

INSTITUTO TECNOLÓGICO DE MEXICALI



FUNDAMENTOS DE BASE DE DATOS

TAREA 3

MAESTRA: JOSE RAMON BOGARIN VALENZUELAJOSE

RAMON BOGARIN VALENZUELA

NOMBRE:

- REBECA ELIZABETT MARTINEZ RENDON

NUMERO DE CONTROL:

23490390

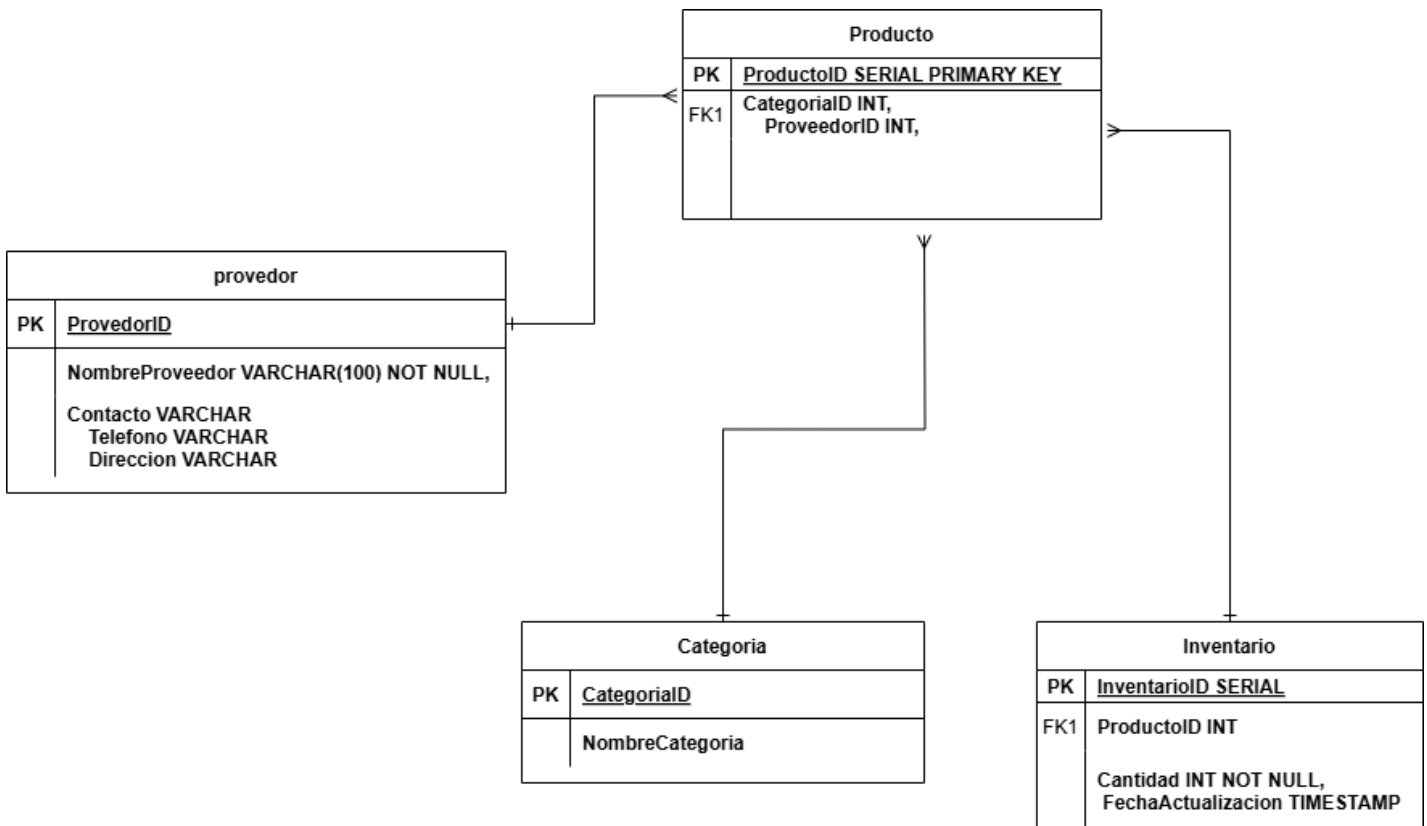
# 1. Sistema de Gestión de Inventarios

## Identificar entidades clave:

- Producto
- Proveedor
- Categoría
- Inventario.

## Relaciones principales:

- Categoría → Producto (Uno a Muchos)
- Proveedor → Producto (Uno a Muchos)
- Producto → Inventario (Uno a Muchos)



**CREATE DATABASE GestionInventarios;**

**CREATE TABLE Categoria (  
    CategorialID SERIAL PRIMARY KEY,  
    NombreCategoria VARCHAR(100) NOT NULL  
);**

**CREATE TABLE Proveedor (  
    ProveedorID SERIAL PRIMARY KEY,  
    NombreProveedor VARCHAR(100) NOT NULL,  
    Contacto VARCHAR(100),  
    Telefono VARCHAR(15),  
    Direccion VARCHAR(255)  
);**

**CREATE TABLE Producto (  
    ProductID SERIAL PRIMARY KEY,  
    NombreProducto VARCHAR(100) NOT NULL,  
    Precio DECIMAL(10,2) NOT NULL,  
    CategorialID INT,  
    ProveedorID INT,  
    FOREIGN KEY (CategorialID) REFERENCES Categoria(CategorialID),  
    FOREIGN KEY (ProveedorID) REFERENCES Proveedor(ProveedorID)  
);**

**CREATE TABLE Inventario (  
    InventariID SERIAL PRIMARY KEY,  
    ProductID INT,  
    Cantidad INT NOT NULL,  
    FechaActualizacion TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
    FOREIGN KEY (ProductID) REFERENCES Producto(ProductID)  
);**

```
INSERT INTO Proveedor (NombreProveedor, Contacto, Telefono,
Direccion) VALUES
('Proveedor A', 'Juan Pérez', '123456789', 'Calle Falsa 123'),
('Proveedor B', 'Ana Gómez', '987654321', 'Avenida Principal 456'),
('Proveedor C', 'Carlos López', '1122334455', 'Calle 45, Ciudad X'),
('Proveedor D', 'María Rodríguez', '5566778899', 'Calle 20, Zona Norte'),
('Proveedor E', 'Roberto Pérez', '9988776655', 'Avenida 123, Ciudad Y');
```

```
INSERT INTO Producto (NombreProducto, Precio, CategoricalID,
ProveedorID) VALUES
('Televisor', 300.00, 1, 1),
('Laptop', 500.00, 1, 2),
('Camiseta', 20.00, 2, 1),
('Manzanas', 1.50, 3, 2),
('Silla de oficina', 85.00, 4, 1),
('Mesa de comedor', 120.00, 4, 2),
('Taladro eléctrico', 45.00, 5, 3),
('Juguete de construcción', 15.00, 6, 4),
('Balón de fútbol', 20.00, 7, 5),
('Bicicleta', 150.00, 7, 2),
('Sofá', 300.00, 3, 1),
('Cuna de bebé', 100.00, 3, 4);
```

```
INSERT INTO Inventario (ProductID, Cantidad) VALUES
(1, 10),
(2, 5),
(3, 50),
(4, 200),
(5, 50),
(6, 30),
(7, 15),
(8, 25),
(9, 40),
(10, 60),
(11, 10),
(12, 20);
```

- **Consulta requerida:** Obtener la lista de productos con sus respectivas categorías y proveedores, ordenados alfabéticamente por nombre de producto.

```
SELECT P.NombreProducto, C.NombreCategoria, V.NombreProveedor
FROM Producto P
LEFT JOIN Categoria C ON P.CategoriaID = C.CategoriaID
LEFT JOIN Proveedor V ON P.ProveedorID = V.ProveedorID
ORDER BY P.NombreProducto ASC;
```

	nombreproducto character varying (100) 🔒	nombrecategoria character varying (100) 🔒	nombreproveedor character varying (100) 🔒
1	Balón de fútbol	Deportes	Proveedor E
2	Bicicleta	Deportes	Proveedor B
3	Camiseta	Ropa	Proveedor A
4	Cuna de bebé	Alimentos	Proveedor D
5	Juguete de construcción	Juguetes	Proveedor D
6	Laptop	Electrónica	Proveedor B
7	Manzanas	Alimentos	Proveedor B
8	Mesa de comedor	Muebles	Proveedor B
9	Silla de oficina	Muebles	Proveedor A
10	Sofá	Alimentos	Proveedor A
11	Taladro eléctrico	Herramientas	Proveedor C
12	Televisor	Electrónica	Proveedor A

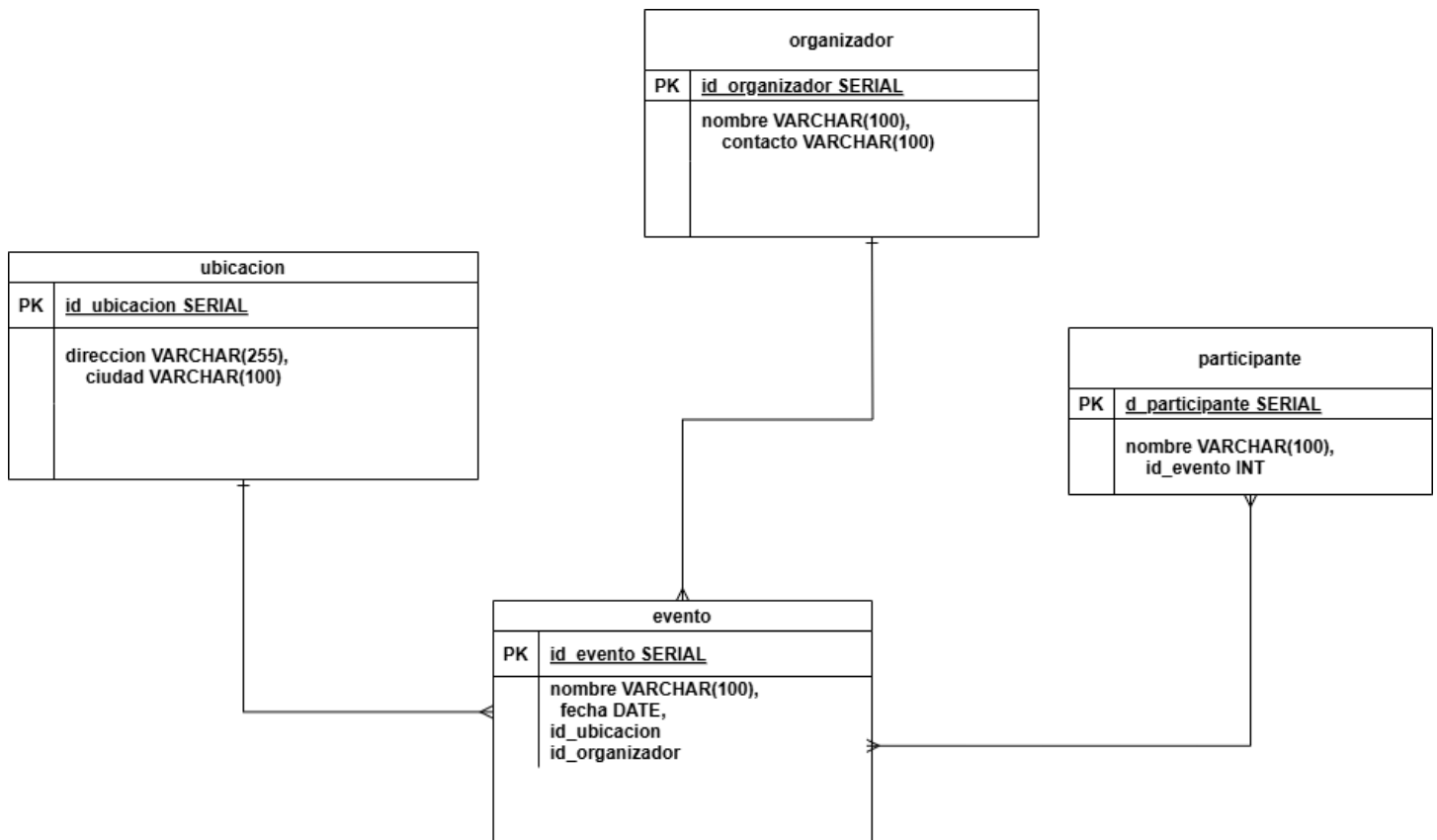
## 2. Sistema de Gestión de Eventos

### Definir entidades clave:

- Evento
- Participante
- Ubicación
- Organizador

### Relaciones principales:

Ubicación → Evento (Uno a Muchos)  
Organizador → Evento (Uno a Muchos)  
Evento → Participante (Muchos a Muchos)



```
CREATE TABLE ubicacion (  
    id_ubicacion SERIAL PRIMARY KEY,  
    direccion VARCHAR(255),  
    ciudad VARCHAR(100)  
);  
  
-- Tabla de Organizador  
CREATE TABLE organizador (  
    id_organizador SERIAL PRIMARY KEY,  
    nombre VARCHAR(100),  
    contacto VARCHAR(100)  
);  
  
-- Tabla de Evento  
CREATE TABLE evento (  
    id_evento SERIAL PRIMARY KEY,  
    nombre VARCHAR(100),  
    fecha DATE,  
    id_ubicacion INT REFERENCES ubicacion(id_ubicacion),  
    id_organizador INT REFERENCES organizador(id_organizador)  
);  
  
-- Tabla de Participante  
CREATE TABLE participante (  
    id_participante SERIAL PRIMARY KEY,  
    nombre VARCHAR(100),  
    id_evento INT REFERENCES evento(id_evento)  
);
```

```
CREATE INDEX idx_evento_fecha ON evento(fecha);  
  
INSERT INTO ubicacion (direccion, ciudad)  
VALUES  
    ('Calle Falsa 123', 'Ciudad A'),  
    ('Avenida Siempre Viva 742', 'Ciudad B'),  
    ('Calle del Sol 456', 'Ciudad C');
```

```
INSERT INTO organizador (nombre, contacto)
VALUES
('Organización A', 'contacto@org-a.com'),
('Eventos B', 'contacto@eventos-b.com'),
('Soluciones C', 'contacto@soluciones-c.com');
```

```
INSERT INTO evento (nombre, fecha, id_ubicacion, id_organizador)
VALUES
('Concierto de Rock', '2025-04-15', 1, 1),
('Congreso de Tecnología', '2025-05-20', 2, 2),
('Festival de Cine', '2025-06-10', 3, 3);
```

```
INSERT INTO participante (nombre, id_evento)
VALUES
('Juan Pérez', 1),
('María González', 1),
('Carlos López', 2),
('Ana Díaz', 2),
('Pedro Martínez', 3);
```






- **Consulta requerida:** Obtener la lista de eventos programados junto con la cantidad de participantes registrados por evento.

```

SELECT
  e.id_evento,
  e.nombre AS evento_nombre,
  COUNT(p.id_participante) AS participantes
FROM
  evento e
LEFT JOIN
  participante p ON e.id_evento = p.id_evento
GROUP BY
  e.id_evento, e.nombre
ORDER BY
  e.fecha;

```

	id_evento [PK] integer 	evento_nombre character varying (100) 	participantes bigint 
1	1	Concierto de Rock	2
2	2	Congreso de Tecnología	2
3	3	Festival de Cine	1

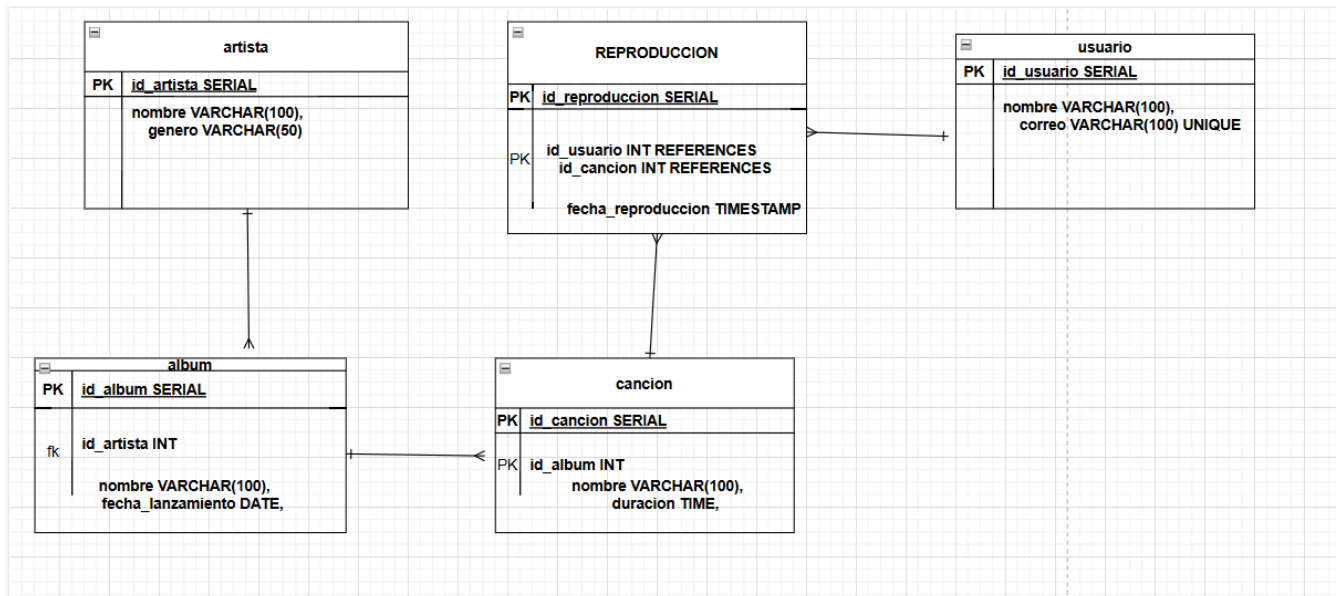
### 3. Plataforma de Streaming de Música

#### Identificar entidades clave:

- Usuario,
- Artista
- Álbum,
- Canción

#### Relaciones principales:

- Artista → Álbum (uno a Muchos)
- Álbum → Canción (uno a Muchos)
- Usuario → Reproducción (uno a Muchos)
- Canción → Reproducción (uno a Muchos)



```
CREATE TABLE usuario (  
  id_usuario SERIAL PRIMARY KEY,  
  nombre VARCHAR(100),  
  correo VARCHAR(100) UNIQUE  
);  
  
-- Tabla de Artista  
CREATE TABLE artista (  
  id_artista SERIAL PRIMARY KEY,  
  nombre VARCHAR(100),  
  genero VARCHAR(50)  
);  
  
-- Tabla de Álbum  
CREATE TABLE album (  
  id_album SERIAL PRIMARY KEY,  
  nombre VARCHAR(100),  
  fecha_lanzamiento DATE,  
  id_artista INT REFERENCES artista(id_artista)  
);  
  
-- Tabla de Canción  
CREATE TABLE cancion (  
  id_cancion SERIAL PRIMARY KEY,  
  nombre VARCHAR(100),  
  duracion TIME,  
  id_album INT REFERENCES album(id_album)  
);  
  
-- Tabla de Reproducción  
CREATE TABLE reproduccion (  
  id_reproduccion SERIAL PRIMARY KEY,  
  id_usuario INT REFERENCES usuario(id_usuario),  
  id_cancion INT REFERENCES cancion(id_cancion),  
  fecha_reproduccion TIMESTAMP  
);
```

```
INSERT INTO usuario (nombre, correo)
VALUES
('Juan Pérez', 'juan.perez@example.com'),
('María López', 'maria.lopez@example.com'),
('Carlos González', 'carlos.gonzalez@example.com');
```

```
INSERT INTO album (nombre, fecha_lanzamiento, id_artista)
VALUES
('Álbum 1', '2025-01-01', 1),
('Álbum 2', '2025-02-15', 2),
('Álbum 3', '2025-03-10', 3);
```

```
INSERT INTO cancion (nombre, duracion, id_album)
VALUES
('Canción 1A', '00:03:45', 1),
('Canción 2A', '00:04:00', 1),
('Canción 1B', '00:02:50', 2),
('Canción 2B', '00:05:30', 2),
('Canción 1C', '00:03:10', 3);
```

```
INSERT INTO reproduccion (id_usuario, id_cancion,
fecha_reproduccion)
VALUES
(1, 1, '2025-03-23 10:00:00'),
(1, 2, '2025-03-23 10:05:00'),
(2, 3, '2025-03-23 11:00:00'),
(2, 4, '2025-03-23 11:10:00'),
(3, 5, '2025-03-23 12:00:00');
```

- **Consulta requerida:** Listar las canciones reproducidas por un usuario específico, incluyendo el nombre del artista y del álbum.

```
SELECT
  u.nombre AS usuario_nombre,
  c.nombre AS cancion_nombre,
  a.nombre AS album_nombre,
  ar.nombre AS artista_nombre,
  r.fecha_reproduccion
FROM
  reproduccion r
JOIN
  usuario u ON r.id_usuario = u.id_usuario
JOIN
  cancion c ON r.id_cancion = c.id_cancion
JOIN
  album a ON c.id_album = a.id_album
JOIN
  artista ar ON a.id_artista = ar.id_artista
WHERE
  u.id_usuario =
ORDER BY
  r.fecha_reproduccion;
```

	usuario_nombre character varying (100) 🔒	cancion_nombre character varying (100) 🔒	album_nombre character varying (100) 🔒	artista_nombre character varying (100) 🔒	fecha_reproduccion timestamp without time zone 🔒
1	Juan Pérez	Canción 1A	Álbum 1	Artist A	2025-03-23 10:00:00
2	Juan Pérez	Canción 2A	Álbum 1	Artist A	2025-03-23 10:05:00

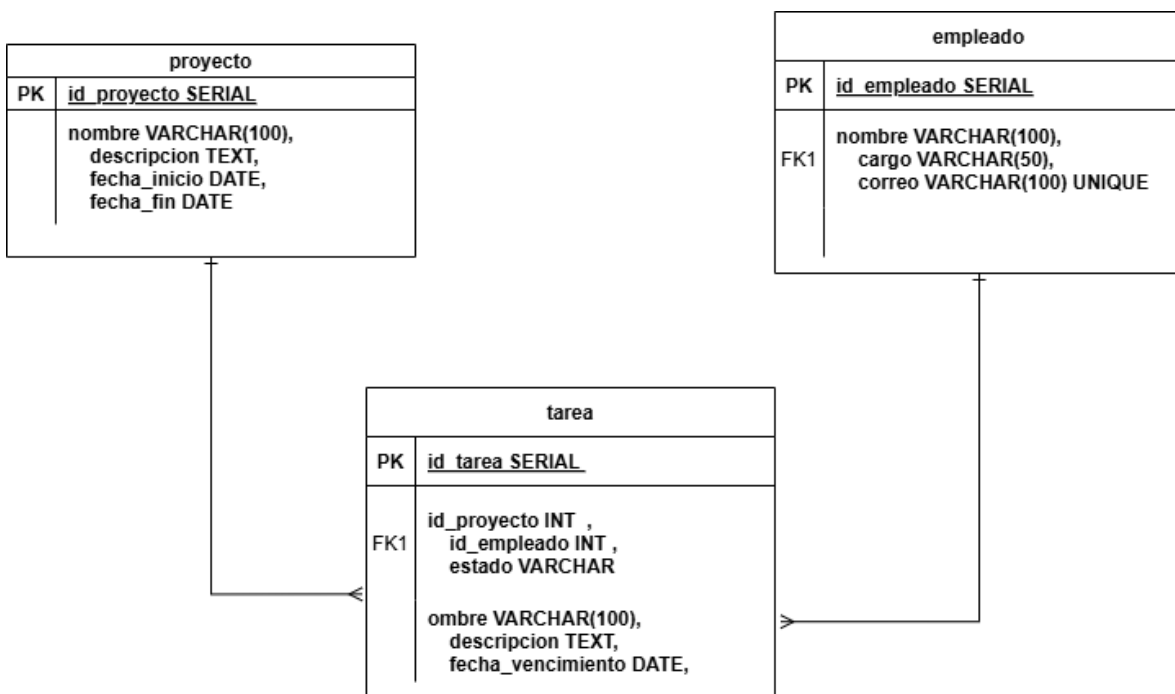
## 4. Sistema de Control de Proyectos

Definir entidades clave:

- Proyecto
- Empleado
- Tarea.

Relaciones principales:

- Proyecto → Tarea (Uno a Muchos)
- Empleado → Tarea (Uno a Muchos)



```
CREATE TABLE proyecto (  
    id_proyecto SERIAL PRIMARY KEY,  
    nombre VARCHAR(100),  
    descripcion TEXT,  
    fecha_inicio DATE,  
    fecha_fin DATE  
);  
  
CREATE TABLE empleado (  
    id_empleado SERIAL PRIMARY KEY,  
    nombre VARCHAR(100),  
    cargo VARCHAR(50),  
    correo VARCHAR(100) UNIQUE  
);  
  
CREATE TABLE tarea (  
    id_tarea SERIAL PRIMARY KEY,  
    nombre VARCHAR(100),  
    descripcion TEXT,  
    fecha_vencimiento DATE,  
    id_proyecto INT REFERENCES proyecto(id_proyecto),  
    id_empleado INT REFERENCES empleado(id_empleado),  
    estado VARCHAR(20) CHECK (estado IN ('Pendiente', 'En progreso',  
    'Completada'))  
);
```

```
INSERT INTO empleado (nombre, cargo, correo)  
VALUES  
    ('Juan Pérez', 'Desarrollador', 'juan.perez@empresa.com'),  
    ('María López', 'Gerente de Proyecto', 'maria.lopez@empresa.com'),  
    ('Carlos González', 'Administrador de Red', 'carlos.gonzalez@empresa.com');
```

- **Consulta requerida:** Mostrar todas las tareas pendientes de un proyecto específico, ordenadas por fecha de vencimiento.

```
INSERT INTO tarea (nombre, descripcion, fecha_vencimiento, id_proyecto, id_empleado, estado)
VALUES
    ('Desarrollar módulo de autenticación', 'Crear el módulo de inicio de sesión para la aplicación.',
    '2025-03-15', 1, 1, 'Pendiente'),
    ('Instalar servidores', 'Instalar servidores en las oficinas centrales.', '2025-03-30', 2, 3,
    'Pendiente'),
    ('Pruebas de seguridad', 'Realizar pruebas de seguridad del software.', '2025-04-10', 1, 1, 'En
    progreso'),
    ('Revisión de red', 'Revisar la configuración de la red interna.', '2025-03-25', 2, 3, 'Pendiente');
```

```
SELECT
    t.id_tarea,
    t.nombre AS tarea_nombre,
    t.descripcion AS tarea_descripcion,
    t.fecha_vencimiento,
    e.nombre AS empleado_nombre,
    t.estado
FROM
    tarea t
JOIN
    empleado e ON t.id_empleado = e.id_empleado
WHERE
    t.id_proyecto = 1
    AND t.estado = 'Pendiente'
ORDER BY
    t.fecha_vencimiento;
```

	id_tarea integer	tarea_nombre character varying (100)	tarea_descripcion text	fecha_vencimiento date	empleado_nombre character varying (100)	estado character varying (20)
1	1	Desarrollar módulo de autenticación	Crear el módulo de inicio de sesión para la aplicación.	2025-03-15	Juan Pérez	Pendiente



## 5. Sistema de Evaluación Académica

Identificar entidades clave:

Estudiante

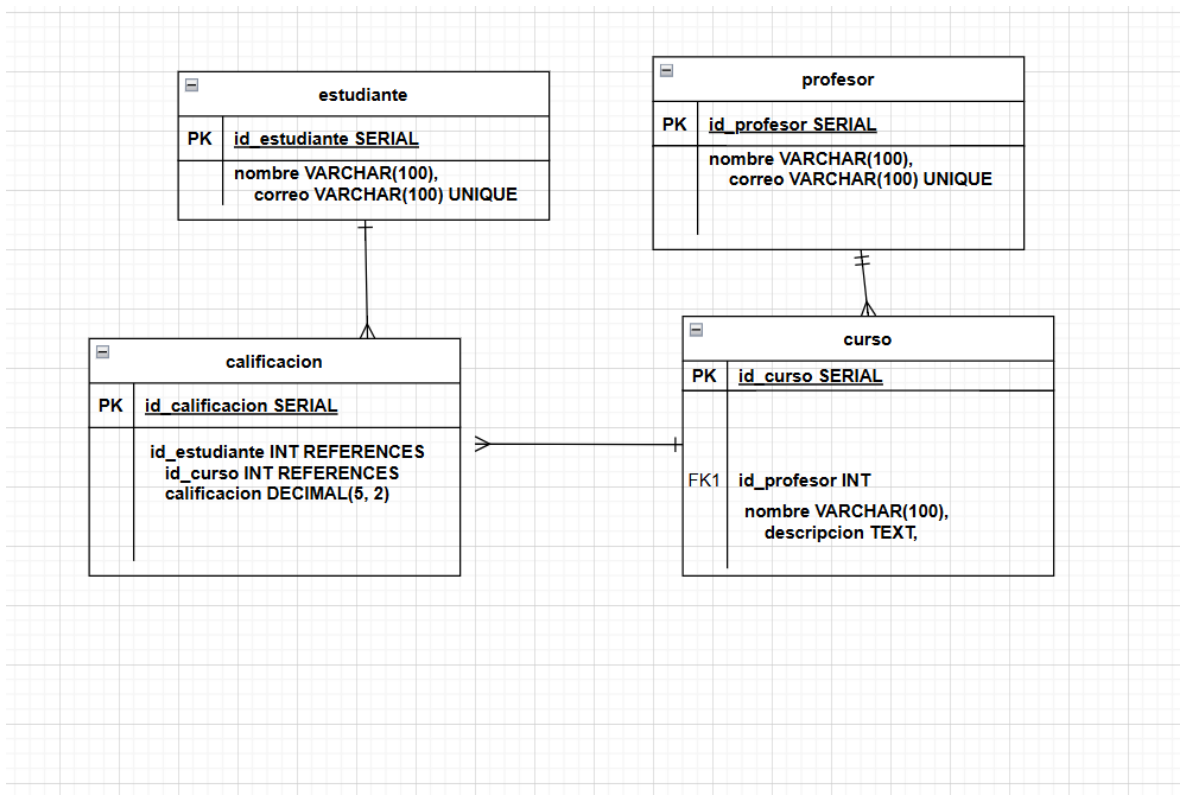
Curso

Profesor

Calificación.

### Relaciones principales:

- Profesor → Curso (Uno a Muchos)
- Curso → Calificación (Uno a Muchos)
- Estudiante → Calificación (Uno a Muchos)



```
CREATE TABLE estudiante (  
    id_estudiante SERIAL PRIMARY KEY,  
    nombre VARCHAR(100),  
    correo VARCHAR(100) UNIQUE  
);  
  
-- Tabla de Profesor  
CREATE TABLE profesor (  
    id_profesor SERIAL PRIMARY KEY,  
    nombre VARCHAR(100),  
    correo VARCHAR(100) UNIQUE  
);  
  
-- Tabla de Curso  
CREATE TABLE curso (  
    id_curso SERIAL PRIMARY KEY,  
    nombre VARCHAR(100),  
    descripcion TEXT,  
    id_profesor INT REFERENCES profesor(id_profesor)  
);  
  
-- Tabla de Calificación  
CREATE TABLE calificacion (  
    id_calificacion SERIAL PRIMARY KEY,  
    id_estudiante INT REFERENCES estudiante(id_estudiante),  
    id_curso INT REFERENCES curso(id_curso),  
    calificacion DECIMAL(5, 2)  
);
```

```
INSERT INTO estudiante (nombre, correo)  
VALUES  
    ('Juan Pérez', 'juan.perez@ejemplo.com'),  
    ('María López', 'maria.lopez@ejemplo.com'),  
    ('Carlos González', 'carlos.gonzalez@ejemplo.com');
```

```
INSERT INTO profesor (nombre, correo)  
VALUES  
    ('Prof. Ana Martínez', 'ana.martinez@ejemplo.com'),  
    ('Prof. Luis Rodríguez', 'luis.rodriguez@ejemplo.com');
```

```

INSERT INTO curso (nombre, descripcion, id_profesor)
VALUES
  ('Matemáticas 101', 'Curso de introducción a las matemáticas', 1),
  ('Historia 101', 'Curso introductorio de historia', 2),
  ('Física 101', 'Curso de introducción a la física', 1);

```

```

INSERT INTO calificacion (id_estudiante, id_curso, calificacion)
VALUES
  (1, 1, 85.50),
  (1, 2, 90.00),
  (1, 3, 88.00),
  (2, 1, 92.00),
  (2, 3, 85.50),
  (3, 2, 78.00);

```

- Consulta requerida: Obtener el promedio de calificaciones de un estudiante en todos sus cursos.

```

SELECT
  e.nombre AS estudiante_nombre,
  AVG(ca.calificacion) AS promedio_calificaciones
FROM
  calificacion ca
JOIN
  estudiante e ON ca.id_estudiante = e.id_estudiante
WHERE
  e.id_estudiante = 1
GROUP BY
  e.id_estudiante, e.nombre;

```

	estudiante_nombre character varying (100)	promedio_calificaciones numeric
1	Juan Pérez	87.83333333333333