INSTITUTO TECNOLOGICO DE MEXICALI

FUNDAMENTOS DE BASE DE DATOS

TAREA 3

MAESTRO: JOSE RAMON BOGARIN VALENZUELA

ALUMNO: YASSER ABRAHAM IRACHETA MEDRANO

NUMERO DE CONTROL: 23490394

MEXICALI BAJA CALIFORNIA AL 23 DE MARZO DEL 2025

1. GESTION DE INVENTARIOS

1.Identificación de Entidades y Relaciones

Las entidades clave son:

- **Producto** (Tiene un nombre, precio, cantidad en inventario, etc.)
- **Proveedor** (Nombre, contacto, dirección, etc.)
- **Categoría** (Tipo de producto, clasificación)
- **Inventario** (Registra la cantidad de productos disponibles)

Las relaciones principales son:

- Un **producto** pertenece a una **categoría**.
- Un **producto** es suministrado por uno o varios **proveedores**.
- Un **proveedor** puede suministrar varios **productos**.

2. Modelo Entidad-Relación (E-R)

Representación gráfica donde:

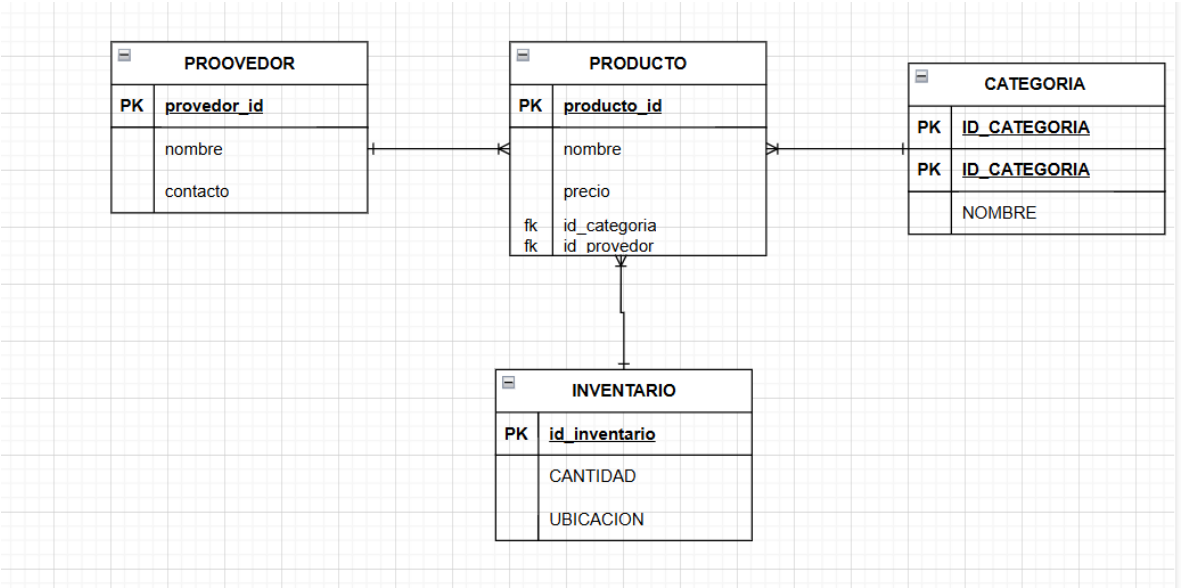
- **Producto** se relaciona con **Categoría** (uno a muchos).
- **Producto** se relaciona con **Proveedor** mediante una tabla intermedia (muchos a muchos).
- **Inventario** mantiene el stock de cada producto.

```
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,
  contacto VARCHAR(100) 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_producto INT PRIMARY KEY,
  cantidad INT NOT NULL,
  ubicacion VARCHAR(100) NOT NULL,
  FOREIGN KEY (id_producto) REFERENCES Producto(id_producto)
);
```

Consulta requerida: 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
JOIN Categoria C ON P.id_categoria = C.id_categoria
JOIN Proveedor PR ON P.id_proveedor = PR.id_proveedor
ORDER BY P.nombre;
```



	producto	categoria	proveedor
1	Balón de Fútbol	Deportes	DeportesMax
2	Batería de Auto	Automotriz	AutoParts
3	Camiseta Adidas	Ropa	ModaExpress
4	Laptop HP	Electrónica	TechSupplier
5	Leche Entera	Alimentos	SuperAlimentos
6	Libro "Cien Años de Soledad"	Libros	LibreríaCentral
7	Muñeca Barbie	Juguetes	JugueteFeliz
8	Perfume Chanel	Cosméticos	BellezaPlus
9	Sofá de Cuero	Muebles	MueblesFinos
10	Taladro Bosch	Herramientas	FerreteríaTotal

2. Sistema de Gestión de Eventos

1. Identificación de Entidades y Relaciones

Las entidades clave son:

- Evento (Nombre, fecha, ubicación, organizador)
- Participante (Nombre, email)
- Ubicación (Nombre, dirección)
- Organizador (Nombre, contacto)

Relaciones principales:

- Un evento ocurre en una ubicación (uno a muchos).

- Un evento es gestionado por un organizador (uno a muchos).
- Un evento tiene múltiples participantes, y un participante puede asistir a varios eventos (muchos a muchos, mediante la tabla Registro).

2. Modelo Entidad-Relación (E-R)

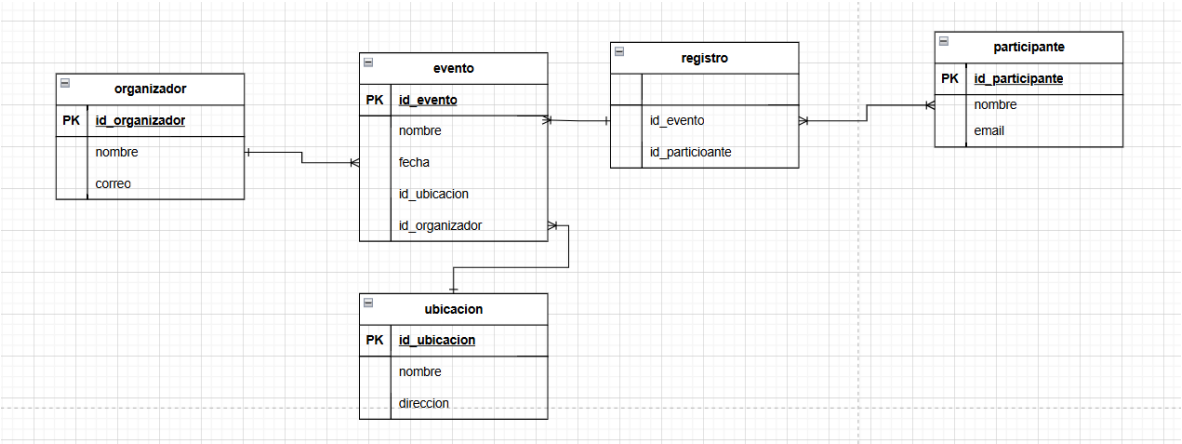
En el modelo E-R:

- Evento se relaciona con Ubicación (uno a muchos).
- Evento se relaciona con Organizador (uno a muchos).
- Evento se relaciona con Participante mediante una tabla intermedia Registro (muchos a muchos).

```
CREATE TABLE Ubicacion (
    id_ubicacion SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    direccion VARCHAR(255) NOT NULL
);

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

CREATE TABLE Evento (
    id_evento SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    fecha DATE NOT NULL,
    id_ubicacion INT NOT NULL,
    id_organizador INT NOT NULL,
    FOREIGN KEY (id_ubicacion) REFERENCES Ubicacion(id_ubicacion),
    FOREIGN KEY (id_organizador) REFERENCES
Organizador(id_organizador)
);

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

CREATE TABLE Registro (
    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)
);
```

- **Consulta requerida:** Obtener la lista de eventos programados junto con la cantidad de participantes registrados por evento
-
```sql
SELECT
    E.nombre AS evento,
    E.fecha,
    COUNT(R.id_participante) AS cantidad_participantes
FROM Evento E
LEFT JOIN Registro R ON E.id_evento = R.id_evento
GROUP BY E.id_evento, E.nombre, E.fecha
ORDER BY E.fecha;
```

	evento	fecha	cantidad_participantes
1	Conferencia de Tecnología	2025-05-10	2
2	Concierto Sinfónico	2025-06-15	2
3	Foro de Negocios	2025-07-20	2
4	Torneo de Fútbol	2025-08-25	2
5	Congreso de Medicina	2025-09-30	2
6	Festival de Teatro	2025-10-05	0
7	Cumbre de Innovación	2025-11-12	0
8	Seminario Universitario	2025-12-01	0
9	Gala Benéfica	2025-12-15	0
10	Feria del Libro	2026-01-20	0



3. Plataforma de Streaming de Música

Identificación de Entidades y Relaciones

Las entidades clave son:

- **Usuario** (Registra a los oyentes, con su nombre y correo electrónico).
- **Artista** (Registra a los artistas que suben música a la plataforma).
- **Álbum** (Cada álbum pertenece a un artista y contiene varias canciones).
- **Canción** (Cada canción pertenece a un álbum y es reproducida por los usuarios).
- **Reproducción** (Registra qué canciones han sido escuchadas por cada usuario y en qué momento).

Las relaciones principales son:

- Un **artista** puede tener **varios álbumes** (uno a muchos).
- Un **álbum** puede contener **varias canciones** (uno a muchos).
- Un **usuario** puede **reproducir varias canciones** y una canción puede ser reproducida por varios usuarios (**muchos a muchos** a través de la tabla **Reproducción**).

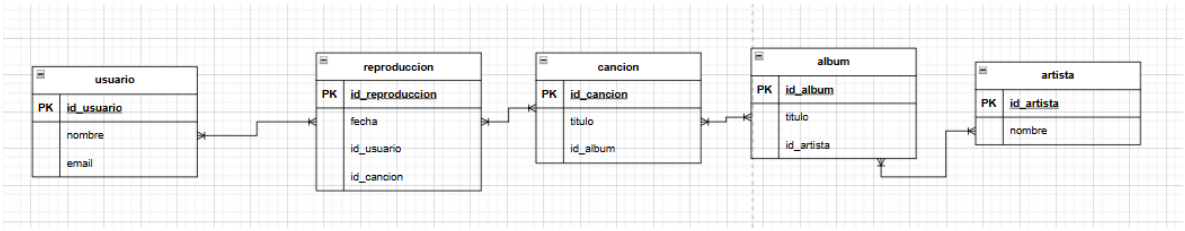
```
CREATE TABLE Usuario (
  id_usuario SERIAL PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  email VARCHAR(100) NOT NULL
);

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,
  id_album INT NOT NULL,
  FOREIGN KEY (id_album) REFERENCES Album(id_album)
);

CREATE TABLE Reproduccion (
  id_usuario INT NOT NULL,
  id_cancion INT NOT NULL,
  fecha TIMESTAMP DEFAULT NOW(),
  PRIMARY KEY (id_usuario, id_cancion, fecha),
  FOREIGN KEY (id_usuario) REFERENCES Usuario(id_usuario),
  FOREIGN KEY (id_cancion) REFERENCES Cancion(id_cancion)
);
```

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

```
SELECT
  U.nombre AS Usuario,
  C.titulo AS Cancion,
  A.titulo AS Album,
  AR.nombre AS Artista
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 = 1
ORDER BY R.fecha DESC;
```

	usuario	cancion	album	artista
1	Juan Pérez	Come Together	Abbey Road	The Beatles



4.Sistema de Control de Proyectos

1. Identificación de Entidades y Relaciones

Entidades clave:

- **Proyecto**: Cada proyecto tiene un nombre, fecha de inicio, fecha de finalización, y un presupuesto.
- **Empleado**: Cada empleado tiene un nombre, correo electrónico, y un cargo.
- **Tarea**: Cada tarea tiene un título, una descripción, una fecha de vencimiento y un estado (pendiente, en progreso, completada).

Relaciones principales:

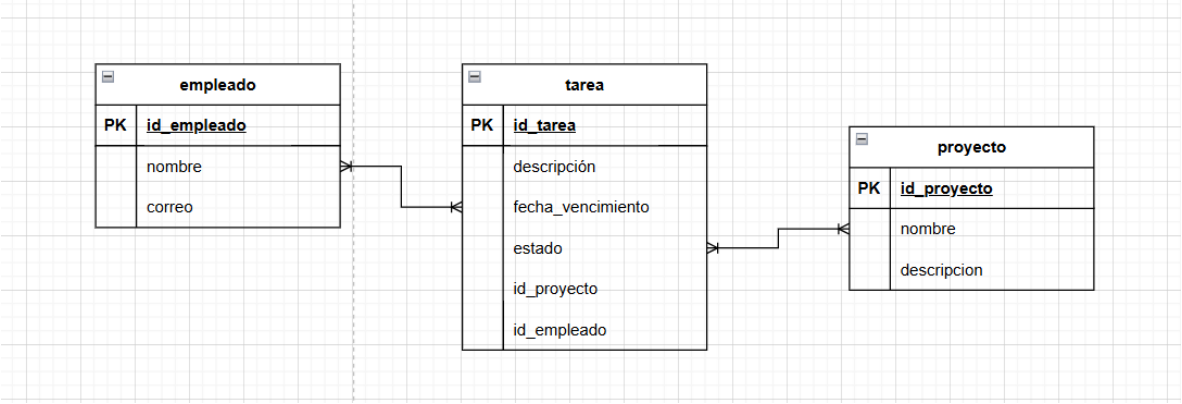
- Un proyecto tiene muchas tareas (uno a muchos).
- Un empleado puede estar asignado a muchas tareas (uno a muchos).
- Una tarea puede tener un solo empleado asignado, o varios empleados si es necesario.

```
CREATE TABLE Proyecto (
    id_proyecto SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    descripcion TEXT
);

CREATE TABLE Empleado (
    id_empleado SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    correo VARCHAR(100) UNIQUE NOT NULL
);
```

```
CREATE TABLE Tarea (
    id_tarea SERIAL PRIMARY KEY,
    descripcion TEXT NOT NULL,
    fecha_vencimiento DATE NOT NULL,
    estado VARCHAR(10) CHECK (estado IN ('pendiente',
'completada')) DEFAULT 'pendiente',
    id_proyecto INT REFERENCES Proyecto(id_proyecto) ON DELETE
CASCADE,
    id_empleado INT REFERENCES Empleado(id_empleado) ON DELETE SET
NULL
);
```

- **Consulta requerida:** Mostrar todas las tareas pendientes de un proyecto específico, ordenadas por fecha de vencimiento
-
```
SELECT P.nombre AS Proyecto, T.descripcion, T.fecha_vencimiento
FROM Tarea T
JOIN Proyecto P ON T.id_proyecto = P.id_proyecto
WHERE T.id_proyecto = 1 AND T.estado = 'pendiente'
ORDER BY T.fecha_vencimiento;
```

	proyecto	descripcion	fecha_vencimiento
1	Sistema de Inventarios	Diseñar la base de datos	2025-04-10



5. Sistema de Evaluación Académica

Identificación de Entidades y Relaciones

Las entidades clave son:

- **Estudiante** (nombre, email, matrícula, etc.)
- **Profesor** (nombre, email, especialidad, etc.)
- **Curso** (nombre, código, profesor asignado, etc.)
- **Calificación** (nota obtenida por un estudiante en un curso)

Las relaciones principales:

- Un **estudiante** puede estar inscrito en varios **cursos**.
- Un **curso** es impartido por un único **profesor**.
- Un **estudiante** tiene una o varias **calificaciones** en un curso.

```
CREATE TABLE Estudiante (
    id_estudiante SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE
);

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

CREATE TABLE Curso (
    id_curso SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    codigo VARCHAR(10) NOT NULL UNIQUE,
    id_profesor INT NOT NULL,
    FOREIGN KEY (id_profesor) REFERENCES Profesor(id_profesor)
);

CREATE TABLE Calificacion (
    id_estudiante INT NOT NULL,
    id_curso INT NOT NULL,
    calificacion DECIMAL(5,2) NOT NULL,
    PRIMARY KEY (id_estudiante, id_curso),
```

```
    FOREIGN KEY (id_estudiante) REFERENCES
Estudiante(id_estudiante),
    FOREIGN KEY (id_curso) REFERENCES Curso(id_curso)
);
```

- **Consulta requerida:** Obtener el promedio de calificaciones de un estudiante en todos sus cursos
-
```
SELECT
    e.nombre AS estudiante,
    AVG(c.calificacion) AS promedio
FROM Calificacion c
JOIN Estudiante e ON c.id_estudiante = e.id_estudiante
WHERE e.id_estudiante = 1
GROUP BY e.nombre;
```

	estudiante	promedio
1	Juan Pérez	87.75