



# Trabajo Voluntario

## Bases de datos

### Gestión de datos tienda de ropa Textil Maria Jesús S.L

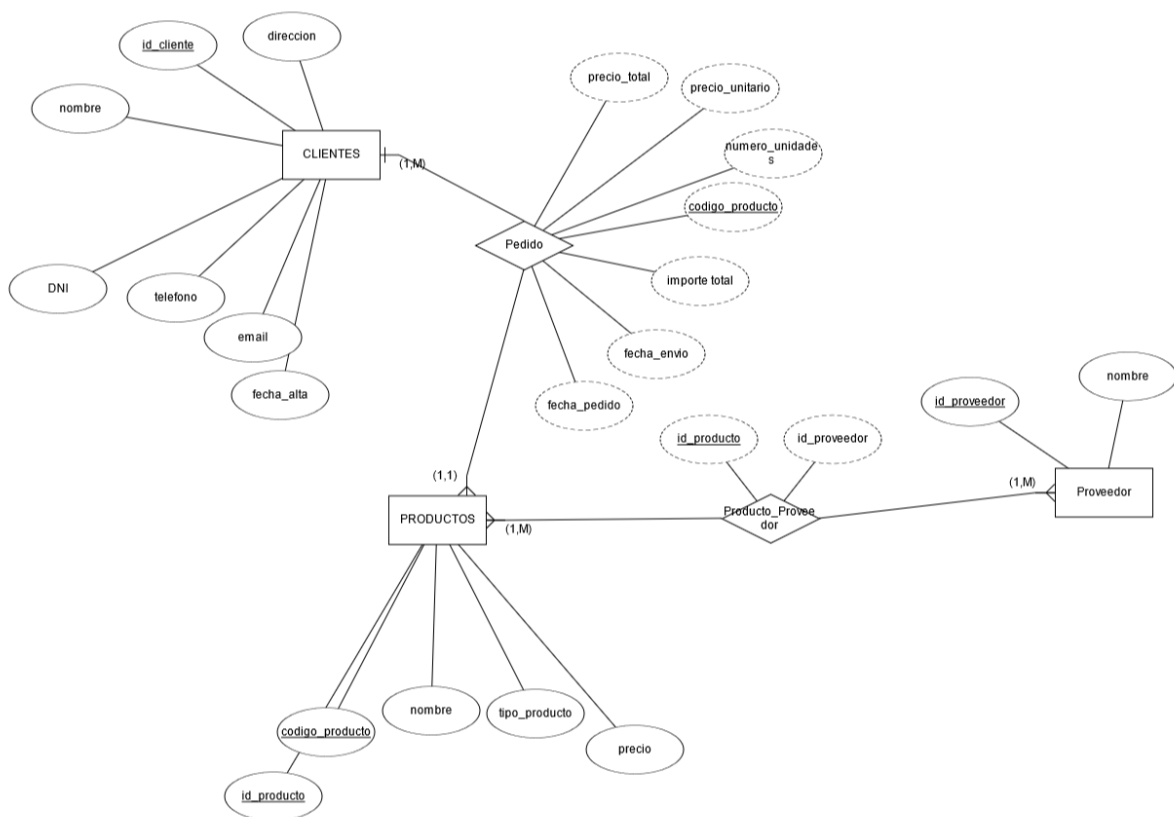
**Trabajo realizado por David Morgade Gil – 1º DAM – MEDAC – PACIFICO**

# INTRODUCCIÓN

Para la realización del diagrama entidad relación he utilizado la herramienta ERDPLUS, siguiendo el modelo de chen.

Tendremos 3 entidades principales con sus atributos que serán Clientes, Productos y Proveedor con sus correspondientes atributos.

Luego como relaciones tendremos pedido que se encarga de relacionar las entidades de clientes con productos y producto\_proveedor que es la que relaciona cada producto con el proveedor que lo tiene (en este caso varios productos pueden tener varios proveedores y un proveedor puede tener varios productos).





# CONTENIDO

Para la realización de este ejercicio he utilizado SQL express edition junto con SQL developer como SGBD para poder crear la base de datos y posteriormente realizar las consultas en el apartado final.

Empezamos creando la tabla cliente:

Hoja de Trabajo: Generador de Consultas

```
CREATE TABLE Cliente (  
  IDCliente NUMBER PRIMARY KEY,  
  Nombre VARCHAR2(255),  
  Direccion VARCHAR2(255),  
  DNI VARCHAR2(15),  
  Telefono VARCHAR2(15),  
  Email VARCHAR2(255),  
  FechaAlta DATE  
);
```

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 IDCLIENTE	NUMBER	No	(null)	1 (null)	
2 NOMBRE	VARCHAR2(255 BYTE)	Yes	(null)	2 (null)	
3 DIRECCION	VARCHAR2(255 BYTE)	Yes	(null)	3 (null)	
4 DNI	VARCHAR2(15 BYTE)	Yes	(null)	4 (null)	
5 TELEFONO	VARCHAR2(15 BYTE)	Yes	(null)	5 (null)	
6 EMAIL	VARCHAR2(255 BYTE)	Yes	(null)	6 (null)	
7 FECHAALTA	DATE	Yes	(null)	7 (null)	

Creamos la tabla producto:

Hoja de Trabajo: Generador de Consultas

```
CREATE TABLE Producto (  
 CodigoProducto NUMBER PRIMARY KEY,  
  Nombre VARCHAR2(255),  
  TipoProducto VARCHAR2(255)  
);
```

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 CODIGOPRODUCTO	NUMBER	No	(null)	1 (null)	
2 NOMBRE	VARCHAR2(255 BYTE)	Yes	(null)	2 (null)	
3 TIPOPRODUCTO	VARCHAR2(255 BYTE)	Yes	(null)	3 (null)	

También crearemos la tabla de proveedor al igual que hicimos en nuestro DER:

Hoja de Trabajo: Generador de Consultas

```
CREATE TABLE Proveedor (  
  IDProveedor NUMBER PRIMARY KEY,  
  Nombre VARCHAR2(255)  
);
```

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 IDPROVEEDOR	NUMBER	No	(null)	1 (null)	
2 NOMBRE	VARCHAR2(255 BYTE)	Yes	(null)	2 (null)	

Ahora crearemos Pedido, que será también una tabla independiente y la cual tendrá como referencia en forma de llave foránea el ID del cliente que hace el pedido:

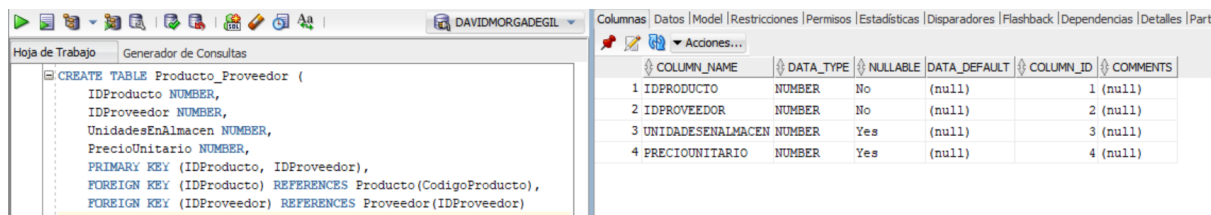
Hoja de Trabajo: Generador de Consultas

```
CREATE TABLE Pedido (  
  NumeroPedido NUMBER PRIMARY KEY,  
  FechaPedido DATE,  
  FechaEnvio DATE,  
  ImporteTotal NUMBER,  
  IDCliente NUMBER,  
  FOREIGN KEY (IDCliente) REFERENCES Cliente(IDCliente)  
);
```

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 NUMEROPEDIDO	NUMBER	No	(null)	1 (null)	
2 FECHAPEDIDO	DATE	Yes	(null)	2 (null)	
3 FECHAENVIO	DATE	Yes	(null)	3 (null)	
4 IMPORTETOTAL	NUMBER	Yes	(null)	4 (null)	
5 IDCLIENTE	NUMBER	Yes	(null)	5 (null)	



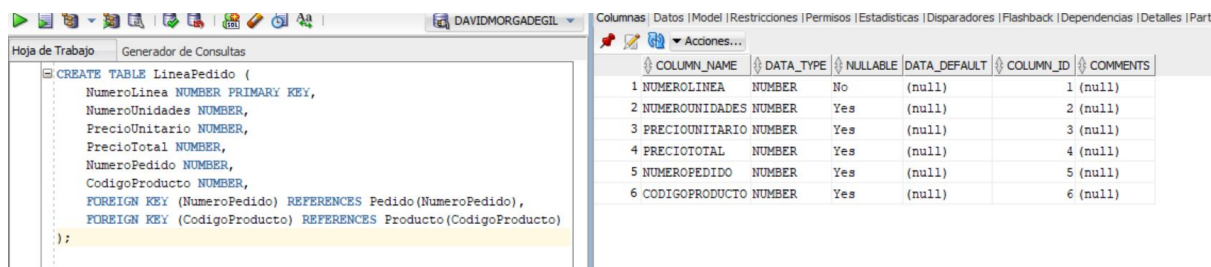
También tendremos que añadir la tabla producto\_proveedor, para relacionar los productos con el proveedor (tabla intermedia) usando el IDProducto y el IDProveedor, aquí podremos definir las unidades en almacén y el precio del producto según el proveedor:



The screenshot shows the SQL Developer interface. On the left, the 'Hoja de Trabajo' (Worksheet) tab is active, displaying the SQL code to create the 'Producto\_Proveedor' table. The code defines columns: IDProducto (NUMBER), IDProveedor (NUMBER), UnidadesEnAlmacen (NUMBER), PrecioUnitario (NUMBER), and sets a primary key on IDProducto and IDProveedor. It also includes foreign key references to the 'Producto' and 'Proveedor' tables. On the right, the 'Columnas' (Columns) tab is selected, showing a table with 6 columns: COLUMN\_NAME, DATA\_TYPE, NULLABLE, DATA\_DEFAULT, COLUMN\_ID, and COMMENTS. The table lists the 4 columns of the 'Producto\_Proveedor' table.

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 IDPRODUCTO	NUMBER	No	(null)	1	(null)
2 IDPROVEEDOR	NUMBER	No	(null)	2	(null)
3 UNIDADESENALMACEN	NUMBER	Yes	(null)	3	(null)
4 PRECIOUNITARIO	NUMBER	Yes	(null)	4	(null)

Finalmente, solo nos faltaría la línea de pedido, que es la que relacionará el numero del pedido con el producto y se encarga de calcular el precio total y tendrá también el número de pedido:



The screenshot shows the SQL Developer interface. On the left, the 'Hoja de Trabajo' (Worksheet) tab is active, displaying the SQL code to create the 'LineaPedido' table. The code defines columns: NumeroLinea (NUMBER, PRIMARY KEY), NumeroUnidades (NUMBER), PrecioUnitario (NUMBER), PrecioTotal (NUMBER), NumeroPedido (NUMBER), and CódigoProducto (NUMBER). It sets a primary key on NumeroLinea and includes foreign key references to the 'Pedido' and 'Producto' tables. On the right, the 'Columnas' (Columns) tab is selected, showing a table with 6 columns: COLUMN\_NAME, DATA\_TYPE, NULLABLE, DATA\_DEFAULT, COLUMN\_ID, and COMMENTS. The table lists the 6 columns of the 'LineaPedido' table.

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 NUMEROLINEA	NUMBER	No	(null)	1	(null)
2 NUMEROUNIDADES	NUMBER	Yes	(null)	2	(null)
3 PRECIOUNITARIO	NUMBER	Yes	(null)	3	(null)
4 PRECIOTOTAL	NUMBER	Yes	(null)	4	(null)
5 NUMEROPEDIDO	NUMBER	Yes	(null)	5	(null)
6 CODIGOPRODUCTO	NUMBER	Yes	(null)	6	(null)

Con esto ya podríamos agregar algunos datos a nuestra base de datos y comenzar a realizar algunas consultas.



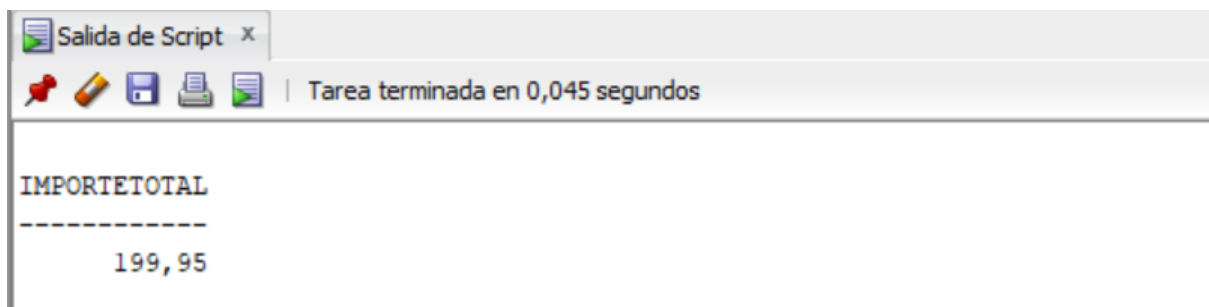
# CONCLUSION

Finalmente realizaremos algunas consultas SQL sobre los datos que hemos introducido de manera ficticia en nuestra base de datos.

Ver el importe de un pedido de un cliente en base a su DNI, la sentencia sería:

```
SELECT IMPORTETOTAL  
FROM Pedido  
WHERE IDCliente = (SELECT IDCliente FROM Cliente WHERE DNI = '12345678X');
```

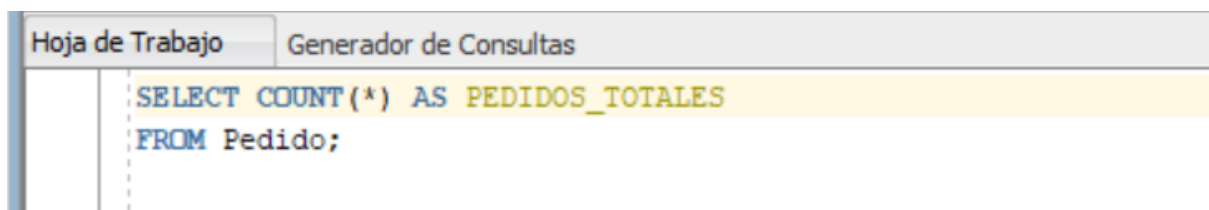
El output:



Salida de Script x | Tarea terminada en 0,045 segundos

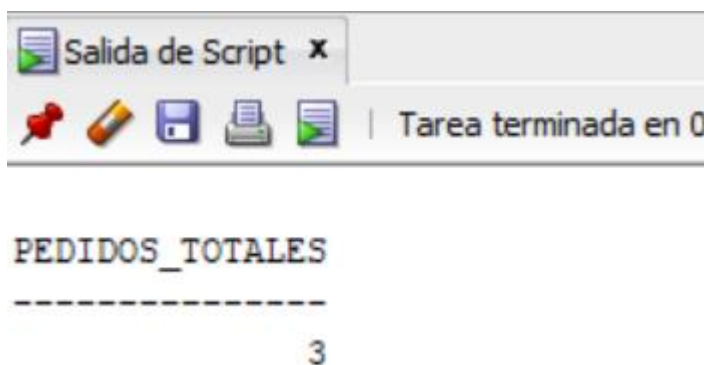
IMPORTETOTAL
199,95

Contar la cantidad de pedidos totales usando COUNT y un alias, sentencia:



```
SELECT COUNT(*) AS PEDIDOS_TOTALES  
FROM Pedido;
```

Output:



Salida de Script x | Tarea terminada en 0

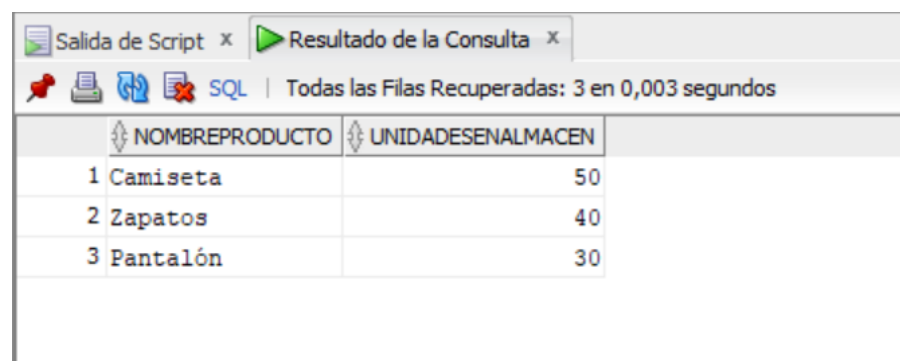
PEDIDOS_TOTALES
3



Ver todas las unidades en almacén de cada producto, sentencia:

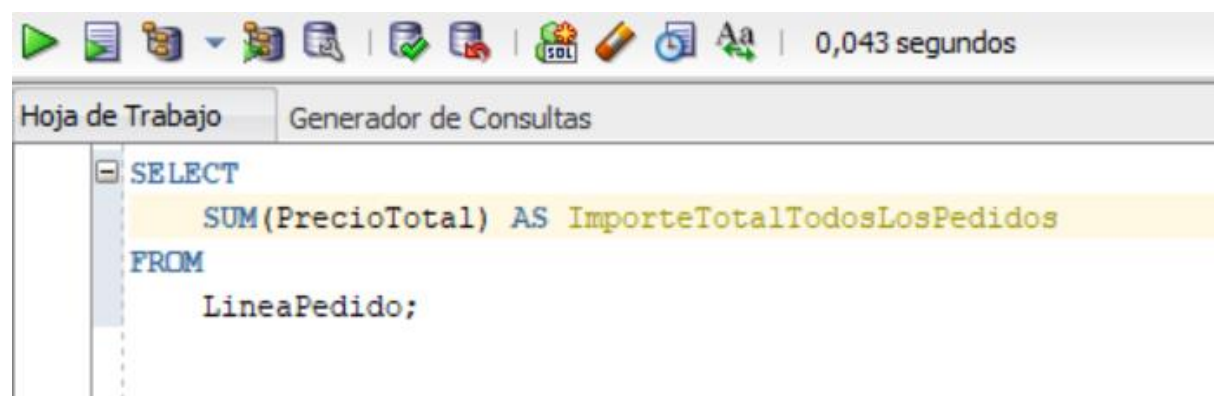
```
SELECT
    Producto.Nombre AS NombreProducto,
    Producto_Proveedor.UnidadesEnAlmacen
FROM
    Producto
JOIN Producto_Proveedor ON Producto.CodigoProducto = Producto_Proveedor.IDProducto
ORDER BY
    UnidadesEnAlmacen DESC;
```

Output:



NOMBREPRODUCTO	UNIDADESENALMACEN
1 Camiseta	50
2 Zapatos	40
3 Pantalón	30

Sacar el importe total de todos los pedidos utilizando la función SUM, sentencia:



```
SELECT
    SUM(PrecioTotal) AS ImporteTotalTodosLosPedidos
FROM
    LineaPedido;
```

Output:

IMPORTETOTALTODOSLOSPEDIDOS

---

269,9



# BIBLIOGRAFÍA

## *Recursos utilizados durante el desarrollo*

[StackOverFlow](#): Para la resolución de dudas concretas con las consultas SQL.

[W3School](#): Práctica de sentencias SQL previo a la realización del trabajo

[Oracle](#): Consulta de la documentación sobre el uso de SQL Developer.