TALLER BASES DE DATOS RELACIONALES

ABEL AUDINO PANTOJA RODRIGUEZ

Presentado a Brayan Arcos

INSTITUTO TECNOLOGICO DEL PUTUMAYO

DESARROLLO DE BASE DE DATOS

MOCOA-PUTUMAYO

2024

INDICE

RESUMEN EJECUTIVO

Este documento presenta el diseño y la implementación de una base de datos para gestionar productos, proveedores y órdenes de compra. El objetivo principal es crear una estructura eficiente que permita registrar y consultar información clave sobre los productos, los proveedores que los suministran y las órdenes de compra generadas. Se han identificado las entidades y sus relaciones, y se ha implementado una solución SQL que garantiza la integridad y escalabilidad del sistema. La base de datos diseñada incluye cuatro tablas principales: Proveedores, Productos, OrdenesCompra, y DetalleOrdenCompra, las cuales están interconectadas para ofrecer un manejo estructurado y eficiente de la información. El proceso sigue una metodología clara, desde el análisis de requerimientos hasta la creación de las tablas y las relaciones correspondientes, verificando la consistencia y funcionalidad del diseño.

INTRODUCCIÓN

El desarrollo de este proceso lo encamino en el ámbito de la gestión empresarial, donde disponer de una base de datos eficiente es crucial para manejar de forma adecuada los productos, proveedores y órdenes de compra. Una base de datos bien diseñada permite automatizar y mejorar los procesos de adquisición de productos y el manejo de proveedores, optimizando la gestión del inventario y los costos operativos. Este informe describe el desarrollo de una base de datos que tiene como objetivo facilitar la administración de productos, registrar los proveedores que los suministran y gestionar las órdenes de compra. La base de datos diseñada está estructurada para garantizar la consistencia, integridad y accesibilidad de la información, permitiendo que los datos estén correctamente organizados y se puedan consultar de manera eficiente.

METODOLOGÍA

El diseño y creación de la base de datos siguió una metodología estructurada en tres fases principales:

1. Análisis de Requerimientos

Se identificaron las necesidades clave del sistema, definiendo las entidades principales: Proveedores, Productos, Órdenes de Compra, y Detalles de Órdenes de Compra. Se analizaron las relaciones entre estas entidades, identificando que cada proveedor puede suministrar múltiples productos, y que cada orden de compra está asociada a un proveedor y contiene varios productos con precios y cantidades específicas.

2. Diseño del Modelo Relacional

Basado en el análisis de requerimientos, se diseñó un modelo relacional que traduce las entidades y sus relaciones en tablas interrelacionadas. Cada tabla se diseñó con claves primarias y foráneas para asegurar la integridad referencial. Se incluyeron campos específicos como el precio y la cantidad de productos en cada orden de compra, con un enfoque en la normalización para evitar redundancias.

3. Implementación en SQL

Una vez definido el diseño, se procedió a la implementación en SQL utilizando MySQL Workbench 8.0 CE, en el cual se utilizó comandos CREATE DATABASE para crear la base de datos ***Inventario*** y CREATE TABLE para generar las tablas con sus relaciones. Las claves primarias y foráneas se implementaron para garantizar la integridad de los datos. Se realizaron pruebas iniciales mediante inserciones de datos y consultas básicas para validar la funcionalidad y consistencia de la base de datos.

4. Herramientas utilizadas

- MySQL Workbench
- Git Hub
- Draw.io

CREACIÓN DE BASE DE DATOS PARA GESTIÓN DE PRODUCTOS, PROVEEDORES Y ÓRDENES DE COMPRA

Requerimientos del Sistema

El sistema debe permitir gestionar:

- Productos: Con nombre, descripción y precio base.
- Proveedores: Con información de contacto y los productos que suministran.
- Órdenes de Compra: Asociadas a proveedores y que contienen múltiples productos, con cantidad y precio específico por producto.

Modelo Conceptual

En esta fase, se identificaron las entidades principales y las relaciones entre ellas:

- Tipo identificación: Entidad que almacena el tipo identificación
- Proveedores: Entidad que almacena información de los proveedores.
- Productos: Entidad que contiene la información de los productos disponibles.
- Órdenes de Compra: Entidad que almacena las órdenes de compra realizadas a los proveedores.
- Detalle de Órdenes de Compra: Entidad intermedia que vincula las órdenes de compra con los productos y añade información específica como la cantidad y el precio por orden.

Diseño Relacional

Una vez identificadas las entidades y relaciones, se procede a convertirlas en un diseño relacional:

Tablas principales:

Tipo identificación (Entidad fuerte):

tipo_identificacion_id: Identificador único de cada tipo de identificación.
tipo_identificacion: descripción de los tipos de identificación (TI, CC, CE, NIT), campo obligatorio.

Proveedores (Entidad fuerte):

proveedor_id: Identificador único de cada proveedor.
tipo_identificacion_id: Identificador del tipo de identificación (clave foránea), campo obligatorio.
identificacion: describe el numero o NIT de identificación del proveedor, campo obligatorio
nombres: Nombre o razón social del proveedor, campo obligatorio.
primerapellido: primer apellido del proveedor.
Segundoapellido: segundo apellido del proveedor.
teléfono: Número de teléfono del proveedor.
email: Correo electrónico del proveedor.
direccion: Dirección del proveedor.

Productos (Entidad fuerte):

producto_id: Identificador único de cada producto.
nombre: Nombre del producto, campo obligatorio.
descripción: Descripción detallada del producto.
precio_base: Precio base del producto, campo obligatorio.

OrdenesCompra(Entidad fuerte):

orden_id: Identificador único de la orden de compra.
proveedor_id: Identificador del proveedor asociado a la orden (clave foránea), campo obligatorio.
fecha: Fecha en la que se realizó la orden, campo obligatorio.

DetalleOrdenCompra (Entidad débil):

detalle_id: Identificador único del detalle de la orden.
orden_id: Identificador de la orden de compra (clave foránea), campo obligatorio.
producto_id: Identificador del producto (clave foránea), campo obligatorio.
cantidad: Cantidad de productos solicitados, campo obligatorio.
precio_unitario: Precio unitario del producto específico para esa orden, campo obligatorio.
precio_total: multiplicación de la cantidad con el precio_unitario

Entidades y Atributos

1. Entidades fuertes:

tipo_identificacion: Tiene su propia clave primaria (tipo_identificacion_id).

Proveedores: Tiene su propia clave primaria (proveedor_id) y no depende de ninguna otra tabla para su existencia.

Productos: Tiene su propia clave primaria (producto_id).

OrdenesCompra: Tiene su propia clave primaria (orden_id).

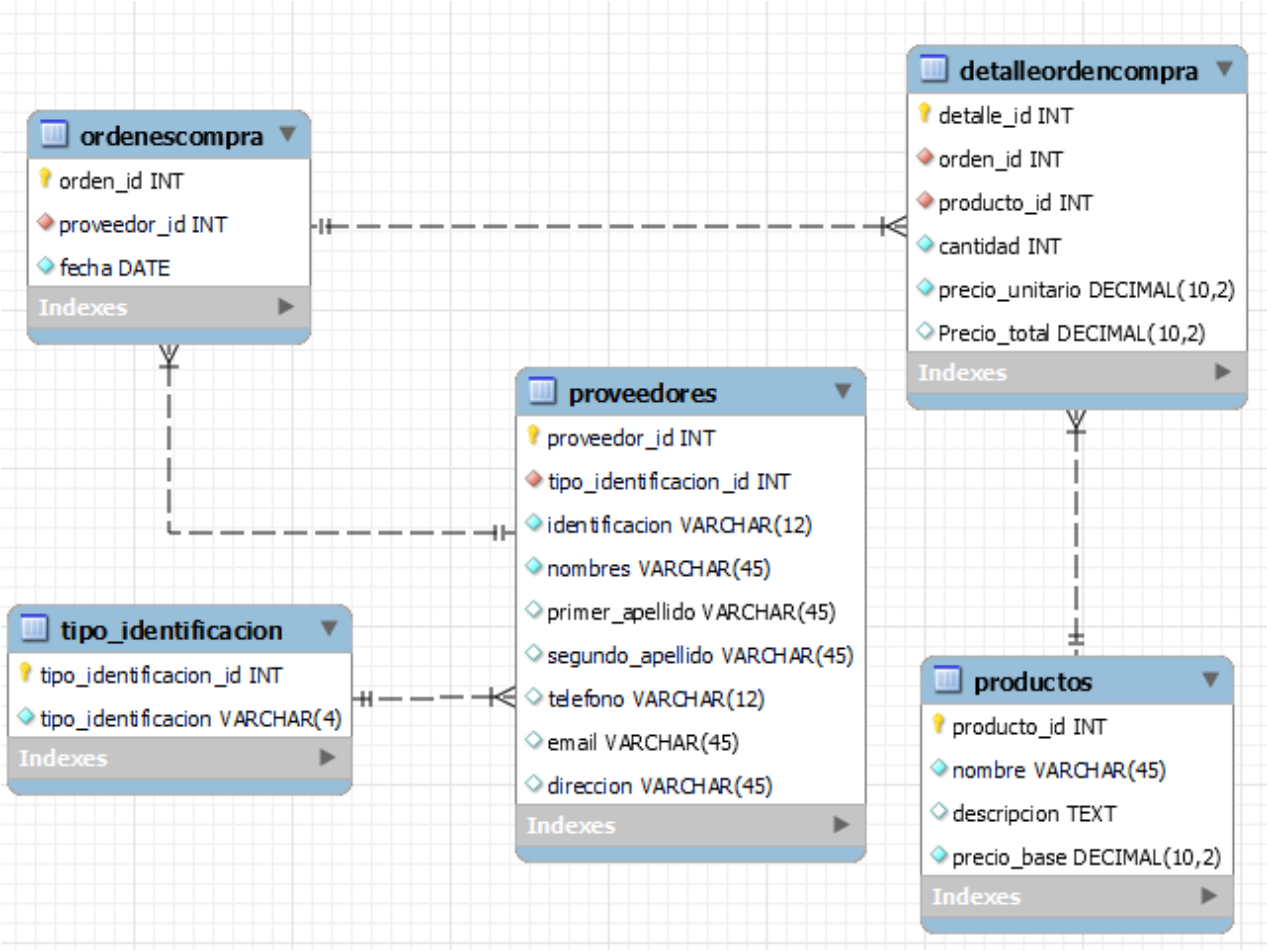
2. Entidades débiles:

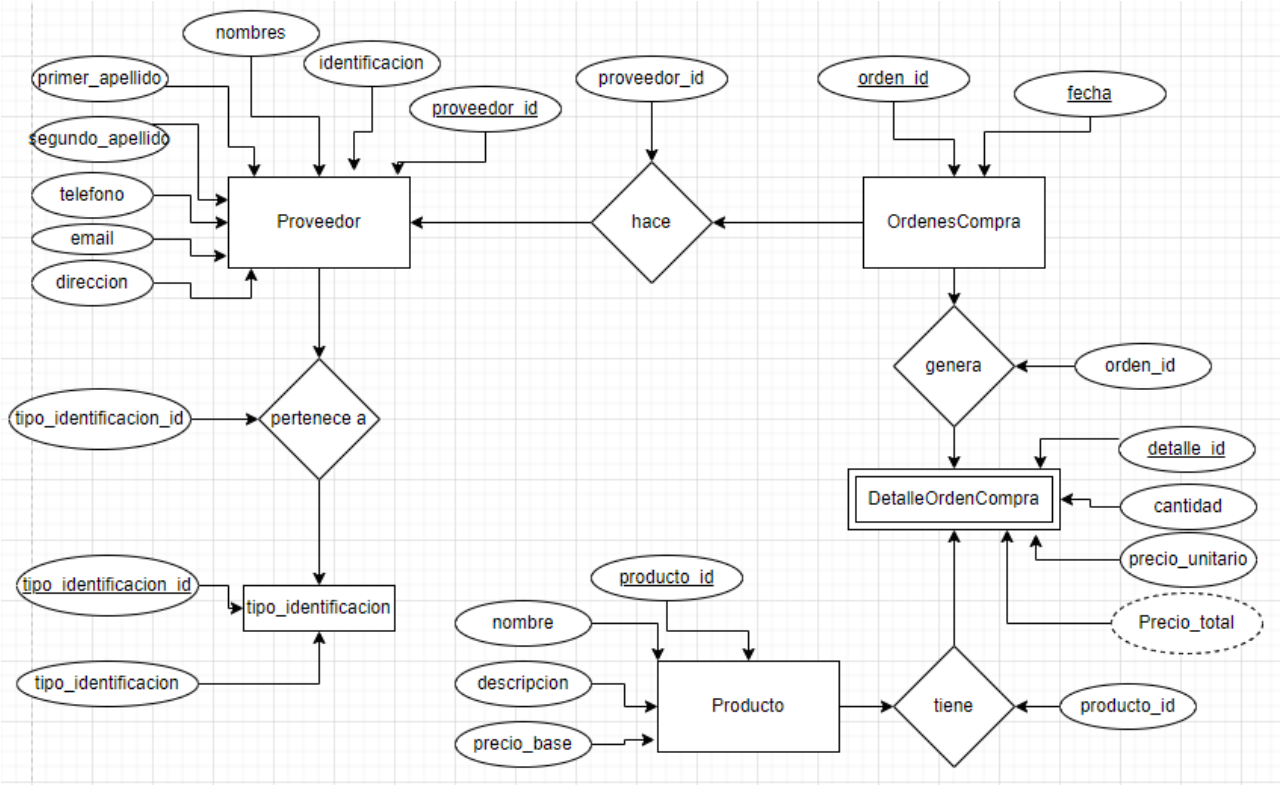
DetalleOrdenCompra: es una entidad débil ya que depende de dos entidades fuertes, OrdenesCompra y Productos. Su clave primaria (detalle_id) es generada, pero depende de la combinación de orden_id y producto_id, que son claves foráneas referenciando a otras tablas. El detalle de una orden solo tiene sentido en el contexto de una orden de compra y productos específicos.

3. Atributos derivados:

Precio_total en la tabla DetalleOrdenCompra: es un atributo derivado, ya que su valor puede calcularse multiplicando la cantidad por el precio_unitario.

Modelo entidad relación





IMPLEMENTACIÓN EN SQL

Ahora procedemos a la implementación del modelo lógico en un sistema de base de datos utilizando MySQL WorkBench.

Paso 1: Creación de la tabla tipo_identificacion

```
CREATE TABLE tipo_identificacion(
tipo_identificacion_id INT PRIMARY KEY AUTO_INCREMENT,
tipo_identificacion VARCHAR(4) NOT NULL
);
```

Explicación: la tabla tipo_identificacion almacena los diferentes tipos de identificación (TI, CC, CE, NIT). Se utiliza AUTO_INCREMENT en tipo_identificacion_id para generar automáticamente el ID único de cada tipo identificación.

Paso 2: Creación de la tabla Proveedores

```
proveedor_id INT PRIMARY KEY AUTO_INCREMENT,
   tipo_identificacion_id INT NOT NULL,
   identificacion VARCHAR(12) NOT NULL,
```

```
    nombres VARCHAR(45) NOT NULL,
    primer_apellido VARCHAR(45),
    segundo_apellido VARCHAR(45),
    telefono VARCHAR(12),
    email VARCHAR(45),
    direccion VARCHAR(45),
    FOREIGN KEY (tipo_identificacion_id) REFERENCES tipo_identificacion(tipo_identificacion_id)
);
```

Explicación: La tabla Proveedores almacena información sobre los proveedores. Se utiliza AUTO_INCREMENT en proveedor_id para generar automáticamente el ID único de cada proveedor.

Paso 3: Creación de la tabla Productos

```
CREATE TABLE Productos (
    producto_id INT PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(45) NOT NULL,
    descripcion TEXT,
    precio_base DECIMAL(10, 2) NOT NULL
);
```

Explicación: La tabla Productos almacena información sobre los productos. El campo precio_base tiene el tipo DECIMAL (10, 2) para manejar valores monetarios con dos decimales.

Paso 4: Creación de la tabla OrdenesCompra

```
CREATE TABLE OrdenesCompra (
    orden_id INT PRIMARY KEY AUTO_INCREMENT,
    proveedor_id INT NOT NULL,
    fecha DATE NOT NULL,
    FOREIGN KEY (proveedor_id) REFERENCES Proveedores(proveedor_id)
);
```

Explicación: La tabla OrdenesCompra almacena información sobre las órdenes de compra. El campo proveedor_id es una clave foránea que referencia a la tabla Proveedores.

Paso 5: Creación de la tabla DetalleOrdenCompra

```
CREATE TABLE DetalleOrdenCompra (
    detalle_id INT PRIMARY KEY AUTO_INCREMENT,
    orden_id INT NOT NULL,
```

```
    producto_id INT NOT NULL,
    cantidad INT NOT NULL,
    precio_unitario DECIMAL(10, 2) NOT NULL,
    Precio_total DECIMAL(10, 2),
    FOREIGN KEY (orden_id) REFERENCES OrdenesCompra(orden_id),
    FOREIGN KEY (producto_id) REFERENCES Productos(producto_id)
);
```

Explicación: La tabla DetalleOrdenCompra almacena los productos incluidos en cada orden, con la cantidad y el precio específico de cada uno. Esta tabla contiene claves foráneas a OrdenesCompra y Productos.

Cardinalidad

ordenescompra ↔ detalleordencompra:

Relación de uno a muchos (1:N).

Una orden de compra puede tener muchos detalles de orden de compra, pero cada detalle de orden de compra pertenece a una única orden de compra.

proveedores ↔ ordenescompra:

Relación de uno a muchos (1:N).

Un proveedor puede tener varias órdenes de compra, pero cada orden de compra está relacionada con un único proveedor.

proveedores ↔ tipo_identificacion:

Relación de muchos a uno (N:1).

Muchos proveedores pueden tener un mismo tipo de identificación, pero cada proveedor tiene solo un tipo de identificación.

detalleordencompra ↔ productos:

Relación de muchos a uno (N:1).

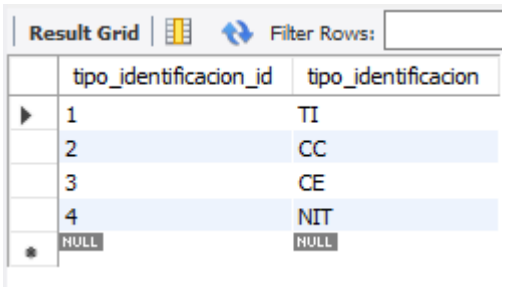
Muchos detalles de orden de compra pueden incluir el mismo producto, pero cada detalle de orden de compra solo puede tener un producto asociado.

Consultas SQL

Mediante código SQL ingreso registros a las tablas de mi base de datos **inventario**:

Tipo de identificación:

```sql
USE inventario;
INSERT INTO tipo_identificacion (tipo_identificacion)
VALUES
("TI"),
("CC"),
("CE"),
("NIT")
```

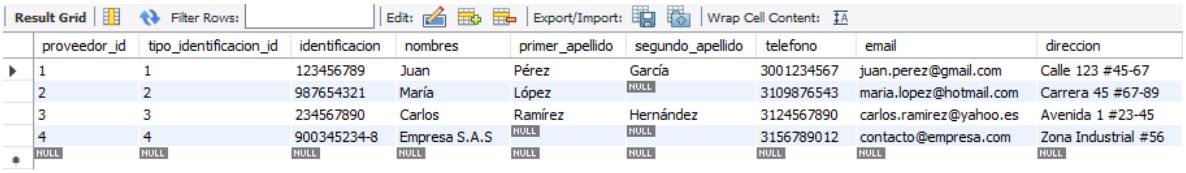


Result Grid | Filter Rows:

tipo_identificacion_id	tipo_identificacion
1	TI
2	CC
3	CE
4	NIT
NULL	NULL

Proveedores:

```sql
USE inventario;
INSERT INTO Proveedores (tipo_identificacion_id, identificacion, nombres, primer_apellido, segundo_apellido, telefono, email, direccion
) VALUES
(1, '123456789', 'Juan', 'Pérez', 'García', '3001234567', 'juan.perez@gmail.com', 'Calle 123 #45-67'),
(2, '987654321', 'María', 'López', NULL, '3109876543', 'maria.lopez@hotmail.com', 'Carrera 45 #67-89'),
(3, '234567890', 'Carlos', 'Ramírez', 'Hernández', '3124567890', 'carlos.ramirez@yahoo.es', 'Avenida 1 #23-45'),
(4, '900345234-8', 'Empresa S.A.S', NULL, NULL, '3156789012', 'contacto@empresa.com', 'Zona Industrial #56');
```



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

proveedor_id	tipo_identificacion_id	identificacion	nombres	primer_apellido	segundo_apellido	telefono	email	direccion
1	1	123456789	Juan	Pérez	García	3001234567	juan.perez@gmail.com	Calle 123 #45-67
2	2	987654321	María	López	NULL	3109876543	maria.lopez@hotmail.com	Carrera 45 #67-89
3	3	234567890	Carlos	Ramírez	Hernández	3124567890	carlos.ramirez@yahoo.es	Avenida 1 #23-45
4	4	900345234-8	Empresa S.A.S	NULL	NULL	3156789012	contacto@empresa.com	Zona Industrial #56
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Productos

```sql
USE inventario;
INSERT INTO Productos (nombre, descripcion, precio_base)
VALUES
('Laptop', 'Laptop de 14 pulgadas, 8GB RAM, 256GB SSD', 1200.00),
('Smartphone', 'Smartphone Android, 6GB RAM, 128GB', 600.00),
('Monitor', 'Monitor Full HD de 24 pulgadas', 250.00),
('Teclado', 'Teclado mecánico retroiluminado', 100.00);
```

Result Grid | Filter Rows: | Edit: | Export/Import:

producto_id	nombre	descripcion	precio_base
1	Laptop	Laptop de 14 pulgadas, 8GB RAM, 256GB SSD	1200.00
2	Smartphone	Smartphone Android, 6GB RAM, 128GB	600.00
3	Monitor	Monitor Full HD de 24 pulgadas	250.00
4	Teclado	Teclado mecánico retroiluminado	100.00
NULL	NULL	NULL	NULL

OrdenesCompra

```sql
USE inventario;
INSERT INTO OrdenesCompra (proveedor_id, fecha)
VALUES
(1, '2024-09-01'),
(2, '2024-09-02'),
(3, '2024-09-03'),
(4, '2024-09-04');
```

Result Grid | Filter Rows:

orden_id	proveedor_id	fecha
1	1	2024-09-01
2	2	2024-09-02
3	3	2024-09-03
4	4	2024-09-04
NULL	NULL	NULL

DetalleOrdenCompra

```sql
USE inventario;
INSERT INTO DetalleOrdenCompra (orden_id, producto_id, cantidad, precio_unitario)
VALUES
(1, 1, 5, 1150.00),
(1, 3, 2, 240.00),
(2, 2, 10, 580.00),
(3, 4, 3, 95.00);
```

detalle_id	orden_id	producto_id	cantidad	precio_unitario	Precio_total
1	1	1	5	1150.00	5750.00
2	1	3	2	240.00	480.00
3	2	2	10	580.00	5800.00
4	3	4	3	95.00	285.00
NULL	NULL	NULL	NULL	NULL	NULL

Obtener detalles de las órdenes de compra con información del proveedor y el producto donde la orden sea la 2:

```sql
#Obtener detalles de las órdenes de compra con información del proveedor
#y el producto donde la orden sea la 2
SELECT o.orden_id, p.nombres AS proveedor_nombre,
    p.identificacion AS proveedor_identificacion,
    pr.nombre AS producto_nombre,
    d.cantidad, d.precio_unitario, d.Precio_total
FROM OrdenesCompra o
INNER JOIN Proveedores p ON o.proveedor_id = p.proveedor_id
INNER JOIN DetalleOrdenCompra d ON o.orden_id = d.orden_id
INNER JOIN Productos pr ON d.producto_id = pr.producto_id
WHERE o.orden_id=2
```

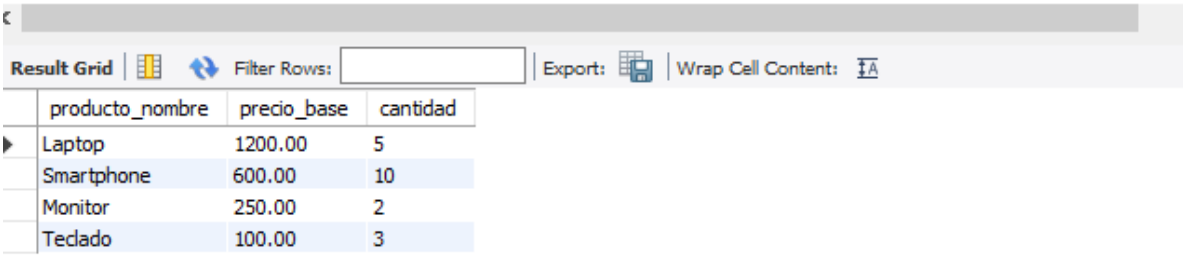
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

orden_id	proveedor_nombre	proveedor_identificacion	producto_nombre	cantidad	precio_unitario	Precio_total
2	María	987654321	Smartphone	10	580.00	5800.00

Listar los productos con sus precios base y la cantidad comprada en cada orden:

```sql
#Listar los productos con sus precios base y la cantidad comprada en cada orden:
SELECT pr.nombre AS producto_nombre, pr.precio_base, d.cantidad
FROM Productos pr
INNER JOIN DetalleOrdenCompra d ON pr.producto_id = d.producto_id;

```



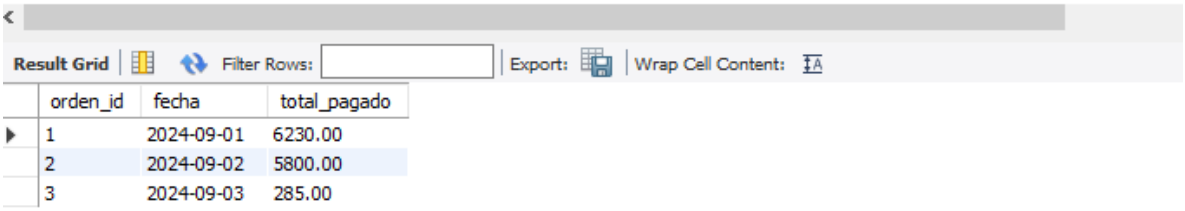
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

producto_nombre	precio_base	cantidad
Laptop	1200.00	5
Smartphone	600.00	10
Monitor	250.00	2
Teclado	100.00	3

Obtener la información de las órdenes de compra junto con el total pagado en cada una:

```sql
#Obtener la información de las órdenes de compra junto con el total pagado en cada una:
SELECT o.orden_id, o.fecha,
    SUM(d.Precio_total) AS total_pagado
FROM OrdenesCompra o
INNER JOIN DetalleOrdenCompra d ON o.orden_id = d.orden_id
GROUP BY o.orden_id, o.fecha;

```



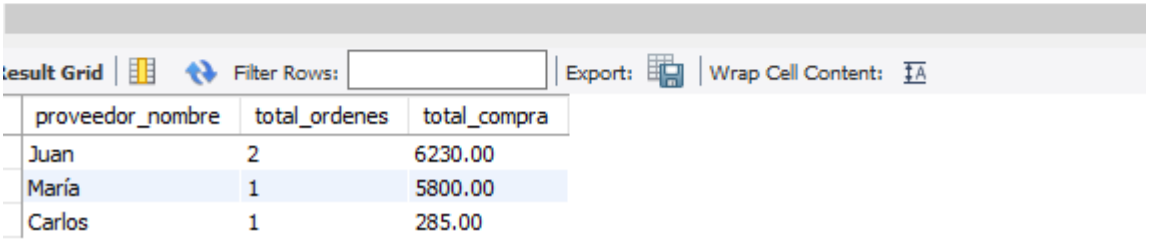
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

orden_id	fecha	total_pagado
1	2024-09-01	6230.00
2	2024-09-02	5800.00
3	2024-09-03	285.00

Contar cuantas ordenes un proveedor ha generado y sumar el precio total, agruparlo por la id del proveedor

```sql
#Contar cuantas ordenes un proveedor ha generado y sumar el precio total,
#agruparlo por la id del proveedor
SELECT p.nombres AS proveedor_nombre,
    COUNT(o.orden_id) AS total_ordenes,
    SUM(d.Precio_total) AS total_compra
FROM Proveedores p
INNER JOIN OrdenesCompra o ON p.proveedor_id = o.proveedor_id
INNER JOIN DetalleOrdenCompra d ON o.orden_id = d.orden_id
GROUP BY p.proveedor_id;

```



proveedor_nombre	total_ordenes	total_compra
Juan	2	6230.00
María	1	5800.00
Carlos	1	285.00

ANÁLISIS Y DISCUSIÓN

El propósito de generar integridad y eficiencia se demuestra con esta base de datos donde su diseño muestra una estructura robusta y eficiente para el manejo de las relaciones entre proveedores, productos y órdenes de compra. Dando así la recuperación oportuna de detalles mediante una consulta de las órdenes de compra con la información de los proveedores y productos muestra que el sistema permite un control eficiente de inventarios y compras para cualquier tienda. Esto asegura que las entidades y atributos están debidamente relacionados y que el flujo de datos es consistente con los objetivos planteados.

Además, los resultados permiten identificar factores importantes en la relación entre proveedores y productos. Por ejemplo, al analizar qué proveedores suministran múltiples productos o cuáles son los productos más adquiridos, se logra cumplir con el objetivo de optimizar la gestión de inventarios y relaciones comerciales.

Ahora bien, la gestión de proveedores nos permite hacer un análisis de las órdenes de compra detalladas para realizar una mejor planificación y gestión de las relaciones con los proveedores. Los datos obtenidos brindan una perspectiva clara sobre el rendimiento

Es por esto que los resultados se alinean con el objetivo de crear un sistema que mejore la administración del inventario, ayudando a tomar decisiones basadas en los datos recolectados de las operaciones diarias.

CONCLUSIONES

Este ejercicio permitió mejorar la eficiencia operativa en la implementación del sistema de base de datos logrando optimizar la gestión de productos y proveedores, permitiendo un manejo más estructurado y preciso de las órdenes de compra y sus detalles.

Es posible garantizar escalabilidad y flexibilidad para mantener el diseño relacional permitiendo asegurar que la base de datos pueda escalar a medida que aumenten los datos de proveedores, productos y órdenes de compra, sin perder integridad ni eficiencia.

Para concluir, puedo afirmar que los conocimientos adquiridos por medio de la realización de este pequeño modelo de base de datos me brindan una amplia posibilidad de analizar ambientes donde se pueda implementar esta relación de compras y se desee llevar un inventario.

RECOMENDACIONES

En el presente documento se estructura la implementación de una pequeña base de datos que permite gestionar de un inventario los productos, proveedores y órdenes de compra. Donde se estructura una metodología para su desarrollo eficiente, de esta manera se plantea la creación y distinción de las diferentes entidades que harán para de esta base de datos, así tenemos entidades fuertes y débiles, en cuanto a sus atributos se estableció un atributo derivado.

Continuando se creo la base de datos por medio de código SQL en MySQL worbench y sus respectivos modelos entidad relación.

Dejando la base de datos llena con registros en cada entidad se procede hacer diferentes consultas por ejemplo con INNER JOIN; GROUP BY.

REFERENCIAS

https://lucid.app/documents#/home?folder_id=recent

https://www.youtube.com/watch?v=TKuxYHb-Hvc

https://miro.com/es/diagrama/como-hacer-diagrama-entidad-relacion/

draw.io= https://app.diagrams.net/?src=about