Beatriz Entrialgo Rodriguez

Tarea Bases de Datos SQL

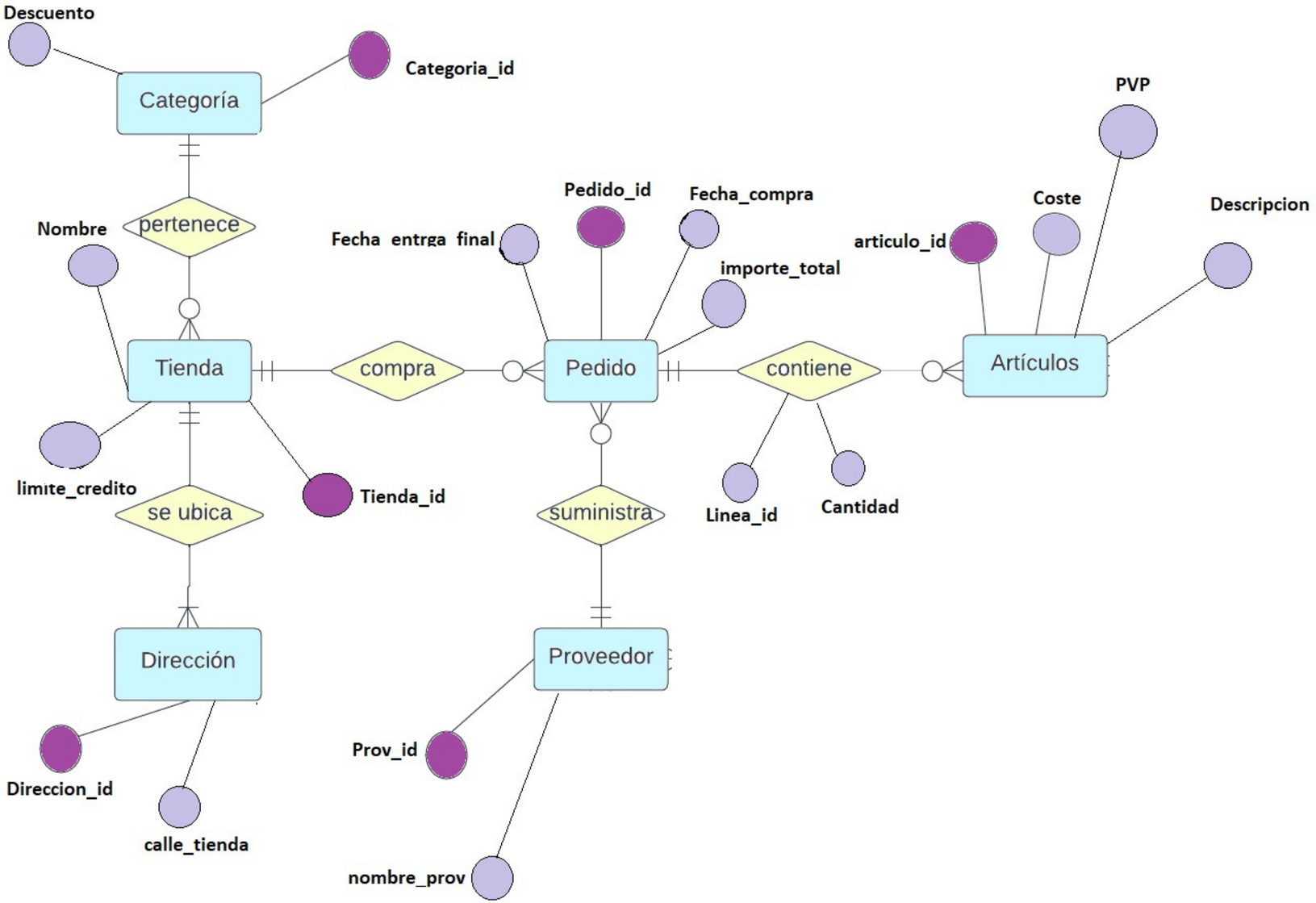
Nos contratan para realizar el diseño funcional de base de datos (diagrama Entidad – Relación y Relacional) de un modelo de negocio de un mayorista de moda que distribuye a 12.000 tiendas en todo el mundo, con las siguientes restricciones:

- Para cada tienda necesitamos almacenar la siguiente información: ID de tienda (único), direcciones de entrega (varias por tienda), límite de crédito de cada tienda (en ningún caso debe superar los 30.000 €) y nivel de descuento que estará en función de la categoría de tienda y en ningún caso superará el 20%.
- Las tiendas se dividen en categorías, de acuerdo con el volumen de compras anuales, de forma que a mayor volumen de compra los descuentos serán mayores.
- Las tiendas realizan pedidos al mayorista, es fundamental identificar en el pedido la tienda que lo realiza, la dirección de entrega, así como la fecha en la que se realiza dicho pedido.
- Cada pedido dispone de líneas de pedido, en cada línea de pedido se determinará el identificador del artículo, descripción, su número de unidades y el importe de este.
- Los artículos se distribuyen a cada tienda por varios proveedores, para identificar a cada proveedor se tendrá en cuenta, el ID de proveedor (único) y dirección de contacto.

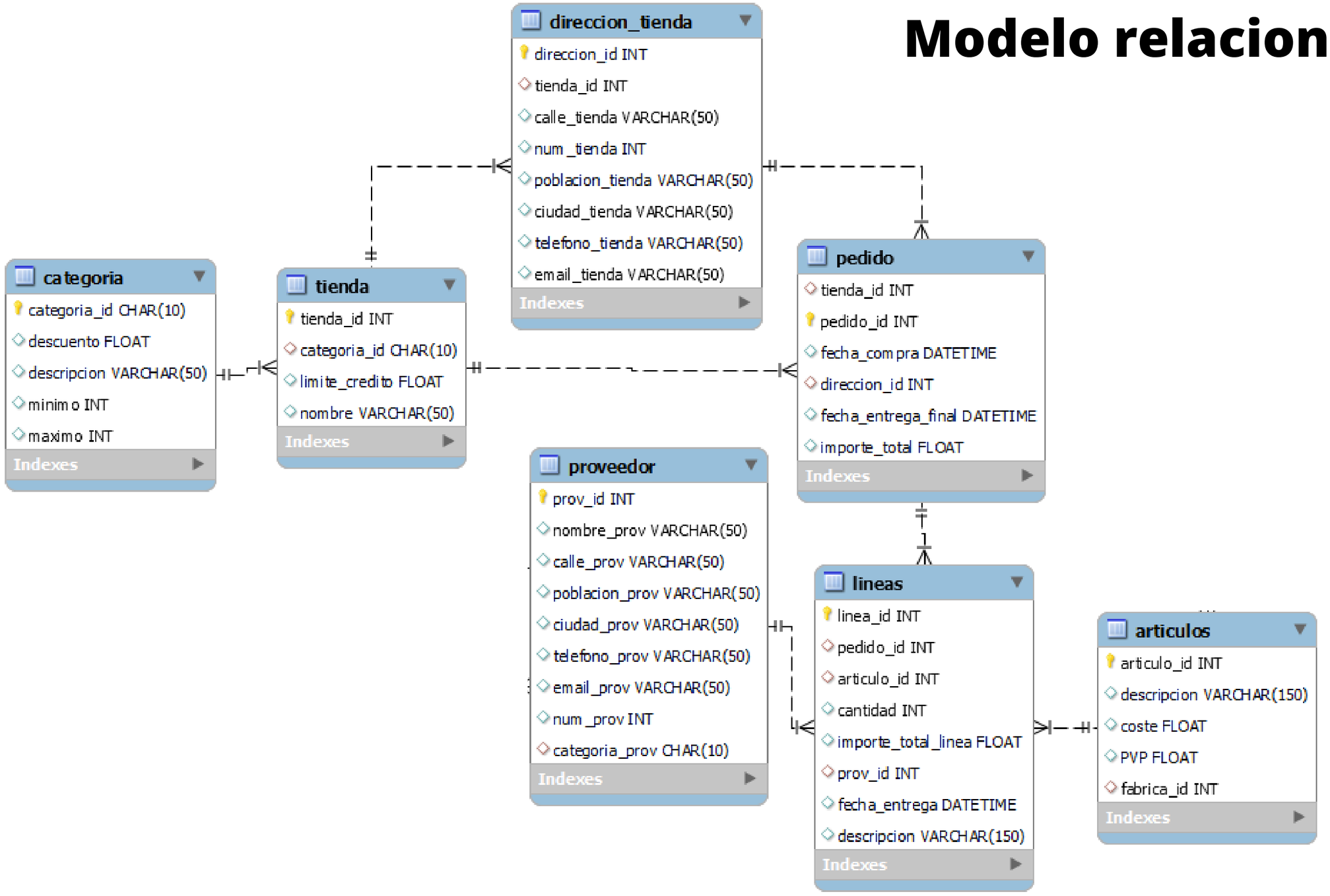
Nota: Una dirección se entenderá como Nº, Calle, Población, Ciudad, Teléfono y email. Una fecha incluye hora.

Determinar el diagrama Entidad – Relación y el paso a tablas (Modelo Relacional).

Modelo Entidad_Relación



Modelo relacional



2. Utilizar la instrucción SQL de inserción de datos para insertar una fila en la tabla de pedidos.

Orden de inserción según las relaciones entre tablas

```
INSERT INTO Categoria (categoria_id, descuento, descripcion, minimo)
VALUES ('A', '0.20', 'Excelente', '25000' );
```

```
INSERT INTO Direccion_Tienda (direccion_id, tienda_id, calle_tienda, num_tienda, poblacion_tienda, ciudad_tienda, telefono_tienda, email_tienda)
VALUES ('0001', '0001', 'Arenal', '1', 'Madrid', 'Madrid', '+34 612 34 56 78', 'primeratienda@gmail.com');
```

```
INSERT INTO Tienda (tienda_id, categoria_id, limite_credito, nombre)
VALUES ('0001','A','30000','Bea');
```

```
INSERT INTO Pedido (tienda_id, pedido_id, fecha_compra, direccion_id, fecha_entrega_final)
VALUES ('0001', '0001','22-02-21 12:00:00', '0001', '22-01-22 12:00:00');
```

3. Visualizar mediante una instrucción SQL todas las tiendas que componen la red de distribución de la fábrica, se deberán detallar: nombre de la tienda, dirección, descripción de la categoría, descuento y límite de crédito asociado a la tienda.

```
SELECT t.categoria_id
,t.nombre
,t.limite_credito
,c.descuento
,d.*
```

RIGHT JOIN para obetener todas las tiendas registradas

```
FROM (Direccion_Tienda AS d RIGHT JOIN Tienda AS t ON
d.tienda_id = t.tienda_id)
INNER JOIN Categoria AS c ON t.categoria_id = c.categoria_id;
```

4. Visualizar mediante una instrucción SQL los pedidos suministrados a cada una de las tiendas en un período determinado (último año). Se deberán obtener los siguientes datos: número de pedido, fecha de suministro, dirección de entrega, y el importe total del pedido.

```
SELECT p.pedido_id
,p.fecha_entrega_final
,l.articulo_id
,a.descripcion
,l.cantidad
,c.descuento
,a.PVP
, l.cantidad *a.PVP AS importe_total
, l.cantidad *a.PVP * c.descuento AS importe_total_con_descuento
,YEAR(p.fecha_entrega_final) AS año
,d.*

FROM (Pedido AS p INNER JOIN Direccion_Tienda AS d ON
p.direccion_id = d.direccion_id)
INNER JOIN Tienda AS t ON t.tienda_id = d.direccion_id
INNER JOIN Categoria AS c ON c.categoria_id = t.categoria_id
INNER JOIN Lineas AS l ON l.pedido_id = p.pedido_id
INNER JOIN Articulos AS a ON a.articulo_id = l.articulo_id

WHERE YEAR(p.fecha_entrega_final) = '2022';
```

5. Identificar mediante una consulta SQL los repartos realizados por cada uno de los proveedores destinados a ello. Se deberá identificar al menos: Nombre del proveedor de reparto, su dirección y la relación de los artículos suministrados en cada reparto.

```
SELECT l.linea_id
, p.*
,a.articulo_id
FROM (Lineas AS l INNER JOIN Proveedor AS p ON
l.prov_id = p.prov_id)
INNER JOIN Articulos as a ON
l.articulo_id = a.articulo_id;
```

6. Totalizar los repartos anuales realizados por cada proveedor de reparto.

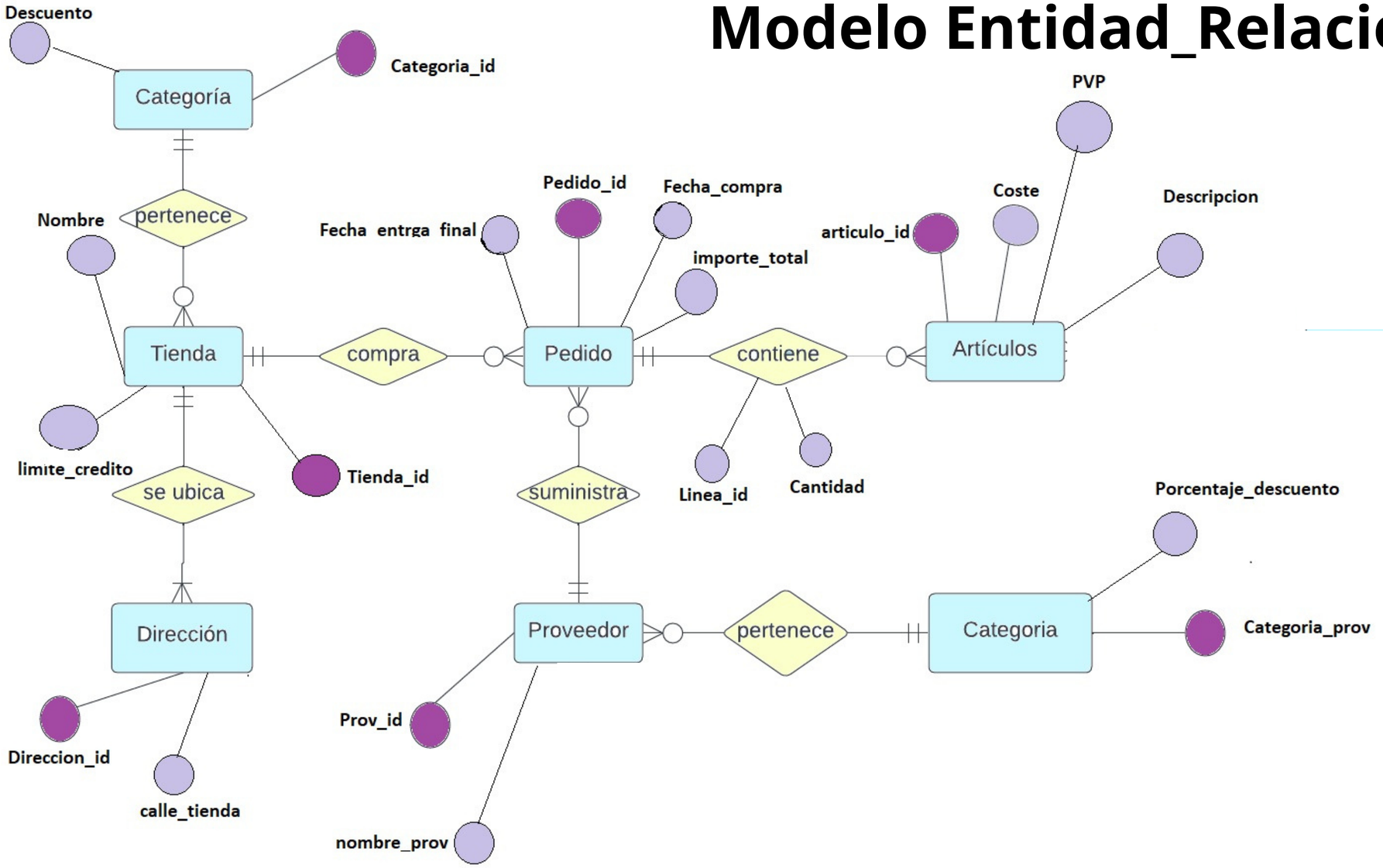
```
SELECT p.prov_id
,COUNT(l.linea_id) AS repartos_totales
,YEAR(l.fecha_entrega) AS año
FROM Lineas AS l INNER JOIN Proveedor AS p ON
l.prov_id = p.prov_id
GROUP BY
prov_id
,año;
```

7. Identificar los cambios a realizar en el modelo relacional y en BBDD para clasificar a los proveedores de reparto en categorías, de la misma forma que clasificamos las tiendas por categorías.

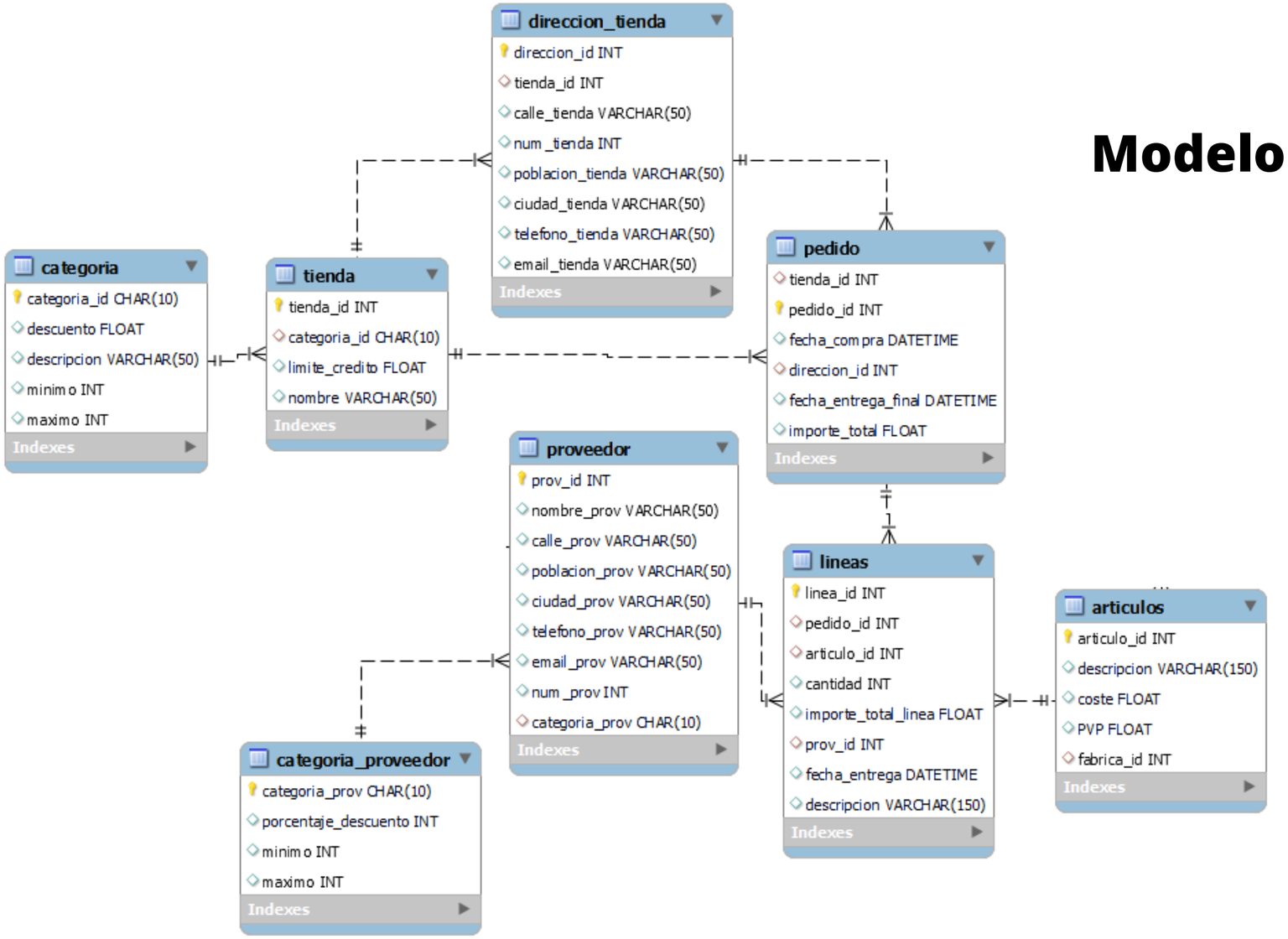
```
CREATE TABLE Categoria_Proveedor (
    categoria_prov char(10),
    porcentaje_descuento int,
    minimo int,
    maximo int,
    PRIMARY KEY (categoria_prov)
);

ALTER TABLE Proveedor
ADD categoria_prov char(10);
ALTER TABLE Proveedor
ADD FOREIGN KEY (categoria_prov) REFERENCES Categoria_Proveedor(categoria_prov);
```

7. Identificar los cambios a realizar en el modelo relacional y en BBDD para clasificar a los proveedores de reparto en categorías, de la misma forma que clasificamos las tiendas por categorías.

Modelo Entidad_Relación actualizado



7. Identificar los cambios a realizar en el modelo relacional y en BBDD para clasificar a los proveedores de reparto en categorías, de la misma forma que clasificamos las tiendas por categorías.



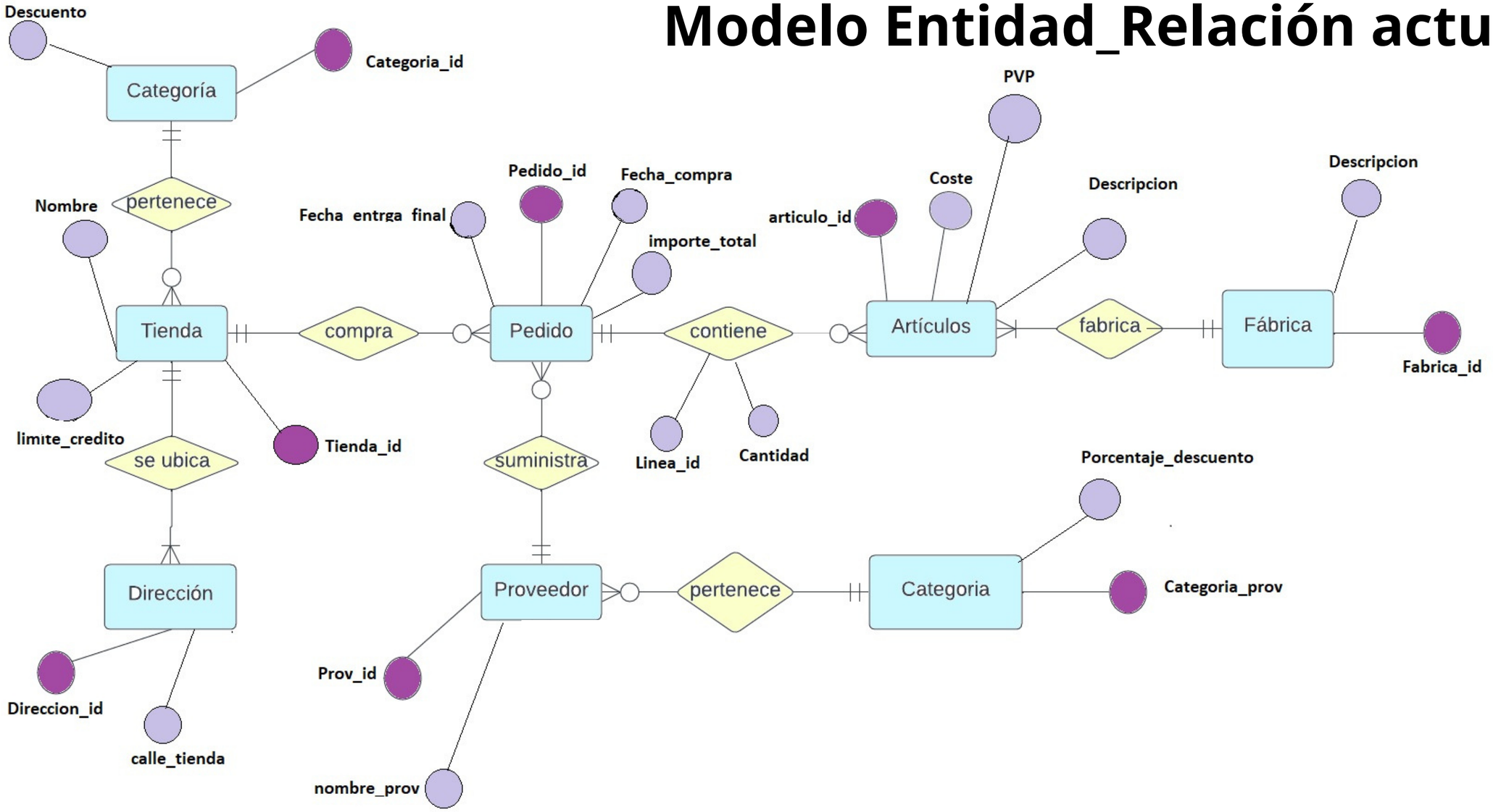
Modelo relacional actualizado

8. Necesitamos introducir nuevos atributos en la tabla de artículos, la fábrica ha descubierto que puede comprar un artículo de parecidas características al nuestro y distribuirlo como marca blanca. Necesitaríamos:

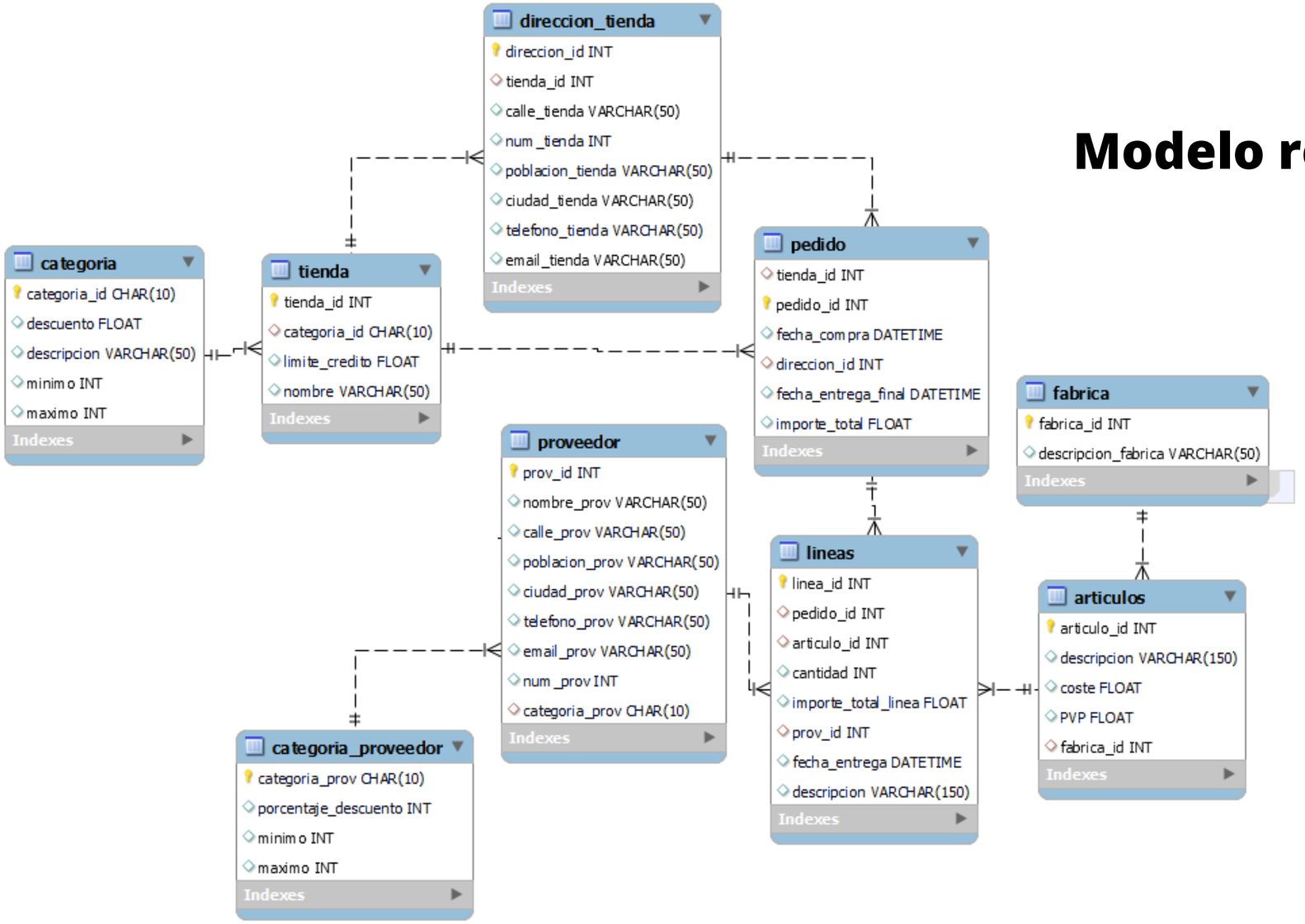
```
CREATE TABLE Fabrica (
  fabrica_id int AUTO_INCREMENT,
  descripcion_fabrica varchar(50),
  PRIMARY KEY (fabrica_id)
);
ALTER TABLE Articulos
ADD fabrica_id int;
ALTER TABLE Articulos
ADD FOREIGN KEY (fabrica_id) REFERENCES Fabrica(fabrica_id);
```

8. Necesitamos introducir nuevos atributos en la tabla de artículos, la fábrica ha descubierto que puede comprar un artículo de parecidas características al nuestro y distribuirlo como marca blanca. Necesitaríamos:

Modelo Entidad_Relación actualizado



8. Necesitamos introducir nuevos atributos en la tabla de artículos, la fábrica ha descubierto que puede comprar un artículo de parecidas características al nuestro y distribuirlo como marca blanca. Necesitaríamos:



Modelo relacional actualizado

9. Queremos ampliar la información del proveedor de suministro, para ello necesitaríamos incorporar los datos relativos a las zonas de cobertura de este (Países y Regiones). Determinar los cambios a realizar a nivel físico y lógico.

```sql
CREATE TABLE Region (
  codigo_region int AUTO_INCREMENT,
  nombre_region varchar(50),
  PRIMARY KEY (codigo_region)
);

CREATE TABLE Pais (
  codigo_pais int AUTO_INCREMENT,
  nombre_pais varchar(50),
  PRIMARY KEY (codigo_pais)
);
```

Es necesario crear las tablas individuales antes de establecer relaciones

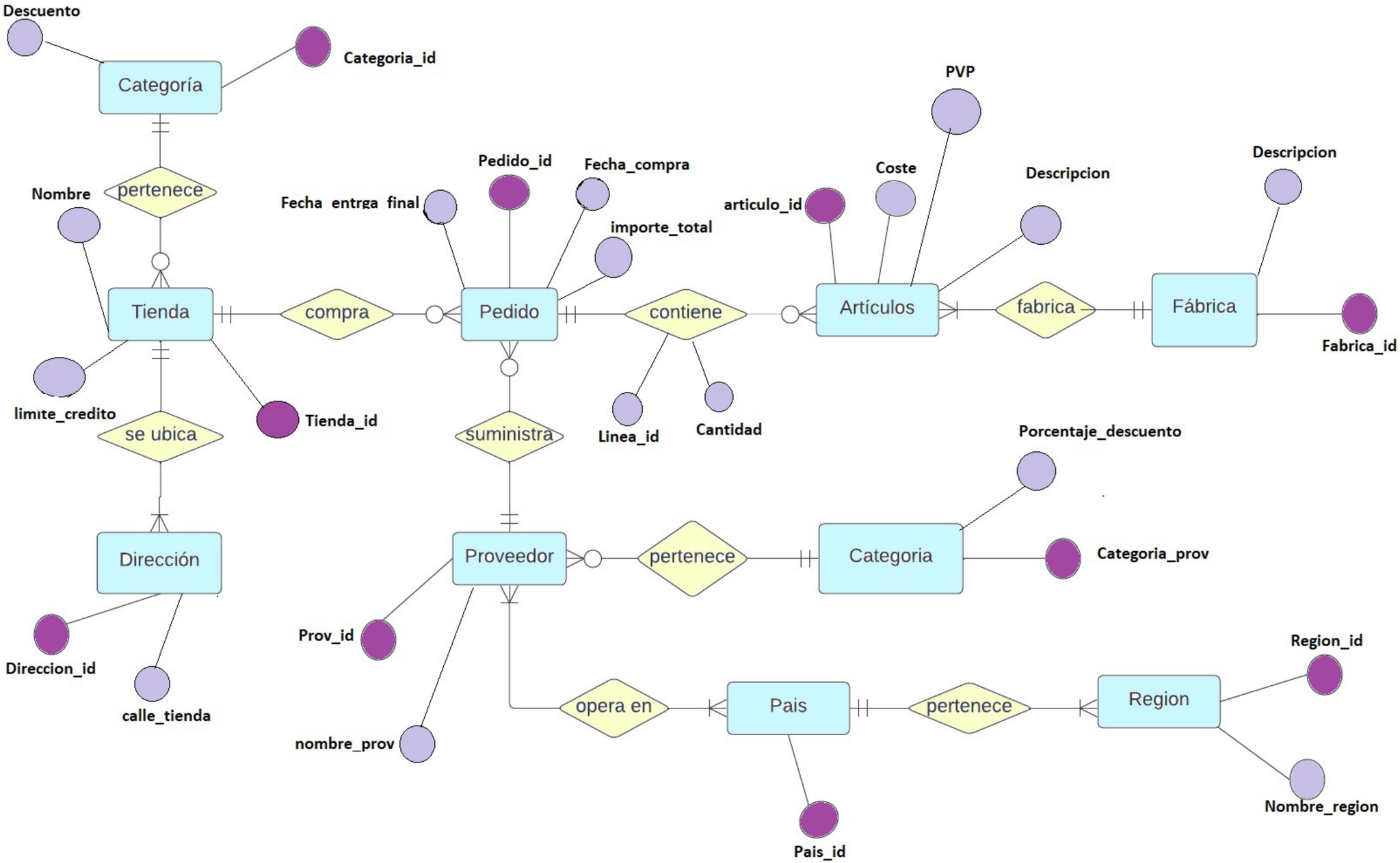
```sql
ALTER TABLE Region
ADD codigo_pais int;
ALTER TABLE Region
ADD FOREIGN KEY (codigo_pais) REFERENCES Pais(codigo_pais);
```

9. Queremos ampliar la información del proveedor de suministro, para ello necesitaríamos incorporar los datos relativos a las zonas de cobertura de este (Países y Regiones). Determinar los cambios a realizar a nivel físico y lógico.

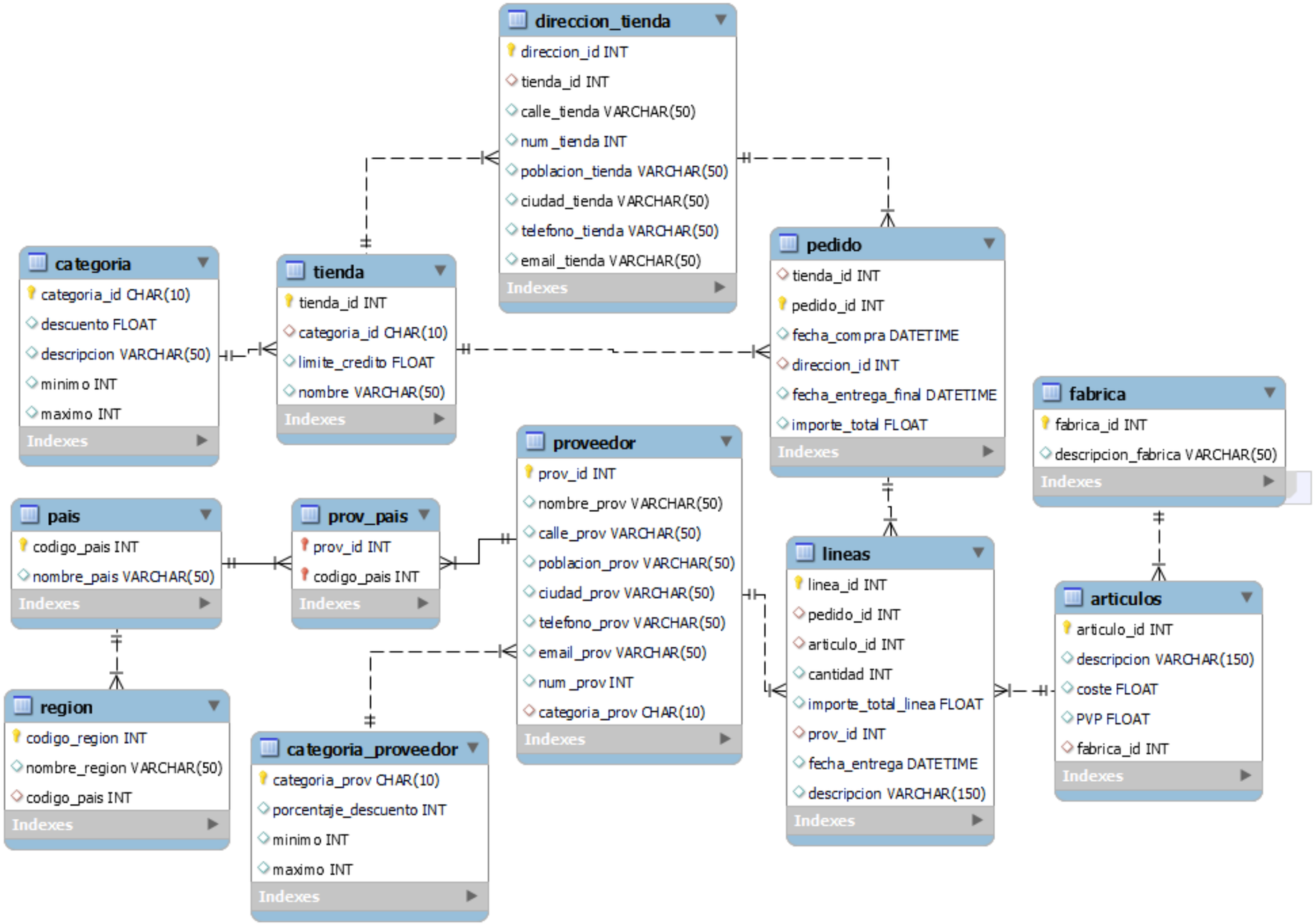
```
CREATE TABLE Prov_Pais (
  prov_id int ,
  codigo_pais int,
  PRIMARY KEY (prov_id, codigo_pais),
  FOREIGN KEY (prov_id) REFERENCES Proveedor(prov_id),
  FOREIGN KEY (codigo_pais) REFERENCES Pais(codigo_pais)
);
```

Soluciona la relación n:m entre las tablas Pais y Proveeder

Modelo Entidad_Relación final



Modelo relacional final



10. ¿Qué podríamos hacer para realizar un backup de la tabla de pedidos / líneas de pedido? Esto es, necesitamos hacer todos los días un proceso de backup a otra Base de Datos en las que se consolida toda la venta del grupo (pedidos, líneas de pedido).

```
CREATE DATABASE Backup_Pedidos_Lineas;
USE Backup_Pedidos_Lineas;

CREATE TABLE Backup_Pedido (
  backup_pedido_id int AUTO_INCREMENT,
  p_tienda_id int,
  p_pedido_id int ,
  p_fecha_compra datetime,
  p_direccion_id int,
  p_fecha_entrega_final datetime,
  p_importe_total float,
  PRIMARY KEY (backup_pedido_id)
);

CREATE TABLE Backup_Lineas (
  backup_linea_id int AUTO_INCREMENT,
  backup_pedido_id int,
  l_linea_id int,
  l_pedido_id int,
  l_articulo_id int,
  l_cantidad int,
  l_importe_total_linea float,
  l_prov_id int,
  l_fecha_entrega datetime,
  l_descripcion varchar(150),
  PRIMARY KEY (backup_linea_id),
  FOREIGN KEY (backup_pedido_id) REFERENCES Backup_Pedido (backup_pedido_id)
  )
```

10. ¿Qué podríamos hacer para realizar un backup de la tabla de pedidos / líneas de pedido? Esto es, necesitamos hacer todos los días un proceso de backup a otra Base de Datos en las que se consolida toda la venta del grupo (pedidos, líneas de pedido).

```
DELIMITER //
CREATE TRIGGER Backup_Trigger AFTER INSERT ON Pedido
FOR EACH ROW
BEGIN
    INSERT INTO Backup_Pedido (p_tienda_id ,p_pedido_id ,p_fecha_compra ,p_direccion_id, p_fecha_entrega_final, p_importe_total)
    VALUES ( NEW.p_tienda_id ,NEW.p_pedido_id,NEW.p_fecha_compra, NEW.p_direccion_id, NEW.p_fecha_entrega_final, NEW.p_importe_total);

    INSERT INTO Backup_Lineas ( backup_pedido_id , l_linea_id ,l_pedido_id ,l_articulo_id ,l_cantidad ,l_importe_total_linea ,l_prov_id ,l_fecha_entrega ,l_descripcion)
    VALUES( NEW.backup_pedido_id , NEW.l_linea_id ,NEW.l_pedido_id ,NEW.l_articulo_id ,NEW.l_cantidad ,NEW.l_importe_total_linea ,NEW.l_prov_id ,NEW.l_fecha_entrega ,NEW.l_descripcion);
END //
```

Trigger para hacer una copia de los datos de ventas en la base de datos Backup_Pedidos_Lineas

Conclusiones

- **Es primordial pensar en el modelo entidad-relación correcto para estructurar la información antes de la creación de la base de datos para evitar información redundante**
- **Un modelo relacional claro facilita la realización de consultas**

Sugerencias

- **Incluir en el modelo información sobre stock de artículos y clasificación**