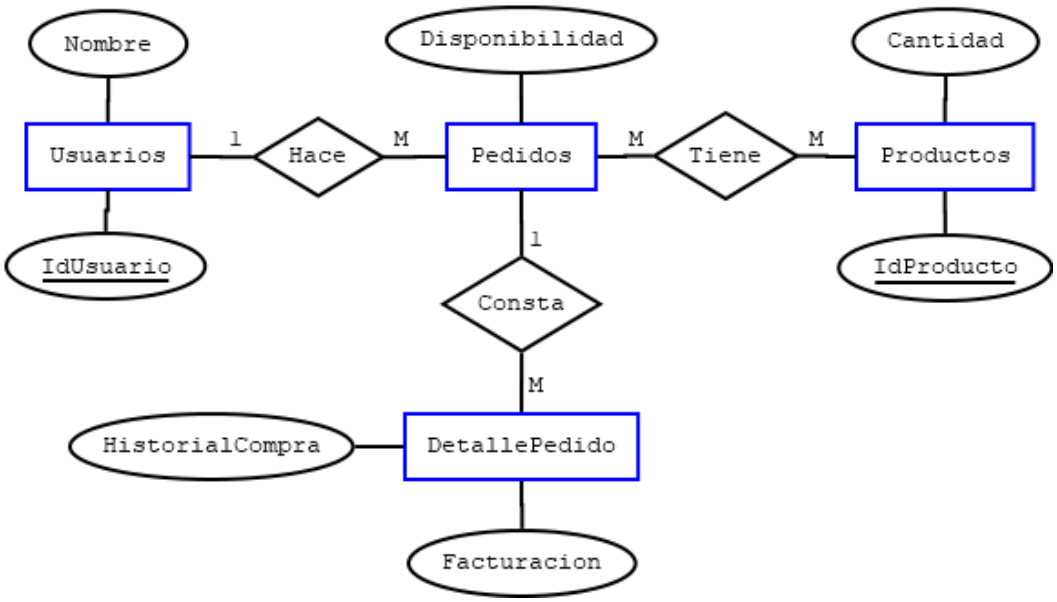
Claves primarias para identificación única

IdUsuarios como clave primaria de Usuarios
IdProducto como clave primaria de Productos

Diagrama de Venn:

Diagrama Entidad - Relación:

2.- Plataforma de comercio Electronico



3.- Sistema de gestión escolar

Requerimientos:

Alumnos se inscriben en cursos.
Los profesores imparten múltiples materias.
Registrar calificaciones de los alumnos.

Prácticas a aplicar:

Identificar entidades: Alumnos, Cursos, Profesores, Inscripciones, Calificaciones.
Reglas de negocio: Un alumno no puede inscribirse en el mismo curso dos veces.
Implementar seguridad para restringir acceso a notas.

Identificar las entidades:

Alumnos: son las personas identificadas
Cursos: son los resultados que se obtienen de la inscripción por parte de los Alumnos
Profesores: es la persona que imparte las clases y obtiene diversos cursos
Inscripciones: es el medio que utilizarán los alumnos para obtener cursos
Calificaciones: es lo que recibe cada estudiante al finalizar un curso

Definir atributos clave para cada entidad:

Alumnos: IdAlumno(PK), Nombre
Cursos: IdCurso(PK), Materias
Profesores: IdProfesor(PK), Nombre
Inscripciones: N/A
Calificaciones: N/A

Establecer relaciones entre entidades:

un Alumnos se inscriben a multiples cursos
en los cursos de pueden inscribir muchos alumnos
los cursos se imparten por un solo Profesor
un solo Profesor puede impartir diversos cursos
los cursos ofrecen calificaciones a los alumnos

Claves primarias para identificación única:

IdAlumno como clave primaria de Alumnos
IdProfesor como clave primaria de Profesores
IdCurso como clave primaria de Cursos

Diagrama de Venn:

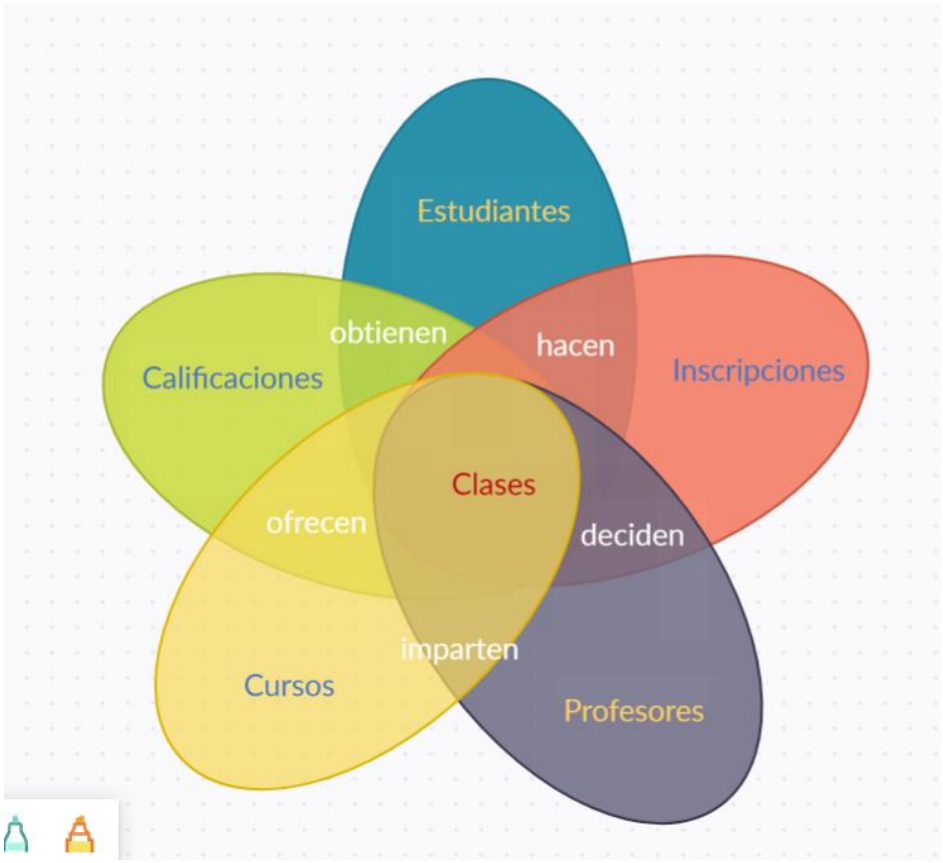
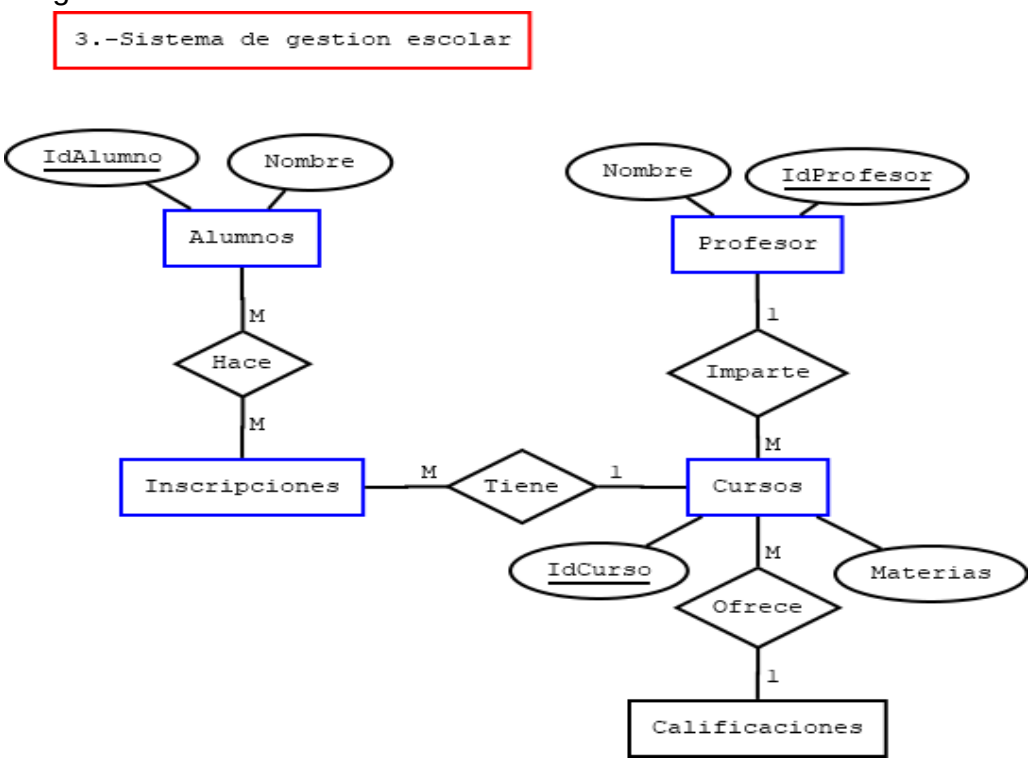


Diagrama entidad - relación:



4.- Aplicacion de mensajeria

Requerimientos:

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

Prácticas a aplicar:

Definir entidades: Usuarios, Mensajes, Adjuntos.
Restricción: Un mensaje solo puede enviarse a usuarios registrados.
Índices en la base de datos para búsquedas rápidas en el historial.

Identificar las entidades:

Usuarios: es la persona que usa el servicio de mensajería
Mensajes: son las interacciones que pueden hacer los usuarios
Adjuntos: es todo aquello que puede contener un mensaje

Definir atributos clave para cada entidad:

Usuarios: Numero(PK), Nombre
Mensajes: Historial
Adjuntos: Documentos, Videos, Audios, Imágenes, Contactos

Establecer relaciones entre entidades:

los Usuarios envían múltiples Mensajes
los Mensajes pueden contener muchos archivos adjuntos

Claves primarias para identificación única:

Numero como clave primaria de Usuarios

Diagrama de Venn:

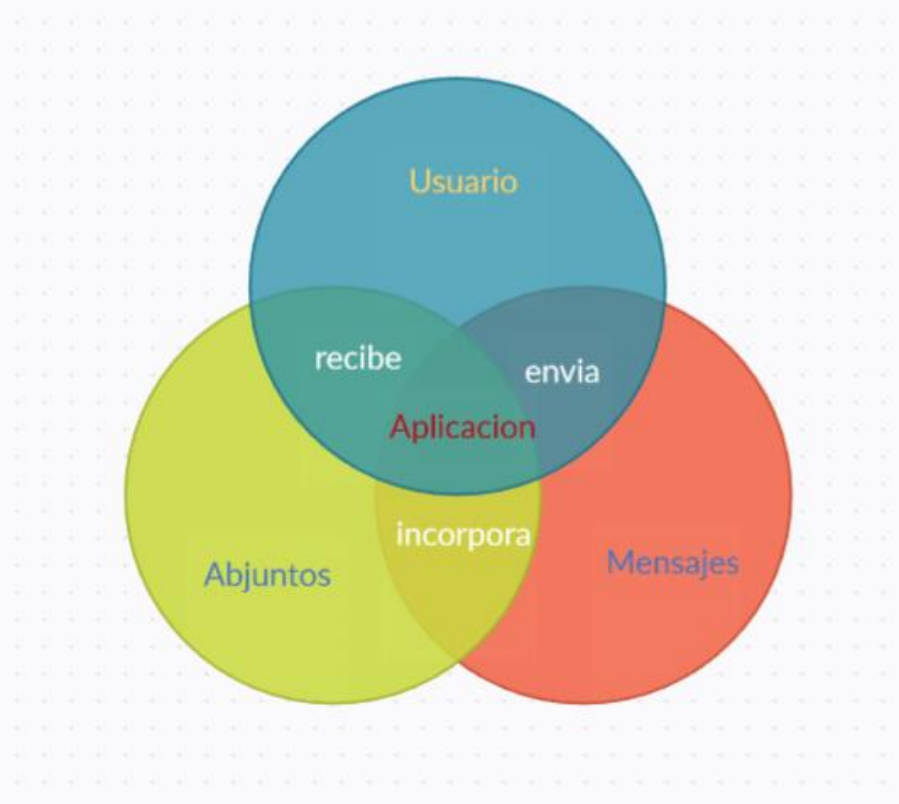
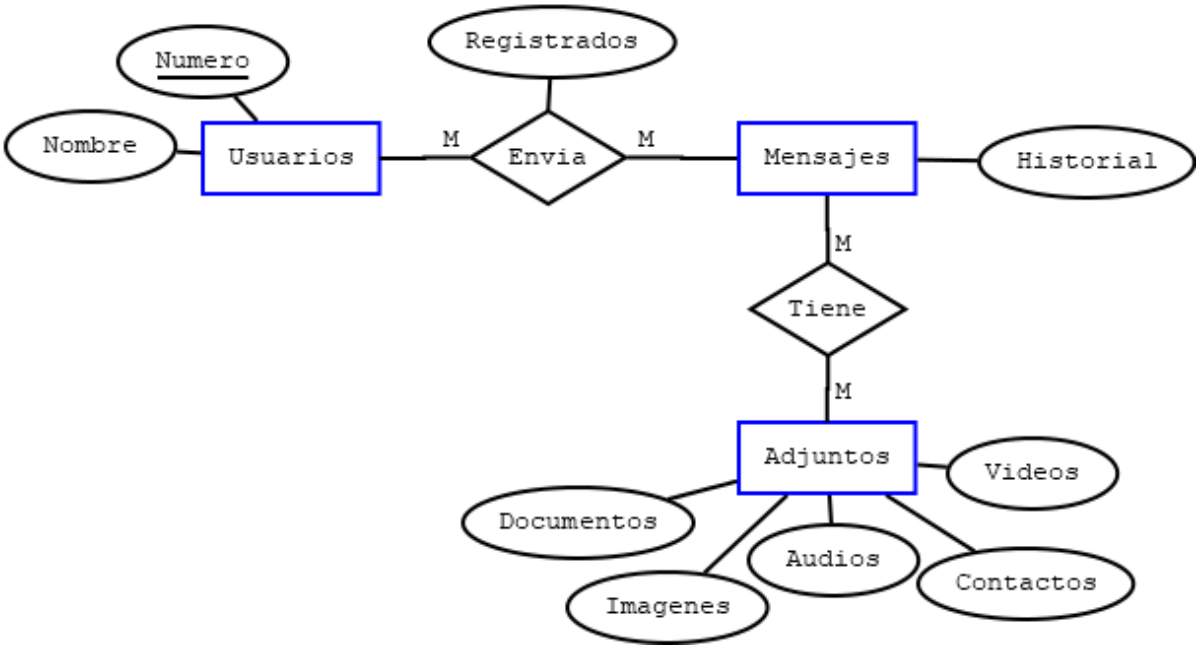


Diagrama Entidad - relación:

4.- Aplicacion de mensajeria



5.- 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.

1.- Identificar las entidades del sistema.

¿Quiénes interactúan con la plataforma?
R = Usuarios.

¿Qué elementos principales maneja la plataforma?
R = Canciones y Playlist.

¿Se necesita registrar interacciones o actividades?
R = Si, un historial de reproducciones.

Entidades identificadas:

Usuario: Representa a las personas que interactúan con la plataforma.
Canción: Es lo que suben los usuarios “Artistas”.
Playlist: Es lo que crean los usuarios.
Historial de reproducciones: Se necesita registrar qué canciones han sido escuchadas.
Álbumes: Este es opcional pero quiero meterlo porque si no no voy a quedar agusto, los usuarios artistas pueden agrupar su música en álbumes.
Género: Este también es opcional, las canciones se les asigna uno o varios géneros musicales.

2.- Definir atributos clave para cada entidad.

Usuario: Id_Usuario (PK), Nombre_Usuario, Correo, Contraseña, Tipo_Usuario, Nombre_Artístico(Opcional) y Biografía (Opcional).

Canción: Id_Cancion (PK), Título, Duración , Id_Usuario (FK) y Id_Album (FK).

Playlist: Id_Playlist (PK), Nombre y Id_Usuario (FK).

Nota** Para la relación muchos a muchos entre Playlists y Canciones, necesitamos una tabla intermedia que le llamaremos: Playlist_Canción.

Playlist_Canción: Id_Playlist (FK) y Id_Cancion (FK).

Historial de reproducciones: Id_Historial (PK), Id_Usuario (FK), Id_Cancion (FK) y Fecha_Hora.

Álbum: Id_Album (PK), Nombre, Id_Usuario (FK) y Fecha_Lanzamiento.

Géneros: Id_Genero (PK) y Nombre.

Nota** Como una canción puede tener más de un género sería una relación de muchos a muchos, así que creamos una tabla intermedia entre Canción y Género, la llamaremos: Canción_Género.

Canción_Género: Id_Cancion (FK) y Id_Genero (PK).

3.- Establecer relaciones entre entidades.

• Usuarios escuchan muchas Canciones.
• Usuarios hacen muchas Playlist.
• Muchas Canciones están en muchas Playlist.
• Las canciones tienen uno o varios Géneros.
• Varias canciones están dentro de un Álbum.
• Usuarios tienen un Historial.
• Historial guarda Canciones.

Entre Canción y Playlist hay una tabla intermedia Playlist_Canción
Y entre Canción y Género hay otra llamada: Cancion_Genero.

Quedaria asi:

Usuario 1:N Canción.
Usuario 1:N Playlist.
Canción 1:N Playlist_Cancion M:1 Playlist.
Canción 1:N Cancion_Genero M:1 Género.
Canción 1:N Album.
Usuario 1:N Historial.
Canción 1:N Historial.

4.- Elegir clave primaria para identificación única.

Usuario:
Clave Primaria (PK): Id_Usuario

Canción:
Clave Primaria (PK): Id_Canción
Clave Foránea (FK): Id_Usuario
Clave Foránea (FK, Opcional): Id_Álbum

Playlist:
Clave Primaria (PK): Id_Playlist
Clave Foránea (FK): Id_Usuario

Playlist_Canción (Intermedia):
Clave Foránea (FK): Id_Playlist y Id_Canción.

Género:
Clave Primaria (PK): Id_Género.

Canción_Género (Intermedia):
Clave Foránea (FK): Id_Canción y Id_Género.

Álbum:
Clave Primaria (PK): Id_Álbum.
Clave Foránea (FK): Id_Usuario.

Historial de Reproducción:
Clave Primaria (PK): Id_Historial.
Clave Foránea (FK): Id_Usuario y Id_Canción.

5.- Refinar el diseño para optimizar la estructura.

Normalización:
Ya cumple la normalización hasta la 3FN.
Para cumplir la 4FN se podría cuestionar ¿y si una canción tiene varios artistas?

Solución para cumplir 4FN:
• Crear una tabla Canción_Artista para manejar canciones con múltiples artistas.
• Evitamos que una canción tenga solo un campo Id_Usuario, lo cual obligaría a duplicar registros.

Canción_Artista:
Clave Foránea (FK): Id_Canción y Id_Usuario.

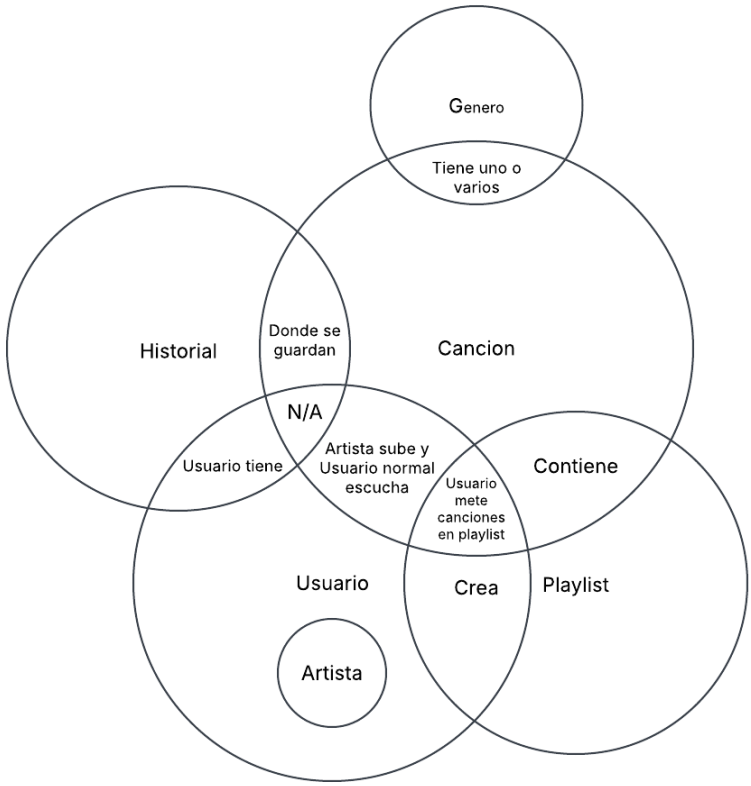
Esto permite que una canción tenga múltiples artistas sin redundancia.
Ahora sí, el modelo cumple hasta la 4FN.

Aplicación de Reglas de Negocio

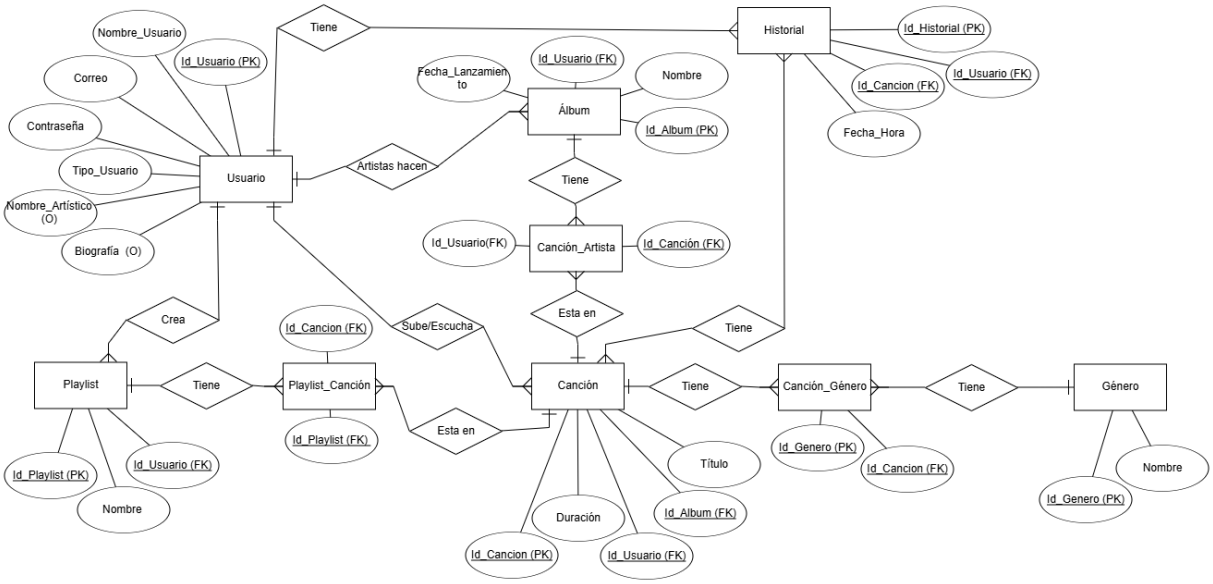
Una playlist solo puede pertenecer a un usuario.

Ya se cumple porque en la tabla Playlist, la clave foránea Id_Usuario evita que una playlist tenga más de un dueño.

6.- Diagrama de Venn.



7.- Diagrama de Modelo ER.



8.- Creación de base de datos en PostgreSQL

Tabla Usuario

```
CREATE TABLE Usuario (
    Id_Usuario SERIAL PRIMARY KEY,
    Nombre VARCHAR(100) NOT NULL,
    Email VARCHAR(150) UNIQUE NOT NULL,
    Contraseña TEXT NOT NULL,
    Tipo_Usuario VARCHAR(20) CHECK (Tipo_Usuario IN ('Artista', 'Oyente')) NOT
NULL
);
```

Tabla Álbum

```
CREATE TABLE Album (
    Id_album SERIAL PRIMARY KEY,
    Titulo VARCHAR(150) NOT NULL,
    Fecha_Lanzamiento DATE NOT NULL,
    Id_Usuario INT NOT NULL,
    FOREIGN KEY (Id_Usuario) REFERENCES Usuario (Id_Usuario) ON DELETE
CASCADE
);
```

Tabla Canción

```
CREATE TABLE Canción (
    Id_Cancion SERIAL PRIMARY KEY,
    Titulo VARCHAR(150) NOT NULL,
    Duracion INTERVAL NOT NULL,
    Fecha_Subida DATE NOT NULL DEFAULT CURRENT_DATE,
    Id_Album INT NOT NULL,
```

```
    FOREIGN KEY (Id_Album ) REFERENCES Album(Id_Album ) ON DELETE
CASCADE
);

Tabla Género
CREATE TABLE Género (
    Id_Genero SERIAL PRIMARY KEY,
    Nombre VARCHAR(50) UNIQUE NOT NULL
);

CREATE TABLE Canción_Género (
    id_cancion INT NOT NULL,
    id_genero INT NOT NULL,
    PRIMARY KEY (Id_Cancion, id_genero),
    FOREIGN KEY (Id_Cancion) REFERENCES Cancion(Id_Cancion) ON DELETE
CASCADE,
    FOREIGN KEY (Id_Genero) REFERENCES Género(Id_Genero) ON DELETE
CASCADE
);

Tabla Playlist
CREATE TABLE Playlist (
    Id_Playlist SERIAL PRIMARY KEY,
    Nombre VARCHAR(100) NOT NULL,
    Fecha_Creacion DATE NOT NULL DEFAULT CURRENT_DATE,
    Id_Usuario INT NOT NULL,
    FOREIGN KEY (Id_Usuario) REFERENCES usuario(Id_Usuario) ON DELETE
CASCADE
);

Tabla Playlist-Canción
CREATE TABLE Playlist_Cancion (
    id_playlist INT NOT NULL,
    id_cancion INT NOT NULL,
    PRIMARY KEY (Id_Playlist, Id_Cancion),
    FOREIGN KEY (Id_Playlist) REFERENCES Playlist(Id_Playlist) ON DELETE
CASCADE,
    FOREIGN KEY (Id_Cancion) REFERENCES Canción(Id_Cancion) ON DELETE
CASCADE
);

Tabla Historial de Reproducción
CREATE TABLE Historial (
    Id_Historial SERIAL PRIMARY KEY,
```

```
    Id_Usuario INT NOT NULL,
    Id_Cancion INT NOT NULL,
    Fecha_Reproduccion TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (Id_Usuario) REFERENCES Usuario(Id_Usuario) ON DELETE
CASCADE,
    FOREIGN KEY (Id_Canción) REFERENCES Canción(Id_Cancion) ON DELETE
CASCADE
);
```

6.- Sistema de Gestión de Citas Médicas.

Requerimientos:

• Los pacientes pueden agendar citas con los médicos.
• Cada cita debe incluir fecha, hora y estado (pendiente, atendida, cancelada).
• Los médicos deben contar con horarios específicos.

1.- Identificar las entidades del sistema.

¿Quiénes interactúan con el sistema?
R = Pacientes y Médicos

¿Qué acciones se realizan en el sistema?
R = Agendar una cita.

¿Los médicos tienen información relevante adicional?
R = Tienen una especialidad.

Entidades identificadas:

Paciente: Interactúa con el sistema y agenda citas.
Médico: Interactúa con el sistema y atiende citas.
Cita Médica: Es la acción principal del sistema.
Especialidad: Si un médico puede tener más de una especialidad.

2.- Definir atributos clave para cada entidad.

Paciente: Id_Paciente (PK), Nombre Completo (Es compuesto de: Nombres, Apellido Paterno y Apellido Materno), Fecha_Nacimiento, Telefono, Correo y Dirección(Es compuesto de: Calle, Número interno, Número externo, Colonia).

Médico: Id_Medico (PK), Nombre Completo (Es compuesto de: Nombres, Apellido Paterno y Apellido Materno), Telefono, Correo y Id_Especialidad (FK).

Cita: Id_Cita (PK), Id_Paciente (FK), Id_Medico (FK), Fecha, Hora y Estado.

Especialidad: Id_Especialidad (PK), Nombre, Descripcion.

Como los médicos pueden tener más de una especialidad creamos una tabla intermedia llamada: Médico_Especialidad.

Médico_Especialidad: Id_Medico (FK) y Id_Especialidad (FK).

3.- Establecer relaciones entre entidades.

• Paciente hace muchas citas.
• El Médico atiende muchas citas.
• El médico tiene una o varias especialidades.

Quedaria asi:
Paciente 1:N Cita.
Cita N:1 Médico.
Médico 1:N Médico_Especialidad M:1 Especialidad.

4.- Elegir clave primaria para identificación única.

Claves primarias:

Paciente: Id_Paciente (PK).
Médico: Id_Médico (PK).
Cita: Id_Cita (PK).
Especialidad: Id_Especialidad (PK).

Claves foráneas:

Cita: Id_Paciente (FK) y Id_Médico (FK).
Médico_Especialidad: Id_Médico (FK) y Id_Especialidad (FK).

5.- Refinar el diseño para optimizar la estructura.

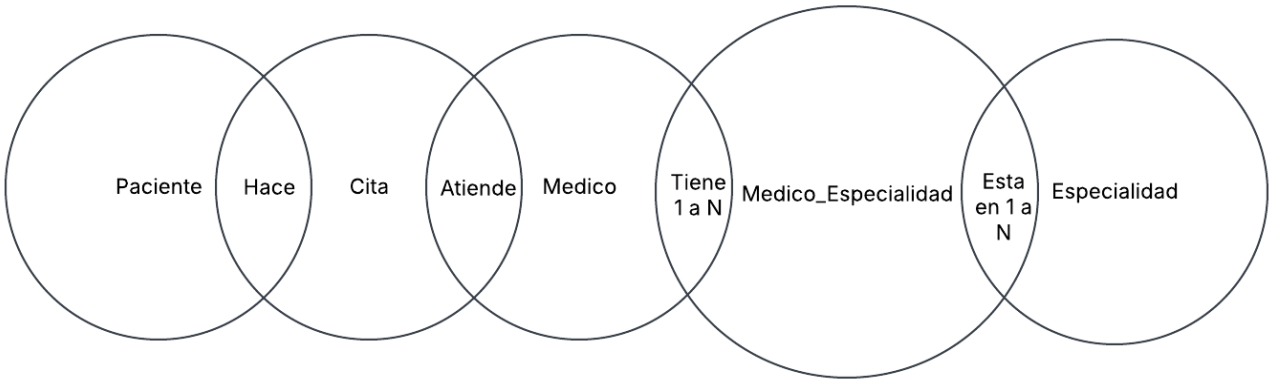
Por los pasos anteriores (es decir, del 1 al 4) ya está en las cuatro formas normales.

Restricción: No permitir citas duplicadas en el mismo horario.
Esta restriccion la tendria que cumplir en SQL con un Query, el cual sería el siguiente:

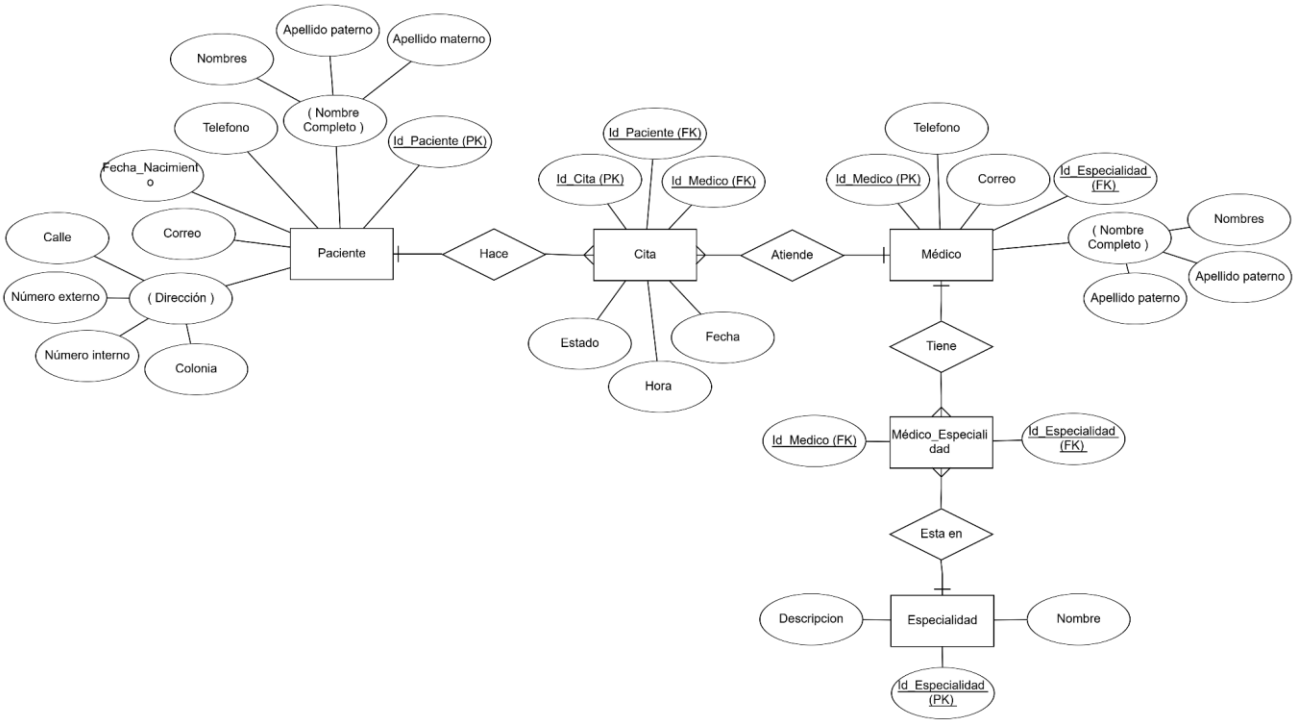
```
ALTER TABLE Cita_Medica
ADD CONSTRAINT unique_cita_medico
UNIQUE (Id_Medico, fecha, hora);
```

Esta restricción impide que se inserten citas duplicadas para un mismo médico en la misma fecha y hora.
Si se intenta agendar una cita en el mismo horario para el mismo médico, la base de datos rechazará la operación.

6.- Diagrama de Venn.



7.- Diagrama de Modelo ER.



8.- Creación de base de datos en PostgreSQL

Tabla Pacientes

```
CREATE TABLE Paciente (
    Id_Paciente SERIAL PRIMARY KEY,
    Nombres VARCHAR(100) NOT NULL,
```

```
    Apellido_Paterno VARCHAR(50) NOT NULL,
    Apellido_Materno VARCHAR(50),
    Fecha_Nacimiento DATE NOT NULL,
    Telefono VARCHAR(20),
    Correo VARCHAR(100) UNIQUE,
    Calle VARCHAR(100) NOT NULL,
    Numero_Externo VARCHAR(10) NOT NULL,
    Numero_Interno VARCHAR(10), -- Opcional
    Colonia VARCHAR(100) NOT NULL
);
```

Tabla Medico

```
CREATE TABLE Medico (
    Id_Medico SERIAL PRIMARY KEY,
    Nombres VARCHAR(100) NOT NULL,
    Apellido_Paterno VARCHAR(50) NOT NULL,
    Apellido_Materno VARCHAR(50),
    Telefono VARCHAR(20),
    Correo VARCHAR(100) UNIQUE
);
```

Tabla Medico_Especialidad

```
CREATE TABLE Medico_Especialidad (
    Id_Medico INT REFERENCES Medico(id_medico) ON DELETE CASCADE,
    Id_Especialidad INT REFERENCES Especialidad(Id_Especialidad) ON DELETE
CASCADE,
    PRIMARY KEY (Id_Medico, Id_Especialidad)
);
```

Tabla Cita

```
CREATE TABLE Cita_Medica (
    Id_Cita SERIAL PRIMARY KEY,
    Id_Paciente INT REFERENCES Paciente(Id_Paciente) ON DELETE CASCADE,
    Id_Medico INT REFERENCES Medico(Id_Medico) ON DELETE CASCADE,
    Fecha DATE NOT NULL,
    Hora TIME NOT NULL,
    estado VARCHAR(20) CHECK (estado IN ('pendiente', 'atendida', 'cancelada')),
    CONSTRAINT unique_cita_medico UNIQUE (Id_Medico, Fecha, Hora)  -- Regla de
negocio: No citas duplicadas
);
```

7.- Biblioteca Digital

Requerimientos:
Usuarios pueden tomar libros prestados
Se debe registrar la fecha de devolución esperada.
Solo solo ciertos usuarios pueden administrar libros.

1.- Identificar las entidades del sistema.

¿Quiénes interactúan con el sistema?
R = Usuarios y Administradores

¿Qué se gestiona?
R = Libros.

¿Qué acciones requieren seguimiento?
R = Préstamos.

Entidades identificadas:

Usuarios: pueden tomar libros prestados.
Administradores: usuarios con permisos especiales para gestionar libros.
Libros: son prestados y devueltos.
Préstamos: registran qué usuario tomó qué libro y la fecha esperada de devolución.

2.- Definir atributos clave para cada entidad.

Usuario: Id_Usuario (PK), Nombre, Email, Contraseña y Tipo_Usuario.

Libro: Id_Libro (PK), Título, Autor, Año Publicación y Género

Préstamo: Id_Prestamo (PK), Id_Usuario (FK), Id_Libro (FK), Fecha_Prestamo y Fecha_Devolucion_Esperada

3.- Establecer relaciones entre entidades.

• Usuario puede pedir libros.
• Un Préstamo asocia un Usuario con un Libro.

Usuario 1:N Prestamo M:1 Libro.

4.- Elegir clave primaria para identificación única.

Claves primarias:

• Id_Usuario en Usuario
• Id_Libro en Libro
• Id_Prestamo en Préstamo

Claves foráneas:

• Id_Usuario en Préstamo, referenciando a Usuario.
•Id_Libro en Préstamo, referenciando a Libro.

5.- Refinar el diseño para optimizar la estructura.

Normalización

- 1FN: Cada atributo almacena un solo valor y cada fila es única.
- 2FN: No hay dependencias parciales (todos los atributos dependen de la PK).
- 3FN: No hay dependencias transitivas.
- 4FN: No hay dependencias multivaluadas.

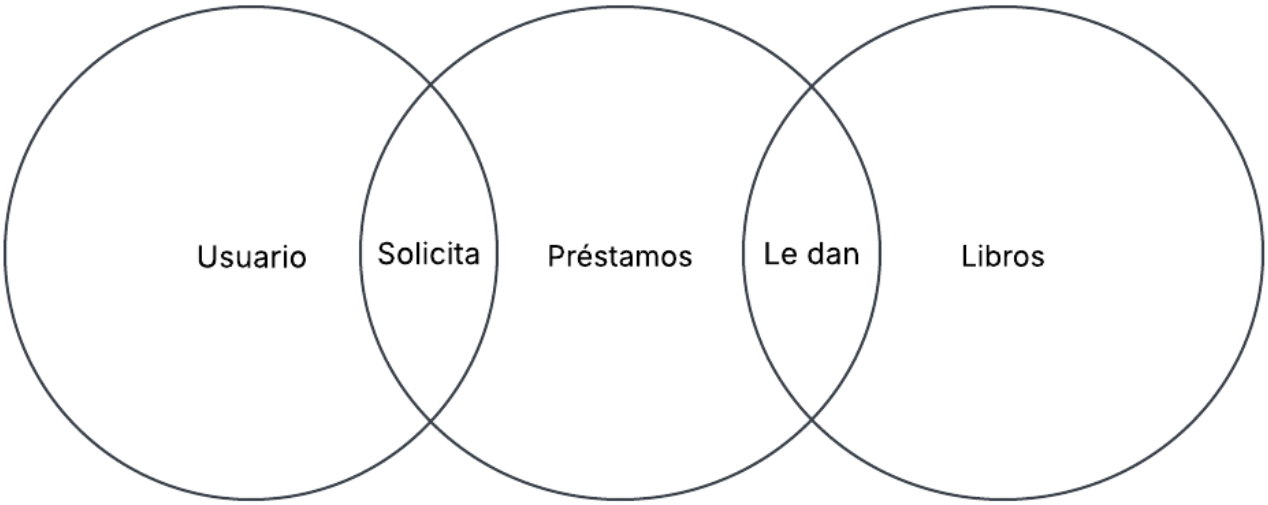
Reglas de negocio

- Un usuario no puede tomar más de 3 libros prestados a la vez.
- Solo ciertos usuarios (administradores) pueden gestionar libros.

Seguridad

- Hash de contraseñas en la tabla Usuario.
- Roles y permisos: Solo los administradores pueden agregar o eliminar libros.

6.- Diagrama de Venn.



7.- Diagrama de Modelo ER.

8.- Creación de base de datos en PostgreSQL

Tabla Usuario

```
CREATE TABLE Usuario (
    Id_Usuario SERIAL PRIMARY KEY,
    Nombre VARCHAR(100) NOT NULL,
    Email VARCHAR(100) UNIQUE NOT NULL,
    Contraseña TEXT NOT NULL,
```

```
   Tipo_Usuario VARCHAR(20) CHECK (Tipo_Usuario IN ('Lector', 'Administrador'))
NOT NULL
);

Tabla Libro
CREATE TABLE Libro (
   Id_Libro SERIAL PRIMARY KEY,
   Titulo VARCHAR(200) NOT NULL,
   Autor VARCHAR(100) NOT NULL,
   Año_Publicación INT CHECK (Año_Publicación > 0),
   genero VARCHAR(50)
);

Tabla Préstamo
CREATE TABLE Prestamo (
   Id_Prestamo SERIAL PRIMARY KEY,
   Id_Usuario INT REFERENCES Usuario(id_usuario) ON DELETE CASCADE,
   Id_Libro INT REFERENCES Libro(id_libro) ON DELETE CASCADE,
   Fecha_Prestamo DATE NOT NULL DEFAULT CURRENT_DATE,
   Eecha_Devolucion_Esperada DATE NOT NULL,
   CONSTRAINT limite_prestamos CHECK (
      (SELECT COUNT(*) FROM Prestamo p WHERE p.Id_Usuario =
Prestamo.Id_Usuario) <= 3)
);

//Seguridad: Crear roles y permisos
CREATE ROLE lector;
CREATE ROLE administrador;

//Permisos para lectores (sólo pueden consultar libros y préstamos)
GRANT SELECT ON Libro TO lector;
GRANT SELECT, INSERT, UPDATE ON Prestamo TO lector;

//Permisos para administradores (pueden gestionar libros)
GRANT ALL PRIVILEGES ON Libro TO administrador;
GRANT ALL PRIVILEGES ON Usuario TO administrador;
GRANT ALL PRIVILEGES ON Prestamo TO administrador;
```

7. Sistema de Gestión de Proyectos

Requerimientos:

- Las empresas pueden registrar proyectos.

- Cada proyecto incluye multiples tareas y responsables.
- Se deben regiostrat los avances de cada tarea.

Entidades del sistema.

- Empresas: Representa quien registra los proyectos.
- Proyectos: Representa los proyectos disponibles por empresa.
- Tareas: Se registran tareas asignas al proyecto.
- Usuarios: Personas asignadas a los proyectos.

Atributos clave para cada entidad.

- Empresas: ID_Empresa (PK), Nombre, Correo.
- Proyectos: ID_Proyecto (PK), ID_Empresa (FK), ID_Usuario (FK), Nombre, Fecha.
- Tareas: ID_Tarea (PK), ID_Proyecto (FK), ID_Usuario (FK), Fecha, Avance, Nombre
- Usuario: ID_Usuario (PK), Nombre

Relaciones entre entidades.

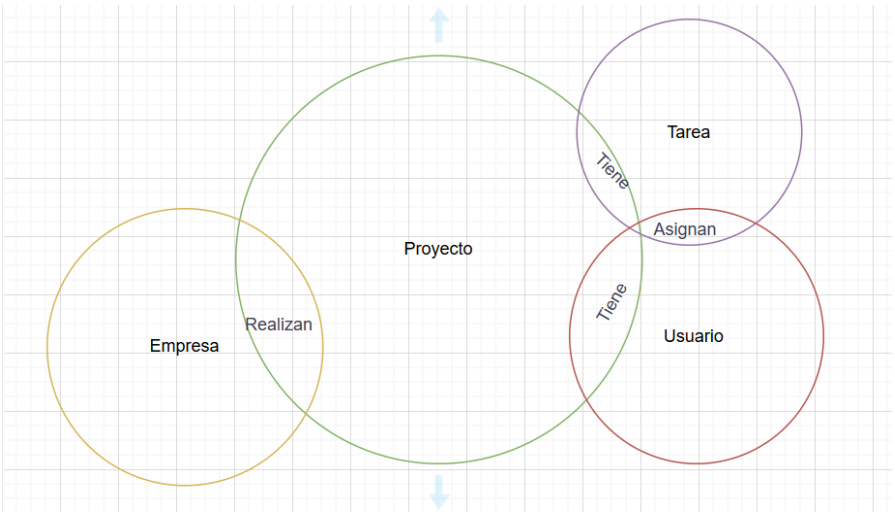
- Una Empresa puede tener multiples Proyectos.
- Cada Proyecto puede tener multiples Tareas.
- Cada Proyecto puede ser asignado a varios Usuarios.
- Una tarea puede ser asignada a un Usuario

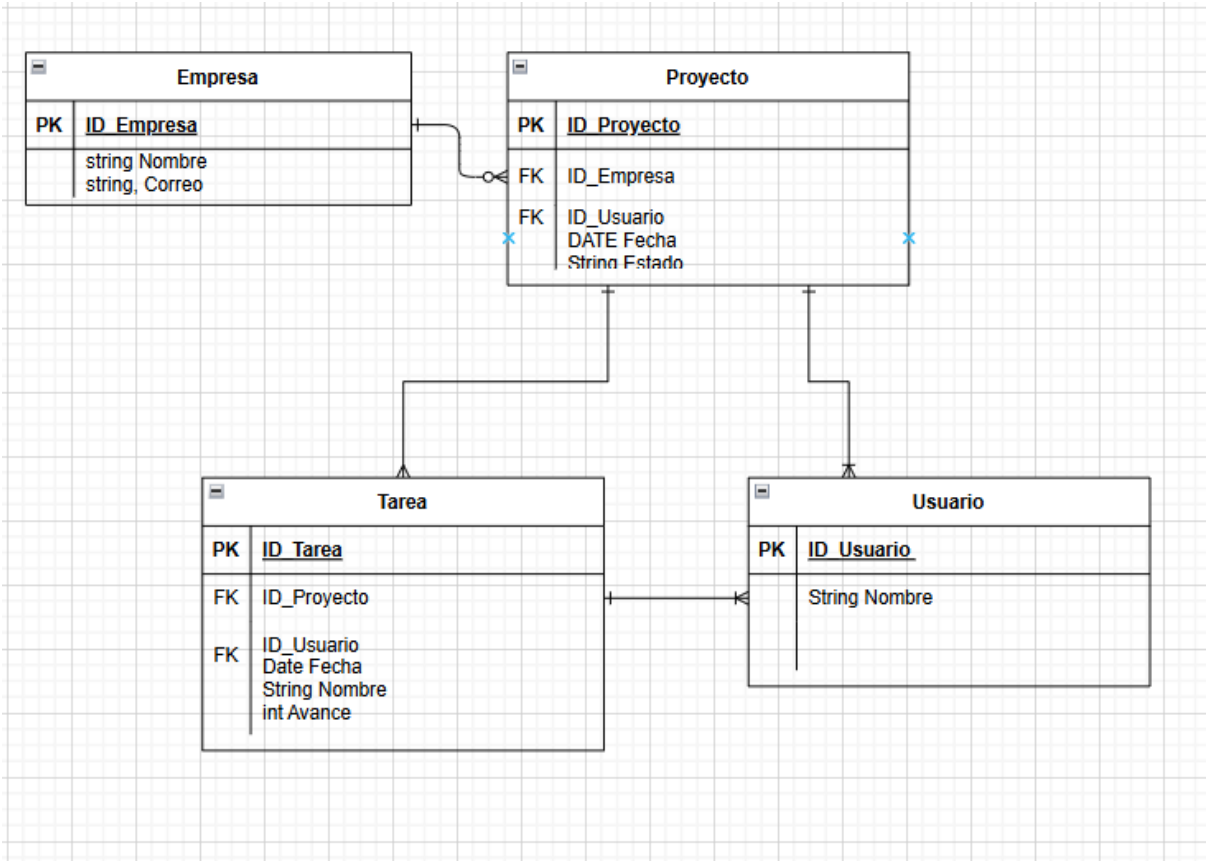
Claves primarias para identificación única.

- ID_Empresa como clave primaria de Empresa
- ID_Proyecto como clave primaria de Proyecto
- ID_Tarea como clave primaria de Tarea
- ID_Usuario como clave primaria de Usuario

Diagramas de Venn y E-R

-





Tablas:

```
CREATE TABLE Empresas (
ID_Empresa SERIAL PRIMARY KEY,
Nombre VARCHAR(50) NOT NULL,
Correo VARCHAR(50) UNIQUE NOT NULL
 );

CREATE TABLE Usuarios (
 ID_Usuario SERIAL PRIMARY KEY,
Nombre VARCHAR(50) NOT NULL
);

CREATE TABLE Proyectos (
 ID_Proyecto SERIAL PRIMARY KEY,
ID_Empresa INT REFERENCES Empresas(ID_Empresa) ON DELETE CASCADE,
Nombre VARCHAR(255) NOT NULL,
Fecha DATE NOT NULL,
 ID_Usuario INT REFERENCES Usuarios(ID_Usuario)
);

CREATE TABLE Tareas (
ID_Tarea SERIAL PRIMARY KEY,
 ID_Proyecto INT REFERENCES Proyectos(ID_Proyecto),
```

```
ID_Usuario INT REFERENCES Usuarios(ID_Usuario),
 Nombre VARCHAR(255) NOT NULL,
Fecha DATE,
 Avance INT CHECK (Avance BETWEEN 0 AND 100)
);
```

8. Red Social

Requerimientos:

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

Entidades del Sistema:

- Usuarios: Representa a las personas que usan la red social
- Publicaciones: Contiene las publicaciones hechas por los usuarios.
- Reacciones: Son las reacciones que los usuarios dan a las publicaciones.
- Comentarios: Relaciona los comentarios de los usuarios a las publicaciones.

Atributos clave para cada entidad.

- Usuarios: ID_Usuario (PK), Nombre, Correo
- Publicaciones: ID_Publicacion(PK), ID_Usuario(FK), Texto,
- Reacciones: ID_Reaccion (PK), ID_Usuario (FK), ID_Publicacion (FK), Reaccion.
- Comentarios: ID_Comentario (PK), ID_Publicacion (FK), Texto.

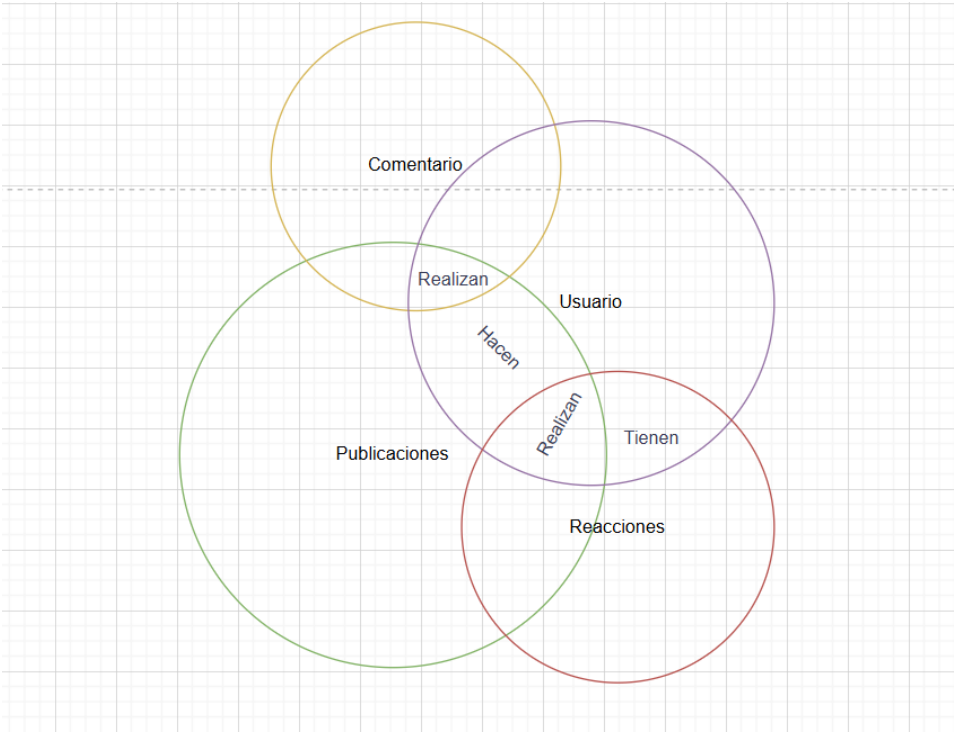
Relación entre entidades.

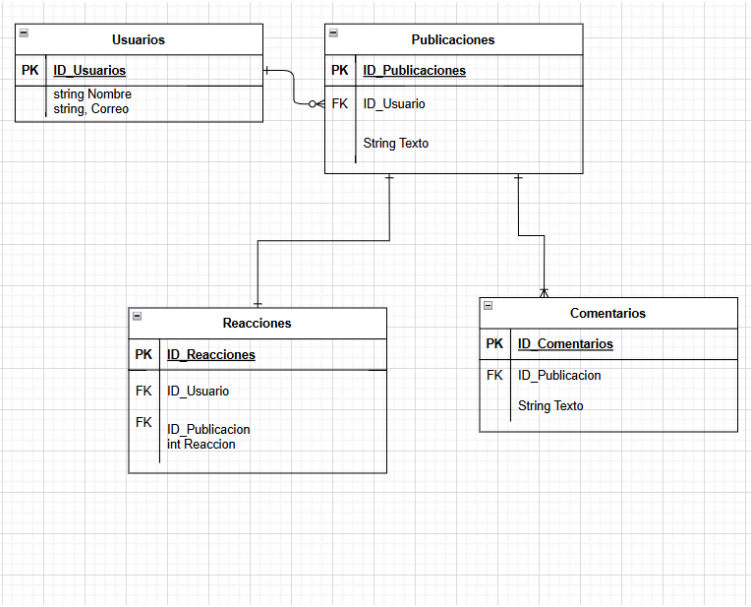
- Los Usuarios pueden hacer muchas Publicaciones.
- Se puede Reaccionar una sola vez por Publicacion.
- Se pueden Comentar varias veces por Publicacion

Claves primarias para identificación única.

- ID_Usuario como clave primaria de Usuario
- ID_Publicaciones como clave primaria de Publicaciones
- ID_Reacciones como clave primaria de Reacciones
- ID_Comentarios como clave primaria de Comentarios

Diagramas de Venn y E-R





Tablas

Tablas:

```
CREATE TABLE Usuarios (
    ID_Usuario SERIAL PRIMARY KEY,
    Nombre VARCHAR(50) NOT NULL,
    Correo VARCHAR(50) UNIQUE NOT NULL,
);
CREATE TABLE Publicaciones (
    ID_Publicacion SERIAL PRIMARY KEY,
```

```
    ID_Usuario INT REFERENCES Usuarios(ID_Usuario),
    Texto TEXT NOT NULL,
);
CREATE TABLE Reacciones (
    ID_Reaccion SERIAL PRIMARY KEY,
    ID_Usuario INT REFERENCES Usuarios(ID_Usuario),
    ID_Publicacion INT REFERENCES Publicaciones(ID_Publicacion),
);
CREATE TABLE Comentarios (
    ID_Comentario(PK),
    ID_Publicacion(FK),
    Texto TEXT NOT NULL
);
```

9. Sistema de Facturación

Requerimientos:

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

Entidades y Atributos:

- Clientes: Representa a la persona que solicita la factura
- Productos: Es la mercancía que se observa en Detalle Factura
- Facturas: Información sobre el pedido del cliente
- Detalle_Factura: Detalles de los productos de cada factura

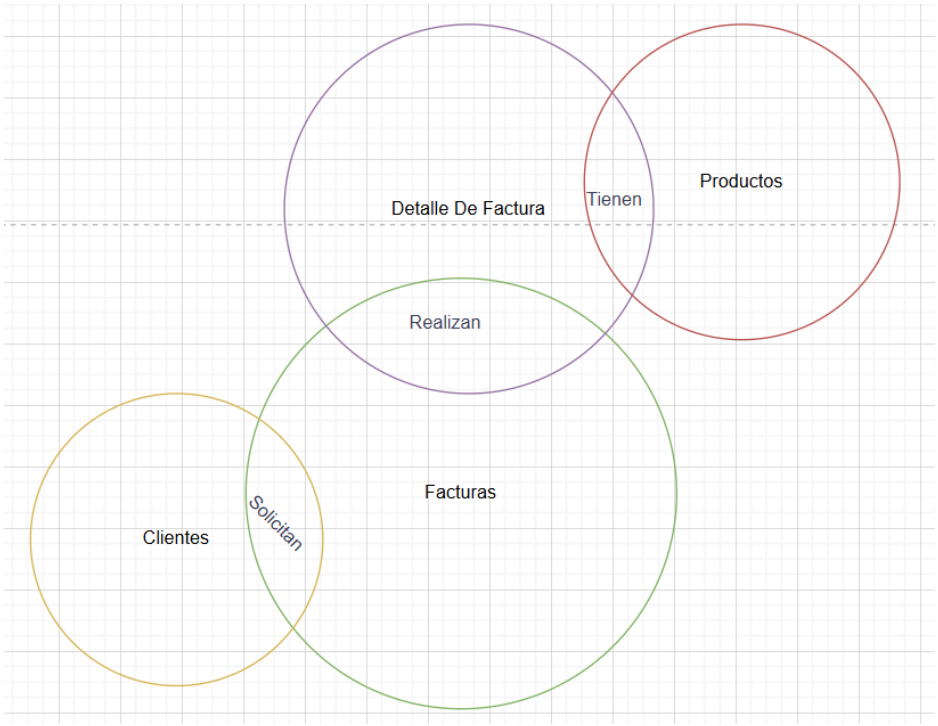
Atributos clave para cada entidad.

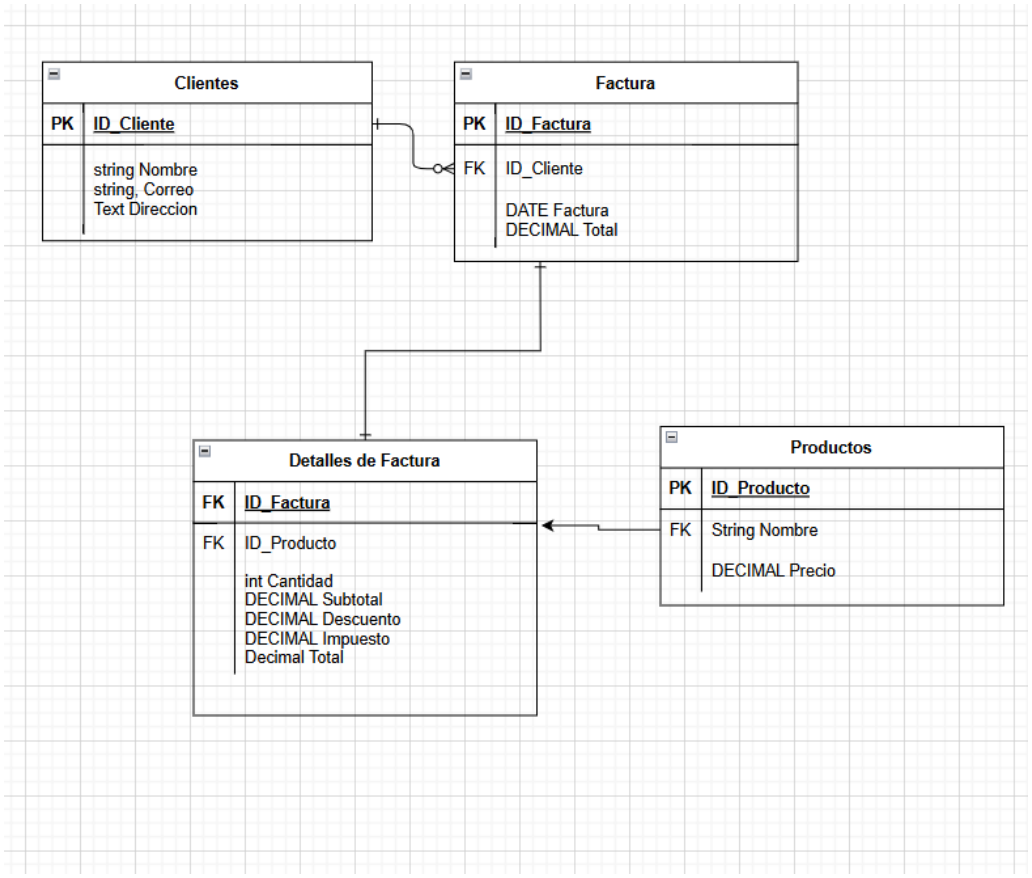
- Clientes: ID_Cliente (PK), Nombre, Correo, Dirección.
- Productos: ID_Producto (PK), Nombre, Precio.
- Facturas: ID_Factura(PK), ID_Cliente (FK), Fecha, Total.
- Detalle_Factura: ID_Factura(FK), ID_Producto(FK), Cantidad, Subtotal, Descuento, Impuesto, Total.

Relación entre entidades.

- Un Cliente puede tener muchas Facturas
- Una Factura puede tener muchos Detalles de Factura
- Un Producto puede estar en muchos Detalles de Factura

Diagramas de Venn y E-R


Detalle De Factura
Tienen
Productos
Realizan
Facturas
Clientes
Solicitan


Clientes
PK ID_Cliente
string Nombre
string, Correo
Text Direccion
Factura
PK ID_Factura
FK ID_Cliente
DATE Factura
DECIMAL Total
Detalles de Factura
FK ID_Factura
FK ID_Producto
int Cantidad
DECIMAL Subtotal
DECIMAL Descuento
DECIMAL Impuesto
Decimal Total
Productos
PK ID_Producto
FK String Nombre
DECIMAL Precio

Tablas.

```
CREATE TABLE Clientes (
    ID_Cliente SERIAL PRIMARY KEY,
    Nombre VARCHAR(50) NOT NULL,
    Correo VARCHAR(50) UNIQUE NOT NULL,
    Direccion TEXT
);

CREATE TABLE Productos (
    ID_Producto SERIAL PRIMARY KEY,
    Nombre VARCHAR(50) NOT NULL,
    Precio DECIMAL(10, 2) NOT NULL
);

CREATE TABLE Facturas (
    ID_Factura SERIAL PRIMARY KEY,
    ID_Cliente INT REFERENCES Clientes(ID_Cliente),
    Fecha DATE NOT NULL,
    Total DECIMAL(10, 2) NOT NULL
);

CREATE TABLE Detalle_Factura (
    ID_Factura INT REFERENCES Facturas(ID_Factura),
    ID_Producto INT REFERENCES Productos(ID_Producto),
    Cantidad INT NOT NULL,
    Subtotal DECIMAL(10, 2) NOT NULL,
    Descuento DECIMAL(5, 2) DEFAULT 0,
    Impuesto DECIMAL(5, 2) DEFAULT 0,
    Total DECIMAL(10, 2) NOT NULL
);
```