

Sistema de Gestión de Inventarios

1. Análisis de Requisitos.
 - Definir Entidades Claves

Producto

Proveedor

Categoría

Inventario

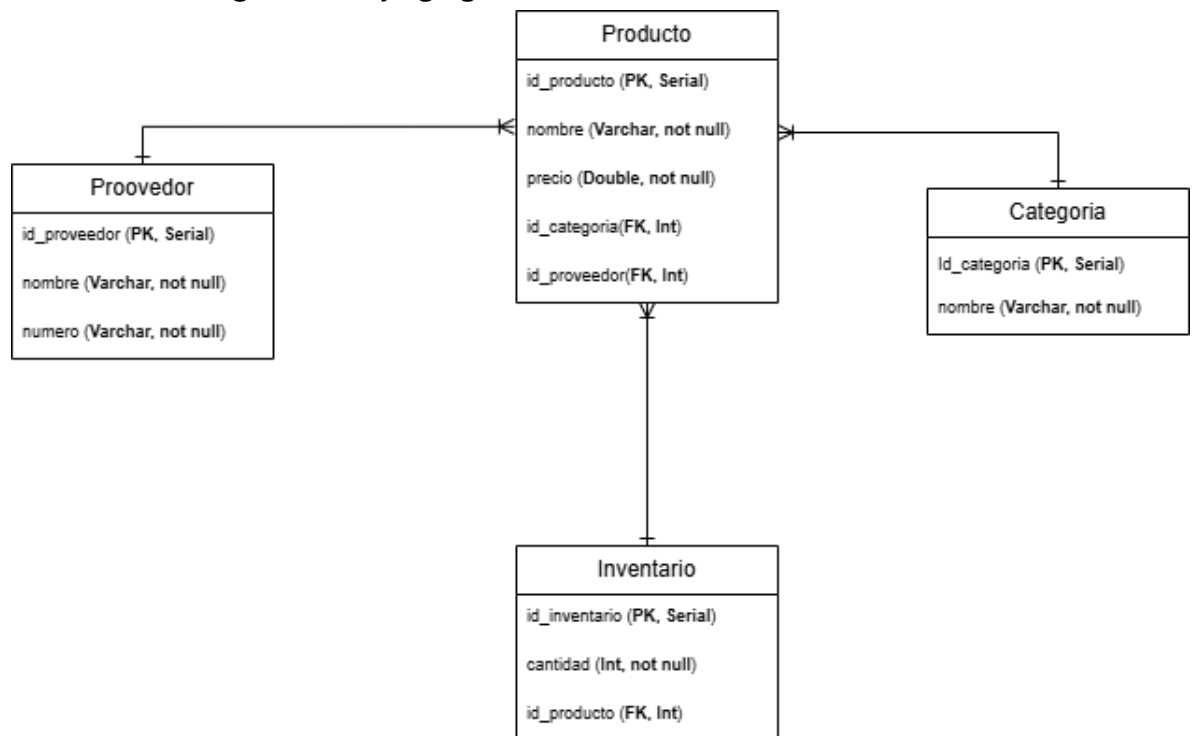
- Relaciones de entidades

Un **Producto** pertenece a una **Categoría**.

Un **Proveedor** tiene un **Producto**.

El **Inventario** mide la cantidad de **Productos** que hay en stock.

2. Creación de Diagrama E-R y agregación de atributos



3. Implementación en SQL y agregación de datos

```
--Creacion de Tablas  
  
CREATE TABLE Categoria (  
    id_categoria SERIAL PRIMARY KEY,
```

```
        nombre VARCHAR(100) NOT NULL
    );

CREATE TABLE Proveedor (
    id_proveedor SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    numero VARCHAR(20) NOT NULL
);

CREATE TABLE Producto (
    id_producto SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    precio DECIMAL(10, 2) NOT NULL,
    id_categoria INT NOT NULL,
    id_proveedor INT NOT NULL,
    FOREIGN KEY (id_categoria) REFERENCES Categoria(id_categoria),
    FOREIGN KEY (id_proveedor) REFERENCES Proveedor(id_proveedor)
);

CREATE TABLE Inventario (
    id_inventario SERIAL PRIMARY KEY,
    id_producto INT NOT NULL,
    cantidad INT NOT NULL,
    FOREIGN KEY (id_producto) REFERENCES Producto(id_producto)
);

--Insercion de Datos

-- Categorías
INSERT INTO Categoria (nombre) VALUES
('Electrónica'),
('Hogar'),
('Ropa'),
('Deportes'),
('Libros'),
('Juguetes'),
('Salud'),
('Oficina'),
('Automotriz'),
('Alimentos');

-- Proveedores
INSERT INTO Proveedor (nombre, numero) VALUES
('Tech Supplier', '123-456-7890'),
('Home Goods', '098-765-4321'),
('Fashion Hub', '222-333-4444'),
('Sports Direct', '555-666-7777'),
('Book World', '888-999-0000'),
('Toy Planet', '111-222-3333'),
('Health Plus', '444-555-6666'),
('Office Supply Co', '777-888-9999'),
('Auto Parts Inc', '999-000-1111'),
('Food Market', '666-777-8888');

-- Productos
INSERT INTO Producto (nombre, precio, id_categoria, id_proveedor)
VALUES
```

```

('Laptop', 1000.00, 1, 1),
('Mesa', 150.00, 2, 2),
('Camiseta', 25.00, 3, 3),
('Pelota de fútbol', 30.00, 4, 4),
('Libro de cocina', 20.00, 5, 7),
('Muñeca', 15.00, 6, 6),
('Vitaminas', 12.00, 7, 7),
('Cuaderno', 5.00, 8, 5),
('Filtro de aceite', 40.00, 9, 9),
('Chocolate', 8.00, 10, 10),
('Escritorio', 40.00, 8, 8);

-- Inventario
INSERT INTO Inventario (id_producto, cantidad) VALUES
(1, 50),
(2, 30),
(3, 100),
(4, 80),
(5, 60),
(6, 40),
(7, 90),
(8, 120),
(9, 25),
(10, 200),
(11, 30);

```

4. Consultas SQL y Validacion

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

SELECT p.nombre AS Producto, c.nombre AS Categoria, pr.nombre AS Proveedor

FROM Producto p

INNER JOIN Categoria c ON p.id_categoria = c.id_categoria

INNER JOIN Proveedor pr ON p.id_proveedor = pr.id_proveedor

ORDER BY p.nombre;

	producto ▾	:	categoria ▾	:	proveedor ▾	:
1	Camiseta		Ropa		Fashion Hub	
2	Chocolate		Alimentos		Food Market	
3	Cuaderno		Oficina		Book World	
4	Escritorio		Oficina		Office Supply Co	
5	Filtro de aceite		Automotriz		Auto Parts Inc	
6	Laptop		Electrónica		Tech Supplier	
7	Libro de cocina		Libros		Health Plus	
8	Mesa		Hogar		Home Goods	
9	Muñeca		Juguetes		Toy Planet	
10	Pelota de fútbol		Deportes		Sports Direct	
11	Vitaminas		Salud		Health Plus	

- Productos ordenados por cantidad en inventario (de mayor a menor)

```
SELECT p.nombre AS Producto, i.cantidad AS Cantidad
FROM Producto p
INNER JOIN Inventario i ON p.id_producto = i.id_producto
WHERE p.nombre LIKE '%o%'
ORDER BY i.cantidad DESC;
```

	producto	cantidad
1	Chocolate	200
2	Cuaderno	120
3	Pelota de fútbol	80
4	Libro de cocina	60
5	Laptop	50
6	Escritorio	30
7	Filtro de aceite	25

- Obtener el número total de productos por categoría

```
SELECT c.nombre AS Categoria, COUNT(p.id_producto) AS Total_Productos
FROM Categoria c
LEFT JOIN Producto p ON c.id_categoria = p.id_categoria
GROUP BY c.nombre
ORDER BY Total_Productos DESC;
```

	categoria	total_productos
1	Oficina	2
2	Hogar	1
3	Juguetes	1
4	Automotriz	1
5	Deportes	1
6	Alimentos	1
7	Libros	1
8	Ropa	1
9	Electrónica	1
10	Salud	1

Sistema de Gestión de Eventos

1. Análisis de Requisitos

- Definir entidades clave

Evento

Participante

Ubicación

Organizador

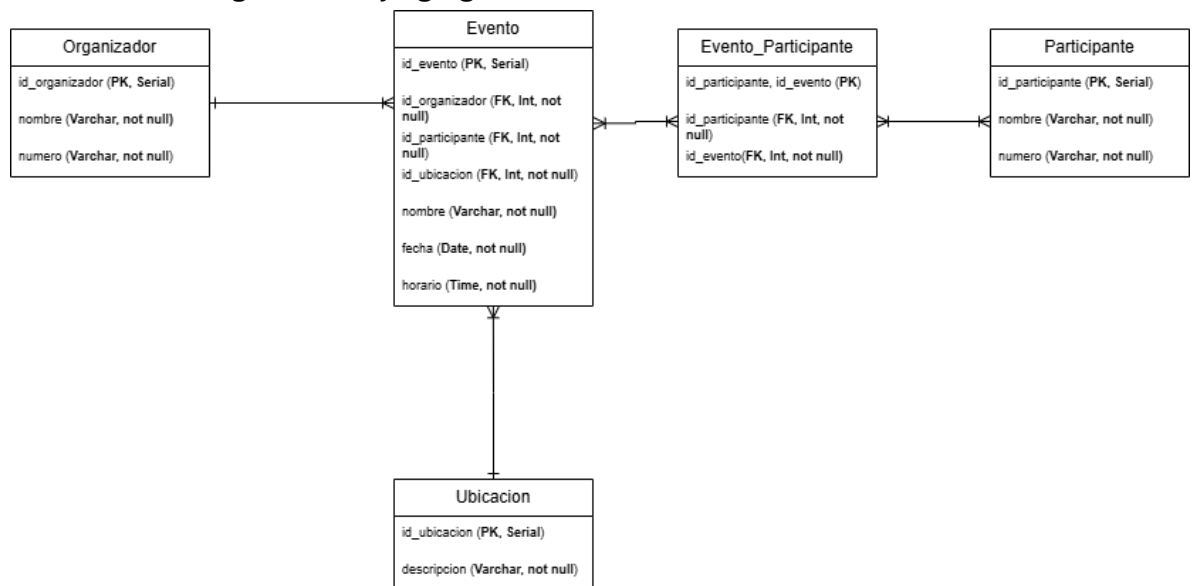
- Relación entre entidades

Un **Evento** tiene uno o varios **Participantes**.

Un **Evento** tiene una **Ubicación**.

Un **Organizador** tiene un **Evento**.

2. Creación de Diagrama E-R y agregación de atributos



3. Implementación SQL y agregación de datos

```

-- Creacion de tablas

CREATE TABLE Organizador (
  id_organizador SERIAL PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  numero VARCHAR(20) NOT NULL
);

CREATE TABLE Participante (
  id_participante SERIAL PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  numero VARCHAR(20) NOT NULL
);
  
```

```
nombre VARCHAR(100) NOT NULL,
numero VARCHAR(20) NOT NULL
);

CREATE TABLE Ubicacion (
id_ubicacion SERIAL PRIMARY KEY,
descripcion VARCHAR(200) NOT NULL
);

CREATE TABLE Evento (
id_evento SERIAL PRIMARY KEY,
id_organizador INT NOT NULL,
id_participante INT NOT NULL,
id_ubicacion INT NOT NULL,
nombre VARCHAR NOT NULL,
horario TIME NOT NULL,
fecha DATE NOT NULL,
FOREIGN KEY (id_organizador) REFERENCES Organizador(id_organizador),
FOREIGN KEY (id_participante) REFERENCES
Participante(id_participante),
FOREIGN KEY (id_ubicacion) REFERENCES Ubicacion(id_ubicacion)
);

CREATE TABLE Evento_Participante (
    id_evento INT NOT NULL,
    id_participante INT NOT NULL,
    PRIMARY KEY (id_evento, id_participante),
    FOREIGN KEY (id_evento) REFERENCES Evento(id_evento),
    FOREIGN KEY (id_participante) REFERENCES
Participante(id_participante)
);

--Insercion de Datos

INSERT INTO Organizador (nombre, numero) VALUES
('Carlos Pérez', '5512345678'),
('Ana López', '5523456789'),
('Luis Gómez', '5534567890'),
('Marta Rodríguez', '5545678901'),
('Javier Martínez', '5556789012'),
('Claudia Díaz', '5567890123'),
('Pedro Sánchez', '5578901234'),
('Laura García', '5589012345'),
('Fernando Torres', '5590123456'),
('Patricia Jiménez', '5601234567');

INSERT INTO Participante (nombre, numero) VALUES
('Juan Morales', '5512345678'),
('María Fernández', '5523456789'),
('Pedro Ramírez', '5534567890'),
('Elena Ruiz', '5545678901'),
('Ricardo Pérez', '5556789012'),
('Ana Sánchez', '5567890123'),
('José Díaz', '5578901234'),
('Clara López', '5589012345'),
('Miguel Torres', '5590123456'),
```

```
('Sofía García', '5601234567');

INSERT INTO Ubicacion (descripcion) VALUES
('Auditorio Nacional, Ciudad de México'),
('Palacio de los Deportes, Ciudad de México'),
('Teatro Diana, Guadalajara'),
('Arena Monterrey, Monterrey'),
('Cinepolis, Cancún'),
('Auditorio Telmex, Guadalajara'),
('Plaza de Toros, León'),
('Centro Banamex, Ciudad de México'),
('Explanada de los Insurgentes, Ciudad de México'),
('Foro Sol, Ciudad de México');

INSERT INTO Evento (id_organizador, id_ubicacion, id_participante,
nombre, horario, fecha) VALUES
(1, 1, 1, 'Evento A', '10:00:00', '2025-04-01'),
(1, 2, 2, 'Evento B', '12:00:00', '2025-04-02'),
(2, 3, 3, 'Evento C', '14:00:00', '2025-04-03'),
(3, 4, 4, 'Evento D', '16:00:00', '2025-04-04'),
(4, 5, 5, 'Evento E', '18:00:00', '2025-04-05'),
(5, 6, 6, 'Evento F', '20:00:00', '2025-04-06'),
(6, 7, 7, 'Evento G', '08:00:00', '2025-04-07'),
(7, 8, 8, 'Evento H', '10:00:00', '2025-04-08'),
(8, 9, 9, 'Evento I', '12:00:00', '2025-04-09'),
(9, 10, 1, 'Evento J', '14:00:00', '2025-04-10');

INSERT INTO Evento_Participante (id_evento, id_participante) VALUES
(1, 1),
(1, 2),
(1, 3),
(2, 4),
(2, 5),
(2, 6),
(3, 7),
(3, 8),
(4, 9),
(5, 1),
(5, 10),
(6, 2),
(6, 3),
(6, 4),
(7, 5),
(7, 6),
(8, 7),
(8, 8),
(9, 9),
(9, 1);
```

4. Consultas SQL y Validación

- Obtener la lista de eventos programados junto con la cantidad de participantes registrados por evento.

```
SELECT e.id_evento, e.nombre AS nombre_evento, e.fecha, e.horario,
COUNT(ep.id_participante) AS cantidad_participantes
FROM Evento e
LEFT JOIN Evento_Participante ep ON e.id_evento = ep.id_evento
GROUP BY e.id_evento, e.nombre, e.fecha, e.horario
ORDER BY e.fecha, e.horario;
```

	id_evento	nombre_evento	fecha	horario	cantidad_participantes
1	1	Evento A	2025-04-01	10:00:00	3
2	2	Evento B	2025-04-02	12:00:00	3
3	3	Evento C	2025-04-03	14:00:00	2
4	4	Evento D	2025-04-04	16:00:00	1
5	5	Evento E	2025-04-05	18:00:00	2
6	6	Evento F	2025-04-06	20:00:00	3
7	7	Evento G	2025-04-07	08:00:00	2
8	8	Evento H	2025-04-08	10:00:00	2
9	9	Evento I	2025-04-09	12:00:00	2
10	10	Evento J	2025-04-10	14:00:00	0

- Buscar Organizadores cuyo nombre empiece con la letra “c” y sus eventos correspondientes

```
SELECT o.nombre AS organizador, o.numero, e.id_evento, e.nombre AS
nombre_evento, e.fecha, e.horario
FROM Organizador o
INNER JOIN Evento e ON o.id_organizador = e.id_organizador
WHERE o.nombre LIKE 'C%'
ORDER BY o.nombre, e.fecha;
```

	organizador	numero	id_evento	nombre_evento	fecha	horario
1	Carlos Pérez	5512345678	1	Evento A	2025-04-01	10:00:00
2	Carlos Pérez	5512345678	2	Evento B	2025-04-02	12:00:00
3	Claudia Diaz	5567890123	7	Evento G	2025-04-07	08:00:00

- Obtener los eventos programados para un día específico

```
SELECT e.id_evento, e.nombre AS nombre_evento, e.fecha, e.horario, u.descripcion
AS ubicacion, o.nombre AS organizador
```


FROM Evento e

LEFT JOIN Ubicacion u ON e.id_ubicacion = u.id_ubicacion

LEFT JOIN Organizador o ON e.id_organizador = o.id_organizador

WHERE e.fecha = '2025-04-01'

ORDER BY e.horario;

	id_evento ▾	nombre_evento ▾	fecha ▾	horario ▾	ubicacion ▾	organizador ▾
1	1	Evento A	2025-04-01	10:00:00	Auditorio Nacional, Ciudad de México	Carlos Pérez

Plataforma de Streaming de Musica

1. Analisis de Datos

- Identificar Entidades Claves

Artista

Usuario

Cancion

Album

Reproduccion (Nueva tabla)

- Relacion de Entidades

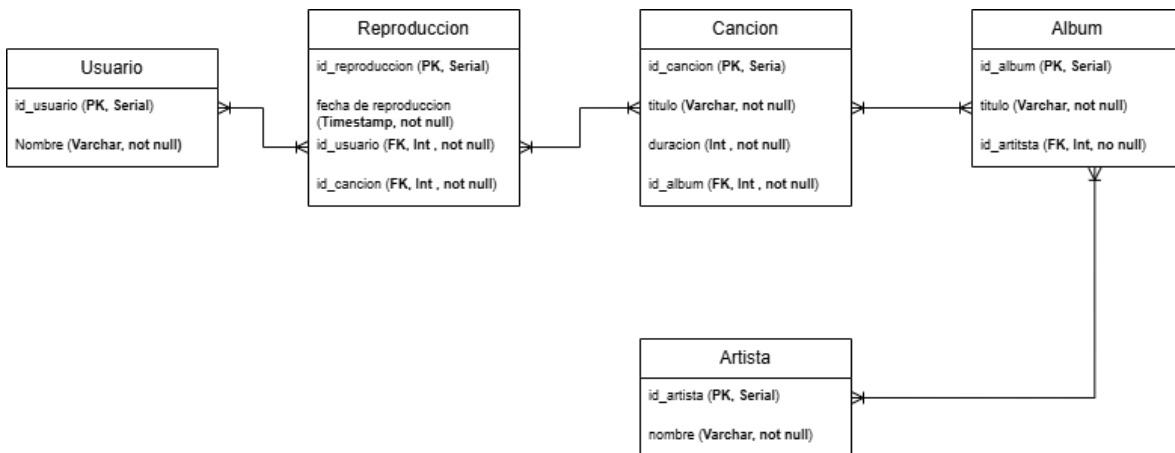
Artista tiene una o varias **Canciones**.

Artista tiene uno o varios **Album**.

Album tiene uno o varias **Canciones**.

Usuario puede **Reproducir** una o varias **Canciones**.

2. Diagrama E-R y Asignación de Atributos



3. Implementación SQL y agregación de datos

```

--Creacion de tablas

CREATE TABLE Usuario (
    id_usuario SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL
);

CREATE TABLE Artista (
    id_artista SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL
);

CREATE TABLE Album (
    id_album SERIAL PRIMARY KEY,
    titulo VARCHAR(100) NOT NULL,
    id_artista INT NOT NULL,
    FOREIGN KEY (id_artista) REFERENCES Artista(id_artista)
);

CREATE TABLE Cancion (
    id_cancion SERIAL PRIMARY KEY,
    titulo VARCHAR(100) NOT NULL,
    duracion INT NOT NULL, -- en segundos
    id_album INT NOT NULL,
    FOREIGN KEY (id_album) REFERENCES Album(id_album)
);

CREATE TABLE Reproduccion (
    id_reproduccion SERIAL PRIMARY KEY,
    id_usuario INT NOT NULL,
    id_cancion INT NOT NULL,
    fecha_reproduccion TIMESTAMP NOT NULL,
    FOREIGN KEY (id_usuario) REFERENCES Usuario(id_usuario),
    FOREIGN KEY (id_cancion) REFERENCES Cancion(id_cancion)
);
  
```

```
);

--Agregacion de Datos

-- Usuarios
INSERT INTO Usuario (nombre) VALUES
('Juan'),
('María'),
('Carlos'),
('Ana'),
('Luis');

-- Artistas
INSERT INTO Artista (nombre) VALUES
('Coldplay'),
('Taylor Swift'),
('Ed Sheeran'),
('Shakira'),
('The Weeknd');

-- Álbumes
INSERT INTO Album (titulo, id_artista) VALUES
('Parachutes', 1),
('1989', 2),
('Divide', 3),
('El Dorado', 4),
('Starboy', 5);

-- Canciones
INSERT INTO Cancion (titulo, duracion, id_album) VALUES
('Yellow', 266, 1),
('Shake It Off', 242, 2),
('Perfect', 263, 3),
('Chantaje', 219, 4),
('Blinding Lights', 200, 5);

-- Reproducciones
INSERT INTO Reproduccion (id_usuario, id_cancion, fecha_reproduccion)
VALUES
(1, 1, '2025-03-01 14:30:00'),
(2, 2, '2025-03-02 15:00:00'),
(3, 3, '2025-03-03 16:00:00'),
(4, 4, '2025-03-04 17:00:00'),
(5, 5, '2025-03-05 18:00:00');
```

4. Consultas SQL y Validación

- Listar las canciones reproducidas por un usuario específico, incluyendo el nombre del artista y del álbum

```
SELECT u.nombre_usuario AS Usuario, c.titulo AS Cancion, a.titulo AS Album,
ar.nombre AS Artista
```

```
FROM Reproduccion r
```

```
INNER JOIN Usuario u ON r.id_usuario = u.id_usuario
```

```
INNER JOIN Cancion c ON r.id_cancion = c.id_cancion
```

```
INNER JOIN Album a ON c.id_album = a.id_album
```

```
INNER JOIN Artista ar ON a.id_artista = ar.id_artista
```

```
WHERE u.nombre_usuario = 'Juan';
```

	usuario ▾	cancion ▾	album ▾	artista ▾
1	Juan	Yellow	Parachutes	Coldplay

- Mostrar todas las canciones de un álbum específico

```
SELECT c.titulo AS Cancion, c.duracion AS Duracion, ar.nombre AS Artista
```

```
FROM Cancion c
```

```
INNER JOIN Album a ON c.id_album = a.id_album
```

```
INNER JOIN Artista ar ON a.id_artista = ar.id_artista
```

```
WHERE a.titulo LIKE '%NFDC%';
```

	cancion ▾	duracion ▾	artista ▾
1	Tintado	220	Yng Naz
2	Viaja Conmigo	201	Yng Naz

- Mostrar los usuarios que reprodujeron canciones de un artista específico

```
SELECT u.nombre_usuario AS Usuario, c.titulo AS Cancion, ar.nombre AS Artista
```

```
FROM Usuario u
```

```
LEFT JOIN Reproduccion r ON u.id_usuario = r.id_usuario
```

```
LEFT JOIN Cancion c ON r.id_cancion = c.id_cancion
```

```
LEFT JOIN Album a ON c.id_album = a.id_album
```

```
LEFT JOIN Artista ar ON a.id_artista = ar.id_artista
```

```
WHERE ar.nombre = 'Yng Naz';
```

	usuario ▼	cancion ▼	artista ▼
1	María	Viaja Conmigo	Yng Naz

Sistema de Gestión de Proyectos

1. Análisis de Requisitos

- Definir Entidades Claves

Empleado

Tarea

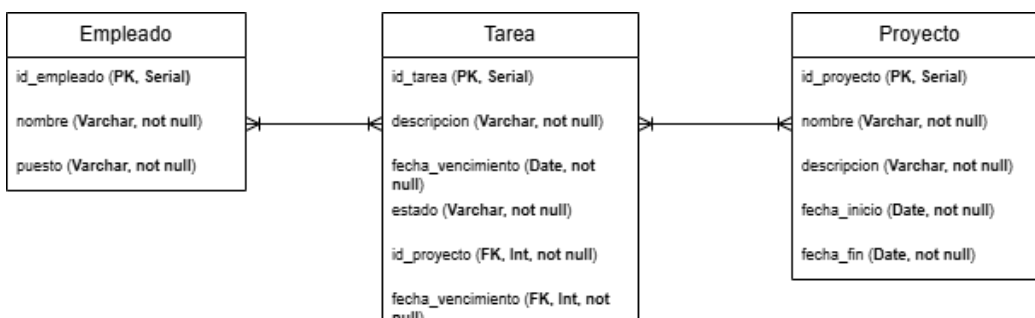
Proyecto

- Relaciones de entidades

Un **proyecto** tiene muchas **tareas**.

Un **empleado** puede estar asignado a muchas **tareas**.

2. Diseño de diagrama E-R y asignación de atributos



3. Implemetacion SQL y inserción de Datos

```
-- Creación de tablas
CREATE TABLE Proyecto (
    id_proyecto SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    descripcion VARCHAR(255) NOT NULL,
    fecha_inicio DATE NOT NULL,
    fecha_fin DATE NOT NULL
);

CREATE TABLE Empleado (
    id_empleado SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    puesto VARCHAR(50) NOT NULL
);

CREATE TABLE Tarea (
    id_tarea SERIAL PRIMARY KEY,
    id_proyecto INT NOT NULL,
    id_empleado INT NOT NULL,
    descripcion VARCHAR(255) NOT NULL,
    fecha_vencimiento DATE NOT NULL,
    estado VARCHAR(20) NOT NULL,
    FOREIGN KEY (id_proyecto) REFERENCES Proyecto(id_proyecto),
    FOREIGN KEY (id_empleado) REFERENCES Empleado(id_empleado)
);

-- Insercion de datos

INSERT INTO Proyecto (nombre, descripcion, fecha_inicio, fecha_fin)
VALUES
('Desarrollo Web', 'Crear un sitio web para un cliente', '2024-03-01',
'2024-06-30'),
('Aplicación Móvil', 'Desarrollo de una app para Android', '2024-04-
15', '2024-09-15'),
('Sistema ERP', 'Implementación de un ERP para la empresa', '2024-05-
01', '2024-12-01'),
('Rediseño UX', 'Mejorar la experiencia de usuario en una plataforma',
'2024-02-10', '2024-07-20');

INSERT INTO Empleado (nombre, puesto) VALUES
('Carlos Pérez', 'Desarrollador'),
('Ana Gómez', 'Diseñadora'),
('Luis Torres', 'Gerente de Proyecto'),
('María López', 'Analista de Negocio'),
('Jorge Ramírez', 'Tester');

INSERT INTO Tarea (id_proyecto, id_empleado, descripcion,
fecha_vencimiento, estado) VALUES
(1, 1, 'Diseñar la estructura de la base de datos', '2024-03-10',
'Pendiente'),
(1, 2, 'Crear el diseño de la interfaz', '2024-03-15', 'Completada'),
(2, 3, 'Definir los requisitos del cliente', '2024-04-20',
'Pendiente'),
(3, 4, 'Analizar procesos de la empresa', '2024-05-15', 'Pendiente'),
```

```
(3, 1, 'Desarrollar módulo de facturación', '2024-06-10', 'Completada'),  
(4, 2, 'Prototipar nuevas pantallas', '2024-03-20', 'En progreso'),  
(4, 5, 'Realizar pruebas de usabilidad', '2024-06-01', 'En progreso');
```

4. Consultas SQL y Validación

- Mostrar todas las tareas pendientes de un proyecto específico, ordenadas por fecha de vencimiento

```
SELECT * FROM Tarea
```

```
WHERE id_proyecto = 1 AND estado = 'Pendiente'
```

```
ORDER BY fecha_vencimiento;
```

id_tarea	id_proyecto	id_empleado	descripcion	fecha_vencimiento	estado
1	1	1	1 Diseñar la estructura de la base de datos	2024-03-10	Pendiente

- Obtener la lista de tareas junto con el nombre del empleado asignado y el proyecto

```
SELECT T.id_tarea, T.descripcion, T.fecha_vencimiento, T.estado, E.nombre AS  
empleado, P.nombre AS proyecto
```

```
FROM Tarea T
```

```
INNER JOIN Empleado E ON T.id_empleado = E.id_empleado
```

```
INNER JOIN Proyecto P ON T.id_proyecto = P.id_proyecto;
```

id_tarea	descripcion	fecha_vencimiento	estado	empleado	proyecto
1	1 Diseñar la estructura de la base de datos	2024-03-10	Pendiente	Carlos Pérez	Desarrollo Web
2	2 Crear el diseño de la interfaz	2024-03-15	Completada	Ana Gómez	Desarrollo Web
3	3 Definir los requisitos del cliente	2024-04-20	Pendiente	Luis Torres	Aplicación Móvil
4	4 Analizar procesos de la empresa	2024-05-15	Pendiente	María López	Sistema ERP
5	5 Desarrollar módulo de facturación	2024-06-10	Completada	Carlos Pérez	Sistema ERP
6	6 Prototipar nuevas pantallas	2024-03-20	En progreso	Ana Gómez	Rediseño UX
7	7 Realizar pruebas de usabilidad	2024-06-01	En progreso	Jorge Ramírez	Rediseño UX

- Buscar tareas cuya descripcion contenga la palabra "d"

```
SELECT * FROM Tarea
```

```
WHERE descripcion
```

```
LIKE '%d%';
```

id_tarea	id_proyecto	id_empleado	descripcion	fecha_vencimiento	estado
1	1	1	1 Diseñar la estructura de la base de datos	2024-03-10	Pendiente
2	2	1	2 Crear el diseño de la interfaz	2024-03-15	Completada
3	3	2	3 Definir los requisitos del cliente	2024-04-20	Pendiente
4	4	3	4 Analizar procesos de la empresa	2024-05-15	Pendiente
5	5	3	1 Desarrollar módulo de facturación	2024-06-10	Completada
7	7	4	5 Realizar pruebas de usabilidad	2024-06-01	En progreso

Sistema de Evaluación Académica

1. Analisis de Requisitos
 - Definir Entidades Claves

Estudiante

Curso

Profesor

Calificación

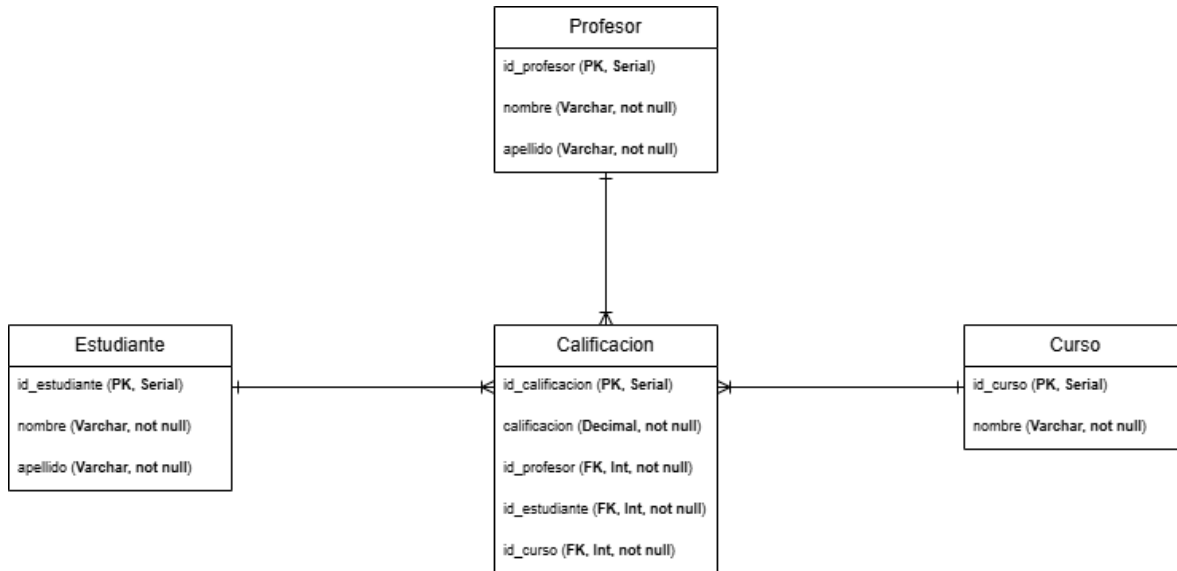
- Relacion entre entidades

Estudiante tiene una o varias **Calificacion**

Profesor asigna una o varias **Calificaciones**

Curso con tiene una o varias **Calificiaciones**

2. Diseño de diagrama E-R y asignación de atributos



3. Implementación en SQL y asignación de datos

```

--Creacion de tablas
CREATE TABLE Estudiante (
    id_estudiante SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    apellido VARCHAR(100) NOT NULL
);

CREATE TABLE Curso (
    id_curso SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL
);

CREATE TABLE Profesor (
    id_profesor SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    apellido VARCHAR(100) NOT NULL
);

CREATE TABLE Calificacion (
    id_calificacion SERIAL PRIMARY KEY,
    id_estudiante INT REFERENCES Estudiante(id_estudiante),
    id_curso INT REFERENCES Curso(id_curso),
    id_profesor INT REFERENCES Profesor(id_profesor),
    calificacion DECIMAL(5,2)
);

--Insercion de Datos
INSERT INTO Estudiante (nombre, apellido) VALUES
('Juan', 'Pérez'),
('María', 'Gómez'),
('Carlos', 'Ramírez'),
  
```

```
('Ana', 'López'),
('Luis', 'Martínez'),
('Sofía', 'Hernández'),
('Pedro', 'García'),
('Elena', 'Torres'),
('Miguel', 'Fernández'),
('Laura', 'Díaz');

INSERT INTO Curso (nombre) VALUES
('Matemáticas'),
('Historia'),
('Física'),
('Química');

INSERT INTO Profesor (nombre, apellido) VALUES
('Carlos', 'Fernández'),
('Ana', 'López'),
('Luis', 'Martínez'),
('Elena', 'García');

INSERT INTO Calificacion (id_estudiante, id_curso, id_profesor,
calificacion) VALUES
(1, 1, 1, 85.5),
(1, 2, 2, 90.0),
(2, 1, 1, 78.0),
(2, 3, 3, 88.5),
(3, 4, 4, 92.0),
(4, 2, 2, 76.5),
(5, 3, 3, 81.0),
(6, 1, 1, 87.5),
(7, 2, 2, 79.0),
(8, 4, 4, 85.0),
(9, 3, 3, 91.5),
(10, 1, 1, 83.0);
```

4. Consultas SQL y validación

- Promedio de calificaciones de un estudiante en todos sus cursos

```
SELECT e.nombre AS Estudiante, e.apellido AS Apellido, c.nombre AS Curso,
cal.calificacion AS Calificacion, AVG(cal.calificacion) AS Promedio_Estudiante
```

```
FROM Calificacion cal
```

```
INNER JOIN Estudiante e ON cal.id_estudiante = e.id_estudiante
```

```
INNER JOIN Curso c ON cal.id_curso = c.id_curso
```

```
WHERE e.id_estudiante = 1
```

```
GROUP BY e.nombre, e.apellido, c.nombre, cal.calificacion
```

```
ORDER BY c.nombre;
```

	estudiante	apellido	curso	calificacion	promedio_estudiante
1	Juan	Pérez	Historia	90.00	90
2	Juan	Pérez	Matemáticas	85.50	85.5

- Obtener el promedio de calificaciones por curso

```
SELECT c.nombre AS Curso, AVG(cal.calificacion) AS Promedio_Curso
```

```
FROM Calificacion cal
```

```
LEFT JOIN Curso c ON cal.id_curso = c.id_curso
```





```
GROUP BY c.id_curso;
```

	curso	promedio_curso
1	Química	88.5
2	Historia	81.83333333333333
3	Matemáticas	83.5
4	Física	87

- Buscar cursos que contengan la palabra 'a'

```
SELECT * FROM Curso
```

```
WHERE nombre LIKE '%a%';
```

	 id_curso 	 nombre 	
1		1 Matemáticas	
2		2 Historia	
3		3 Física	
4		4 Química	