Creación y Gestión de Base de Datos para una Tienda Virtual en MySQL

Presentado por:

Carlos Andrés Montoya Hoyos

Presentado A:

Brayan Arcos Burbano

Instituto Tecnológico del Putumayo

Desarrollo de Base de Datos

2024-09-11

Índice

[1. Resumen Ejecutivo 3](#_Toc177160022)

[2. Introducción 4](#_Toc177160023)

[Contexto y Motivación 4](#_Toc177160024)

[Alcance del Informe 4](#_Toc177160025)

[Objetivos 4](#_Toc177160026)

[3. Metodología 5](#_Toc177160027)

[Herramientas Utilizadas 5](#_Toc177160028)

[Procedimientos 5](#_Toc177160029)

[4. Desarrollo del Informe 6](#_Toc177160030)

[Descripción de la Base de Datos 6](#_Toc177160031)

[Tablas de la Base de Datos: 11](#_Toc177160032)

[Normalización 13](#_Toc177160033)

[Cardinalidad 14](#_Toc177160034)

[5. Análisis y Discusión 16](#_Toc177160035)

[6. Conclusiones 17](#_Toc177160036)

[7. Recomendaciones 18](#_Toc177160037)

[8. Referencias 19](#_Toc177160038)

# Resumen Ejecutivo

Este informe detalla la creación y gestión de una base de datos en MySQL para una tienda virtual. Se describen las tablas diseñadas para gestionar información clave, como compradores, productos, proveedores y categorías, además de las relaciones entre ellas. El informe también incluye ejemplos de consultas SQL que permiten realizar operaciones comunes en una tienda virtual, como la inserción, actualización y eliminación de registros, y la obtención de datos relevantes.

# Introducción

## Contexto y Motivación

El propósito de este informe es desarrollar una base de datos funcional para una tienda virtual, donde se gestionen de manera eficiente los datos relacionados con los compradores, productos, proveedores y las transacciones de compra. Las bases de datos son esenciales para el funcionamiento de las tiendas virtuales, ya que permiten organizar la información de manera estructurada y accesible.

## Alcance del Informe

Este informe abarca los siguientes aspectos clave del desarrollo de la base de datos:

* Creación y estructura de tablas necesarias para una tienda virtual.
* Inserción, modificación y eliminación de registros.
* Consultas SQL que permiten gestionar y analizar la información.
* Diseño de la base de datos mediante la definición de relaciones y claves foráneas.

## Objetivos

El objetivo principal es diseñar y gestionar una base de datos que sirva como pilar para el funcionamiento de una tienda virtual. Esto incluye la capacidad de gestionar productos, categorías, compradores y proveedores, asegurando la integridad y eficiencia de la base de datos.

# Metodología

## Herramientas Utilizadas

* **MySQL Workbench:** Utilizado para diseñar, crear y gestionar la base de datos.
* **MySQL Server:** Plataforma que gestiona la base de datos.
* **Diagrama Entidad-Relación (ERD):** Herramienta visual para diseñar la estructura de la base de datos y las relaciones entre las tablas.

## Procedimientos

1. **Creación de la base de datos:** Se inició creando las tablas necesarias para gestionar productos, categorías, compradores y proveedores.
2. **Inserción, actualización y eliminación de datos:** Se realizaron operaciones CRUD para gestionar los datos de la tienda.
3. **Modificación de la estructura de las tablas:** Se aplicaron alteraciones a las tablas para agregar o eliminar campos conforme a las necesidades del negocio.
4. **Consultas SQL para obtener información relevante:** Se realizaron consultas para extraer información sobre compradores, productos, categorías y proveedores.

# 4. Desarrollo del Informe

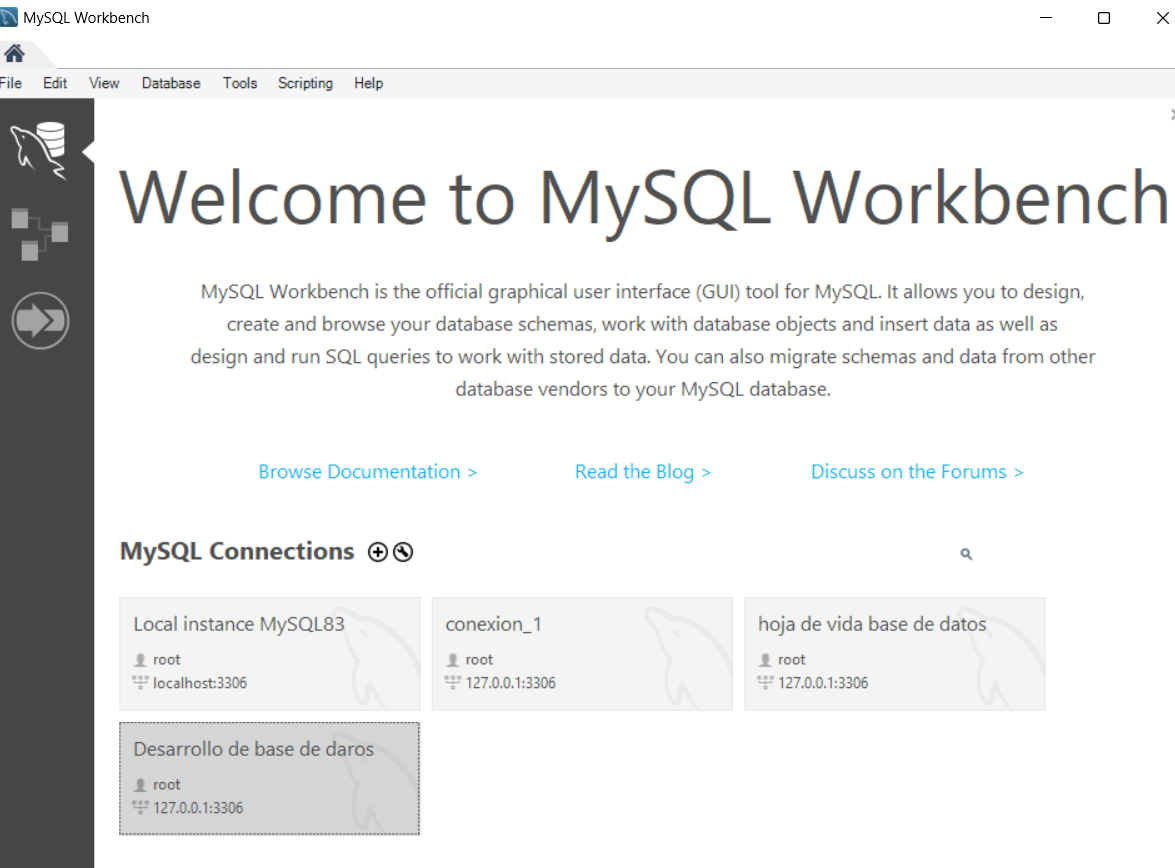
## Descripción de la Base de Datos

Cree una base de datos de prueba

* Crear una tabla Compradores con atributos (id, nombre, nit).
* Insertar, actualizar, eliminar, obtener valores a esos atributos.
* Modificar, agregar, eliminar campos de la tabla.

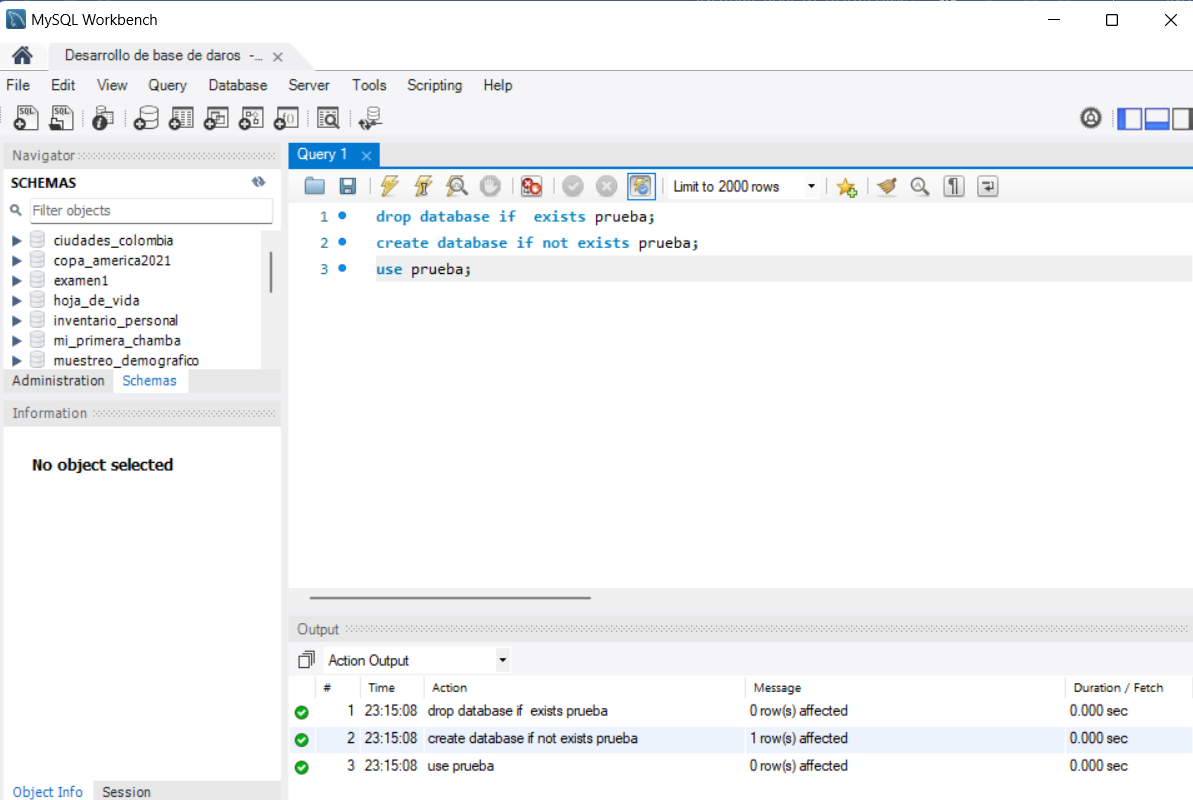
**Desarrollo**

Lo primero que realice fue la instalación y una conexión en MySQL Workbench donde voy a realizar la base de datos.



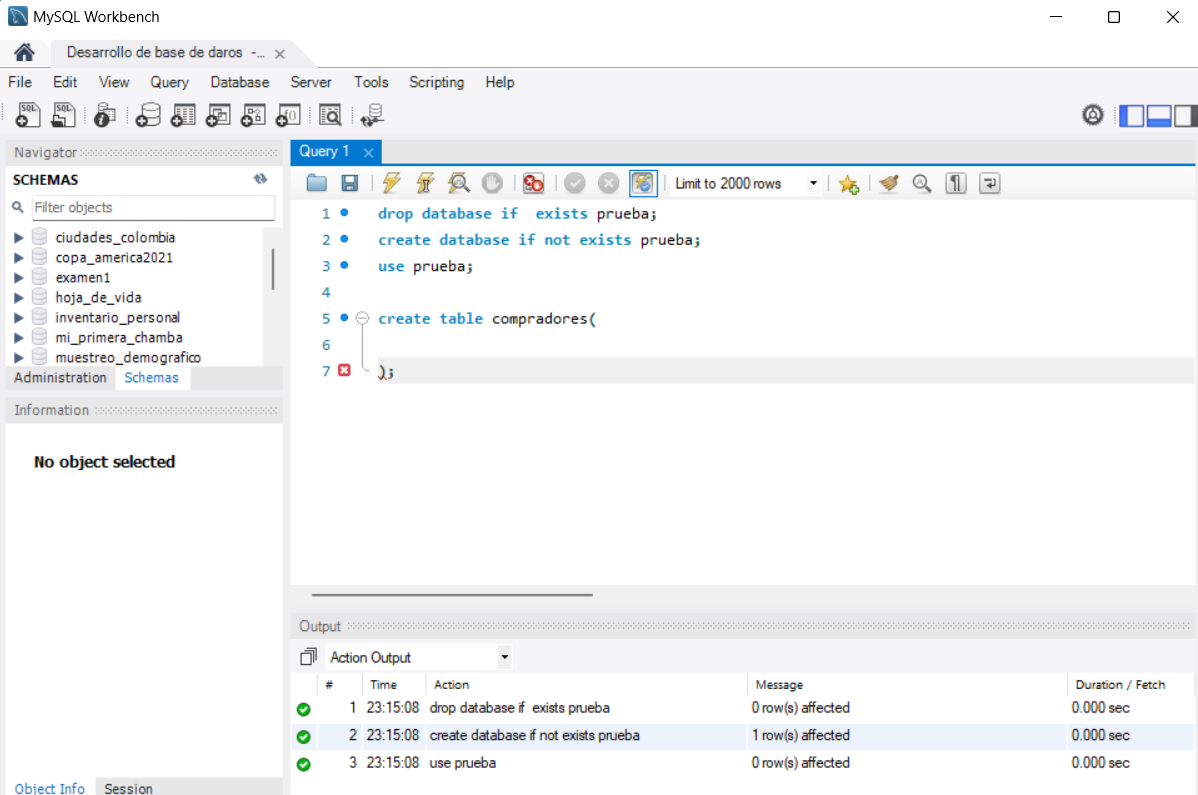
**Creación de la Tabla:**

En una pestaña SQL en blanco se escribe el siguiente código SQL para crear la base de datos:



realice la ejecución del código donde se confirma que la base de datos fue creada exitosamente.

Se crea la tabla de compradores con los siguientes datos:



Luego se agregan los atributos requeridos especificando el tipo de dato en cada uno, así:

* id int auto\_increment primary key:

id: Nombre de la columna.

int: Tipo de dato entero.

auto\_increment: Indica que el valor de esta columna se incrementará automáticamente con cada nuevo registro.

primary key: Define esta columna como la clave primaria, lo que significa que cada valor en esta columna será único y no nulo.

* nombre varchar (60):

nombre: Nombre de la columna.

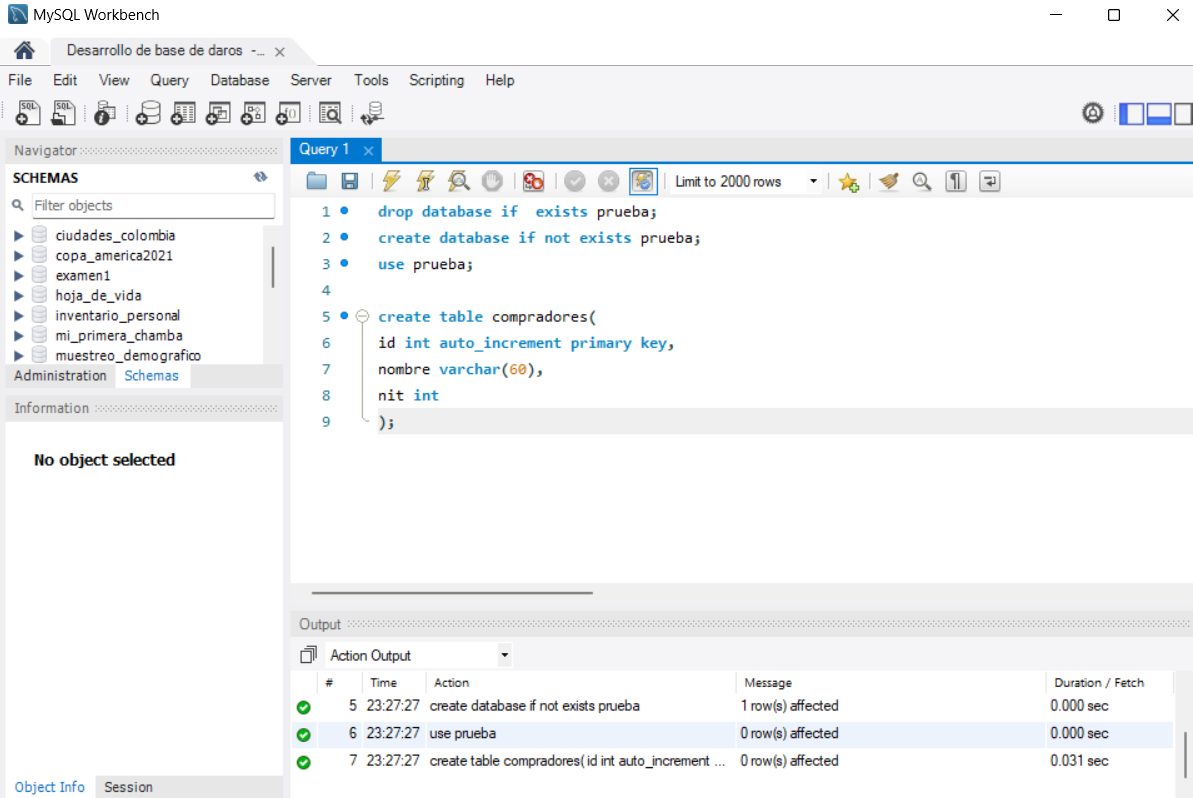
varchar (60): Tipo de dato de cadena de caracteres con una longitud máxima de 60 caracteres.

* nit int:

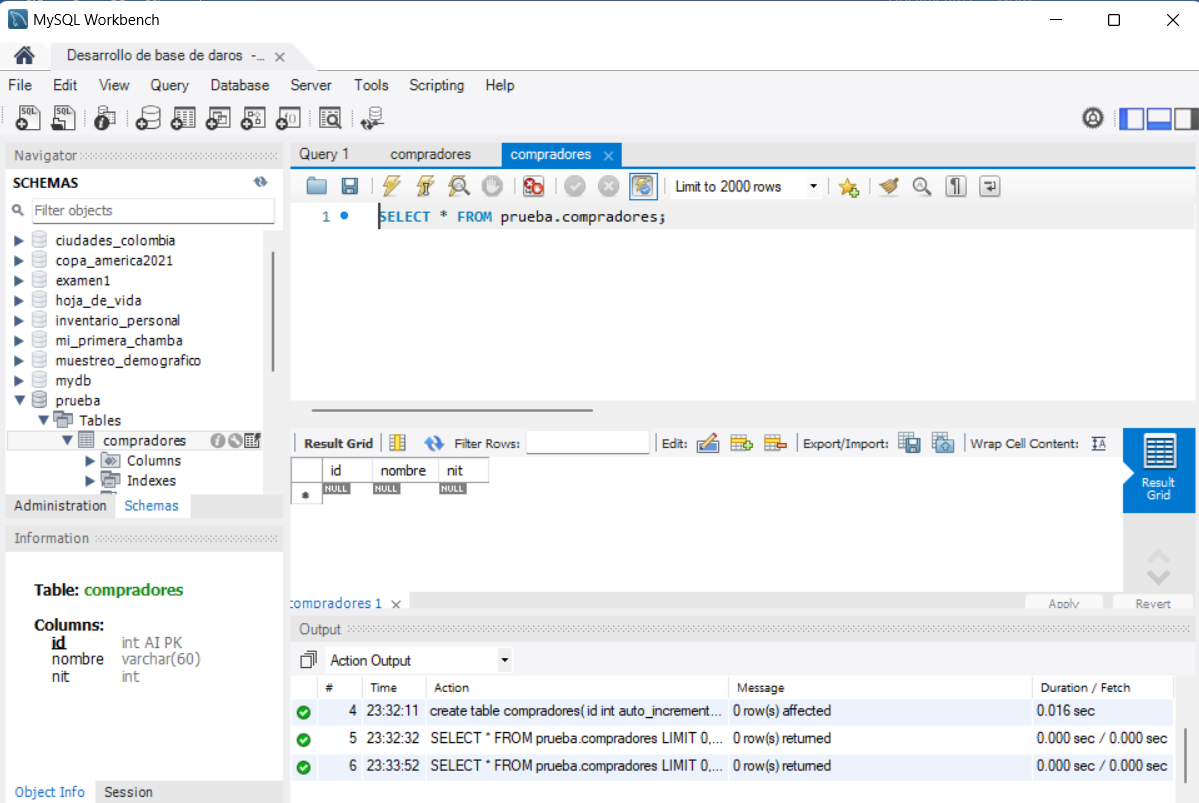
nit: Nombre de la columna.

int: Tipo de dato entero.

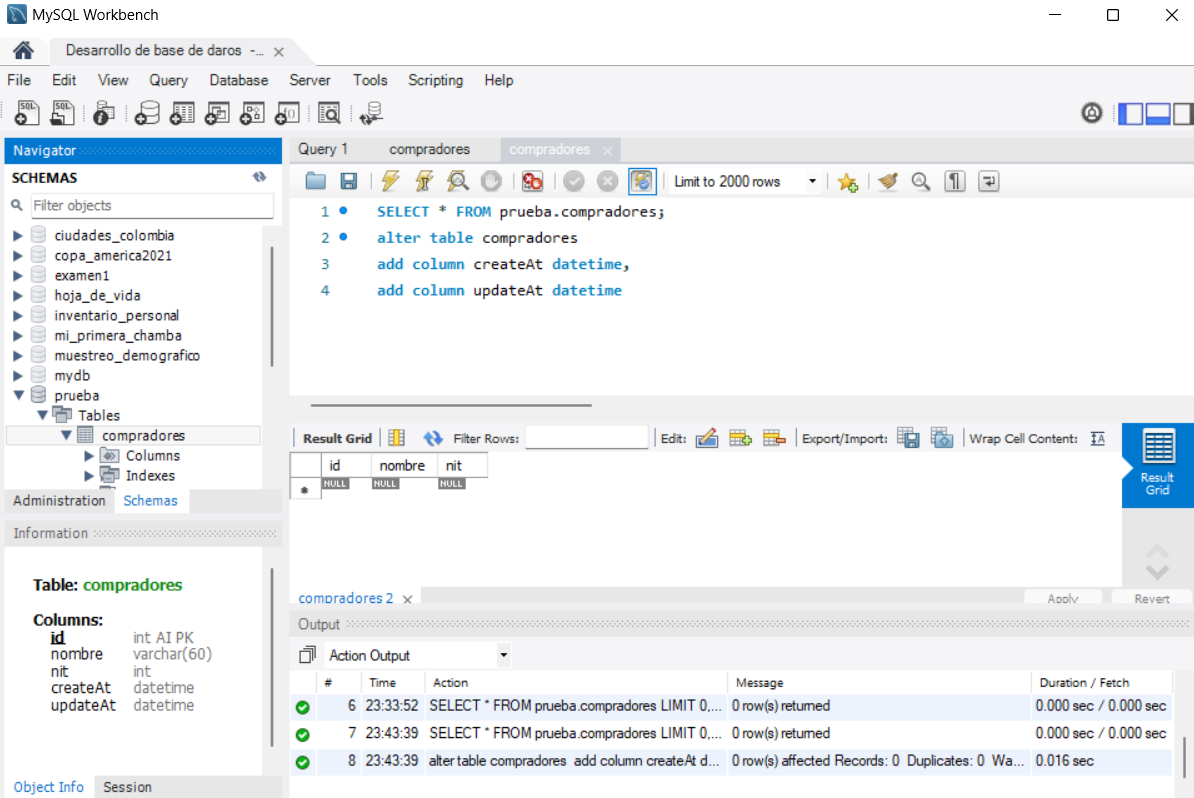
Se actualiza y encontramos la base de datos creada.



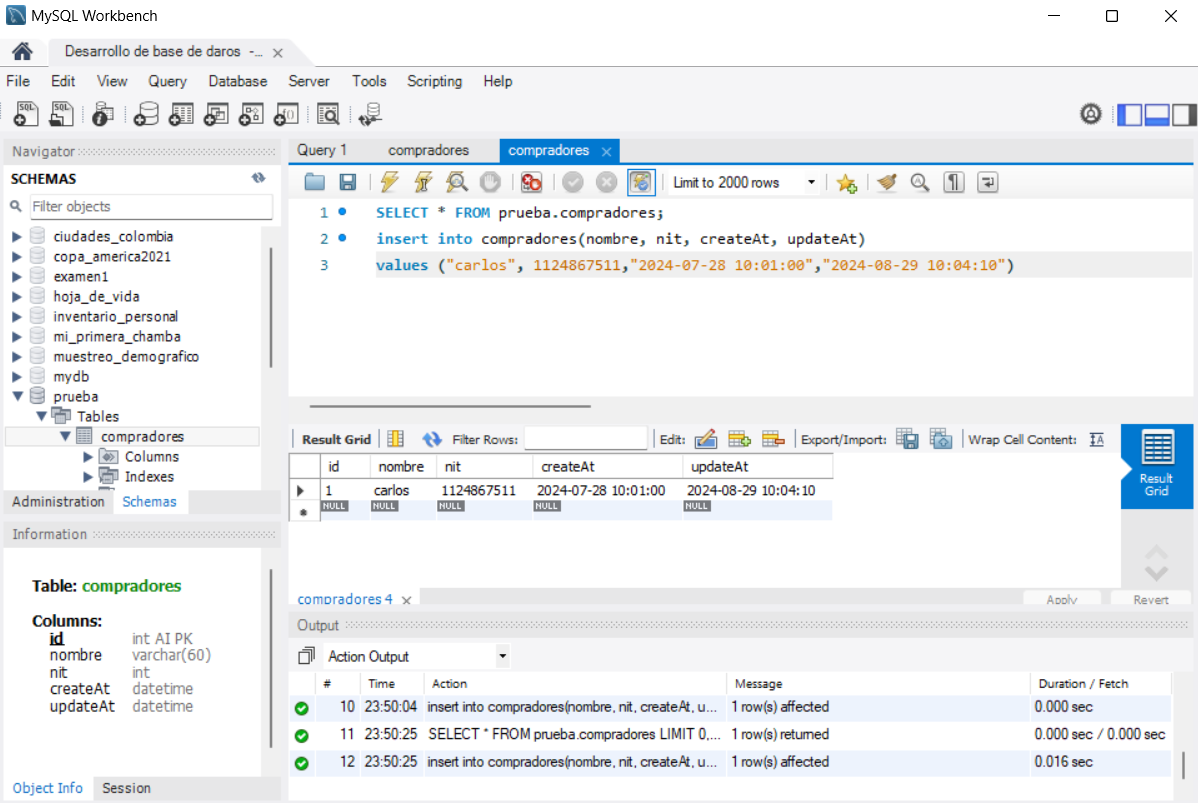
Procedemos a actualizar la base de datos y Al dar clic derecho en la tabla usuarios se puede ingresar a la inspección de esta:



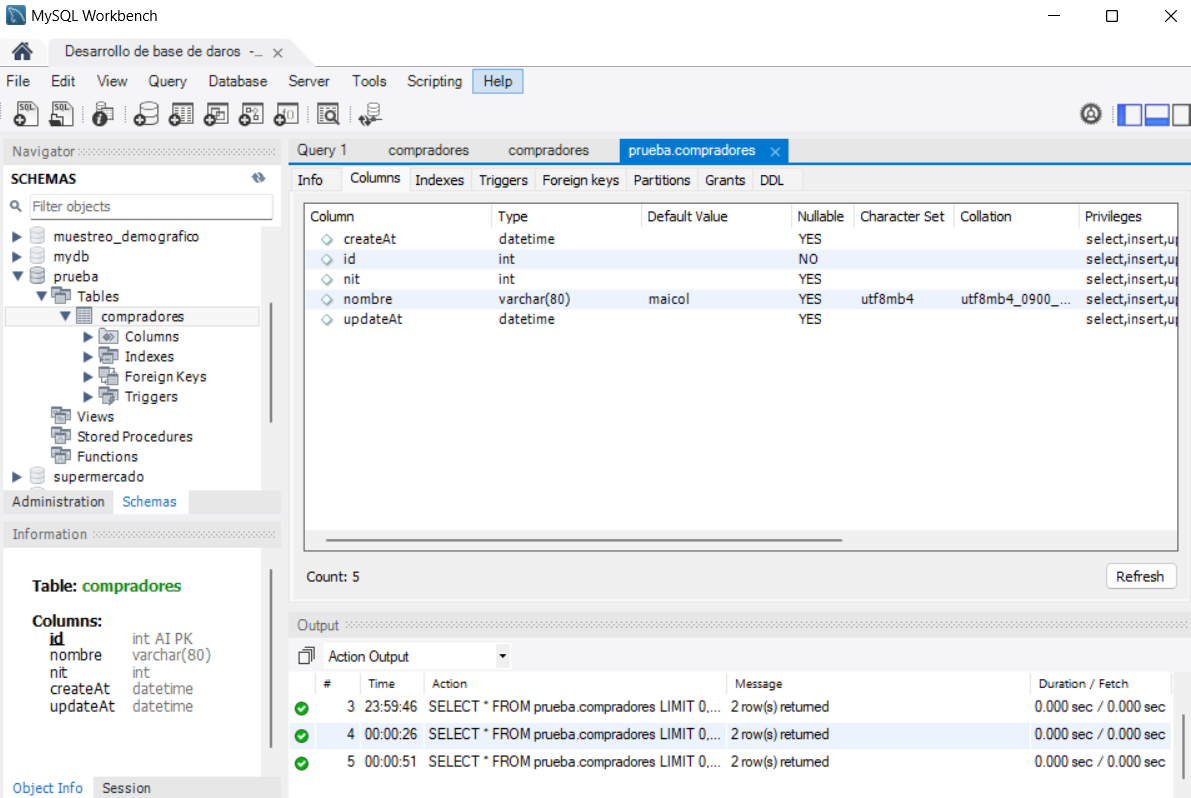
Procedemos a agregar las columnas createAt(fecha de creación) y updateAt(fecha de actualización), que son necesarias en la tabla compradores, utilizando el comando altert y add. Y actualizamos.



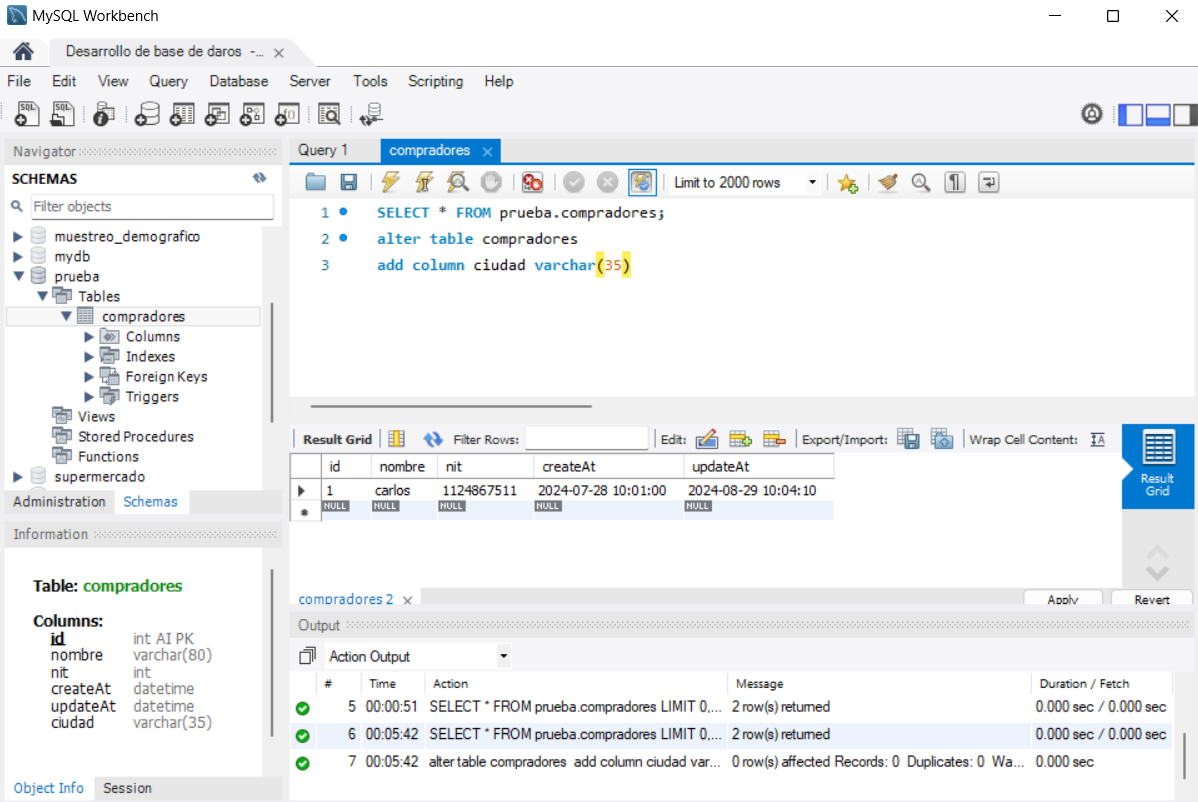
Agregamos los valores en los campos de la tabla compradores utilizando el comando **insert into**



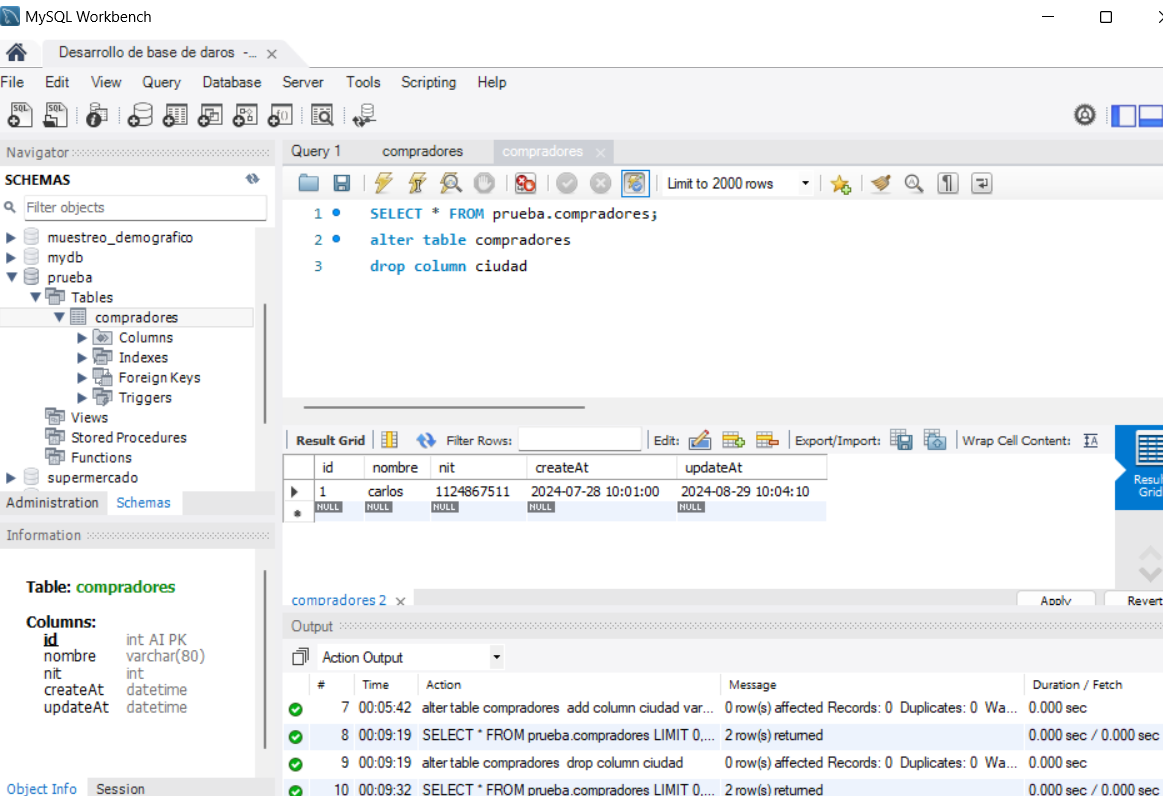
Ahora modificamos los valores de la columna nombre de la tabla compradores.



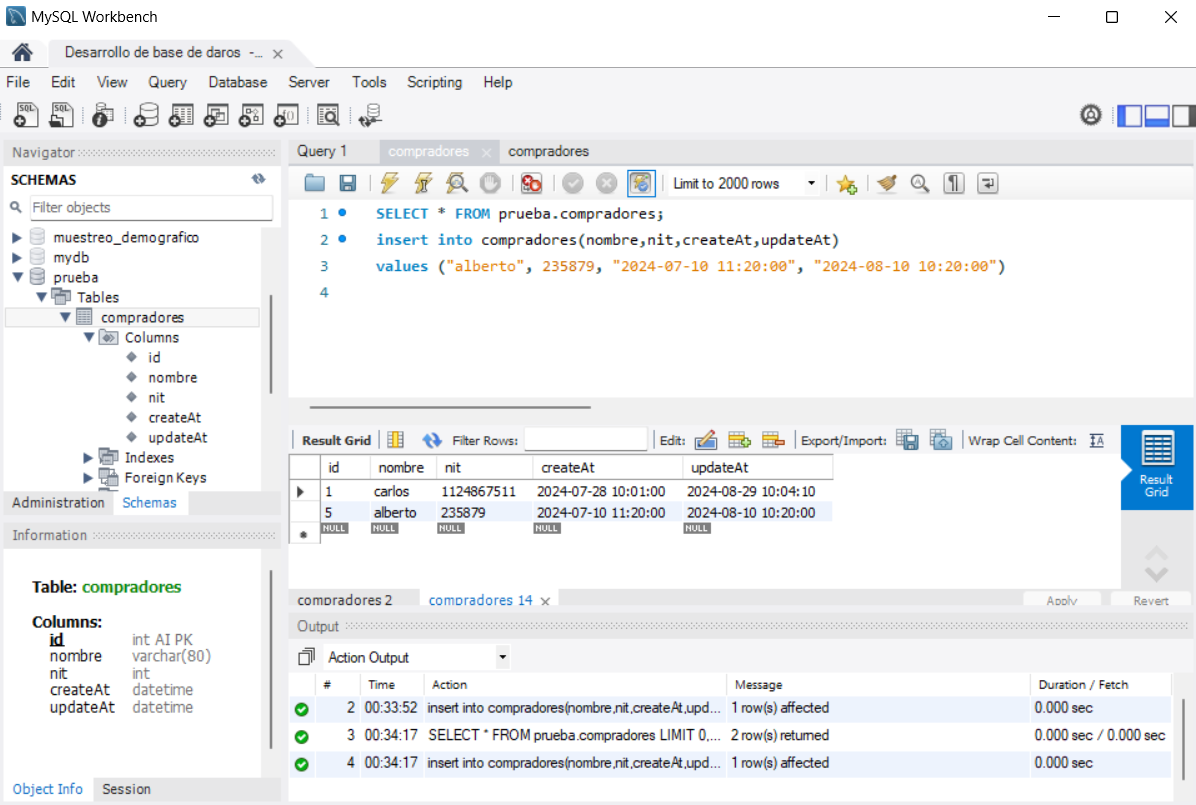
agregamos la columna ciudad a la tabla compradores.



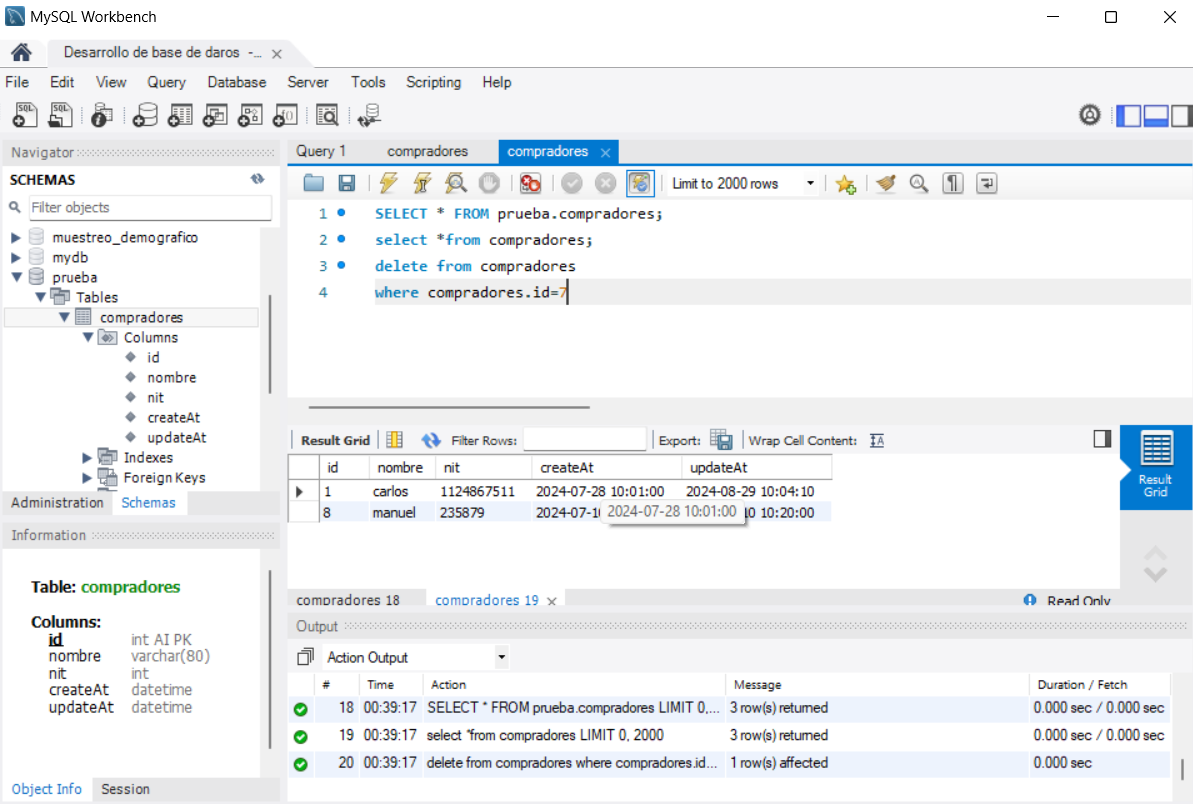
Ahora procederemos a eliminar esta columna la cual no tendrá uso entonces se eliminará usando el comando drop



Se realiza el ejercicio de obtener registros de la tabla usuarios. Ya se ha ingresado un registro en la columna nombre. Se agregará otro y se lo obtendrá



Procedemos a eliminar el registro que estaba antes de actualizar, usando el comando delete from



Luego realice la creación de nuevas tablas:

### Tablas de la Base de Datos:

1. **Tabla ciudad: (fuerte)**

* **Descripción**: Almacena la información de las ciudades donde residen los compradores o proveedores.
* **Columnas**:
  + - id\_ciudad: Entero, clave primaria, autoincremental.
    - ciudad: Nombre de la ciudad, tipo VARCHAR(50).
    - descripcion: Breve descripción de la ciudad, tipo TINYTEXT.

1. **Tabla compradores: (fuerte)**

* **Descripción**: Almacena información de los compradores, incluyendo su nombre, apellidos, identificación fiscal (NIT) y la ciudad donde residen.
* **Columnas**:
  + - id\_compradores: Entero, clave primaria, autoincremental.
    - nombres: Nombre del comprador, tipo VARCHAR(50).
    - apellido1: Primer apellido del comprador, tipo VARCHAR(50).
    - apellido2: Segundo apellido del comprador, tipo VARCHAR(50).
    - nit: Número de identificación fiscal, tipo VARCHAR(15).
    - createAt: Fecha y hora de creación del registro, tipo DATETIME.
    - updateAt: Fecha y hora de última actualización del registro, tipo DATETIME.
    - id\_ciudad: Clave foránea que referencia la tabla ciudad.

1. **Tabla categoria\_product: (fuerte)**

* **Descripción**: Define las diferentes categorías en las que se pueden clasificar los productos.
* **Columnas**:
  + - id\_categoria\_producto: Entero, clave primaria, autoincremental.
    - categoria: Nombre de la categoría del producto, tipo VARCHAR(35).

1. **Tabla producto:(fuerte)**

* **Descripción**: Contiene información sobre los productos ofrecidos, incluyendo su descripción, categoría y precio.
* **Columnas**:
  + - id\_producto: Entero, clave primaria, autoincremental.
    - producto: Nombre del producto, tipo VARCHAR(100).
    - descripcion: Descripción detallada del producto, tipo TEXT.
    - categoria\_producto: Clave foránea que referencia la tabla categoria\_product.
    - precio: Precio del producto, tipo DECIMAL(10,2).

1. **Tabla provedores: (fuerte)**

* **Descripción**: Almacena información de los proveedores, incluyendo su nombre, contacto y ciudad.
* **Columnas**:
  + - id\_provedores: Entero, clave primaria, autoincremental.
    - nombre: Nombre del proveedor, tipo VARCHAR(100).
    - contacto: Información de contacto del proveedor, tipo VARCHAR(100).
    - telefono: Teléfono de contacto, tipo VARCHAR(15).
    - email: Correo electrónico del proveedor, tipo VARCHAR(100).
    - direccion: Dirección del proveedor, tipo VARCHAR(80).
    - id\_ciudad: Clave foránea que referencia la tabla ciudad.

1. **Tabla producto\_proveedor:(débil)**

* **Descripción**: Tabla de relación que asocia productos con proveedores, permitiendo registrar qué productos suministra cada proveedor.
* **Columnas**:
  + id\_producto: Clave foránea que referencia la tabla producto.
    - id\_provedores: Clave foránea que referencia la tabla provedores.
    - **Clave primaria compuesta**: id\_producto, id\_provedores.

### Normalización

**Primera Forma Normal (1NF)**

Para cumplir con la 1NF, todos los valores en cada tabla deben ser atómicos (sin listas o conjuntos dentro de una celda) y cada registro debe ser único.

* **Tabla ciudad** ya está en 1NF. Todas las columnas contienen valores atómicos y tiene una clave primaria.
* **Tabla compradores** está en 1NF ya que los nombres, apellidos, NIT y demás atributos son valores atómicos, y tiene una clave primaria.
* **Tabla categoria\_product** está en 1NF, ya que contiene valores atómicos en cada columna.
* **Tabla producto** está en 1NF, ya que todos los atributos (nombre, descripción, categoría y precio) son valores atómicos.
* **Tabla provedores** está en 1NF. Todos los valores son atómicos y tiene clave primaria.
* **Tabla producto\_proveedor** también está en 1NF, ya que utiliza claves foráneas compuestas y no contiene valores repetidos o no atómicos.

**Segunda Forma Normal (2NF)**

Para cumplir la 2NF, la tabla debe estar en 1NF y no debe haber dependencia parcial, es decir, no deben existir atributos que dependan solo de una parte de la clave primaria si esta es compuesta.

* **Tabla ciudad** está en 2NF ya que tiene clave primaria simple y todas las columnas dependen de esta clave.
* **Tabla compradores** también está en 2NF ya que todos los atributos dependen de la clave primaria id\_compradores.
* **Tabla categoria\_product** está en 2NF, ya que todos los atributos dependen de su clave primaria.
* **Tabla producto** cumple con la 2NF ya que cada atributo depende completamente de la clave primaria id\_producto.
* **Tabla provedores** está en 2NF ya que todos los atributos dependen de su clave primaria id\_provedores.
* **Tabla producto\_proveedor** también cumple con la 2NF ya que los atributos (claves foráneas) dependen de la clave primaria compuesta (id\_producto, id\_provedores).

**Tercera Forma Normal (3NF)**

Para cumplir con la 3NF, una tabla debe estar en 2NF y todos los atributos no clave deben depender exclusivamente de la clave primaria. No debe haber dependencias transitivas.

* **Tabla ciudad** está en 3NF, ya que no hay dependencias transitivas.
* **Tabla compradores** está en 3NF. Los atributos como nombre, apellidos y NIT dependen solo de la clave primaria id\_compradores, sin dependencias transitivas.
* **Tabla categoria\_product** cumple con la 3NF ya que la única columna es la clave primaria y la categoría.
* **Tabla producto** está en 3NF. Todos los atributos dependen de la clave primaria id\_producto, sin dependencias transitivas.
* **Tabla provedores** está en 3NF ya que los atributos dependen solo de id\_provedores.
* **Tabla producto\_proveedor** está en 3NF ya que los atributos (claves foráneas) dependen exclusivamente de la clave primaria compuesta.

### Cardinalidad

La **cardinalidad** describe la relación entre las tablas.

1. **Relación Ciudad - Compradores (1):**

* Una ciudad puede tener múltiples compradores (uno a muchos).
* La clave foránea id\_ciudad en la tabla compradores indica esta relación.

1. **Relación Ciudad - Proveedores (1):**

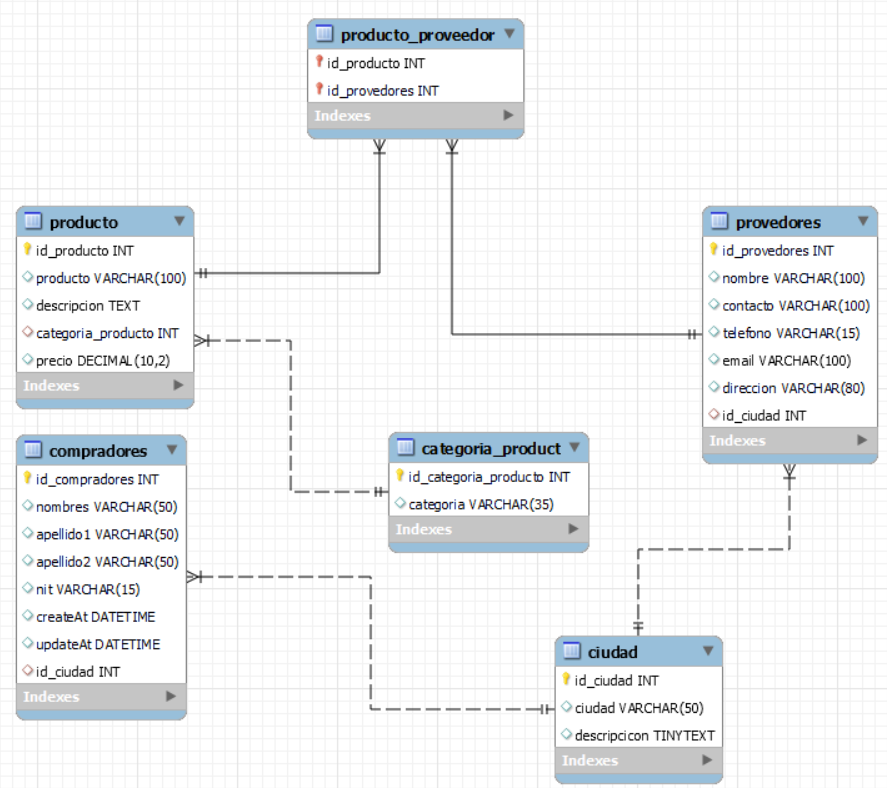
* Una ciudad puede tener múltiples proveedores.
* La clave foránea id\_ciudad en la tabla provedores refleja esta relación.

1. **Relación Categoría Producto - Producto (1):**

* Una categoría puede tener muchos productos (uno a muchos).
* La clave foránea categoria\_producto en la tabla producto representa esta relación.

1. **Relación Producto - Proveedores (N):**

* Un producto puede ser ofrecido por múltiples proveedores y un proveedor puede ofrecer múltiples productos (muchos a muchos).
* Esto se maneja mediante la tabla intermedia **producto\_proveedor**, que tiene una relación de muchos a muchos.



# Análisis y Discusión

El diseño de la base de datos permite gestionar de manera eficiente los elementos clave de una tienda virtual, como el inventario de productos, los proveedores y los compradores. El uso de consultas SQL asegura que la información esté actualizada y disponible para su análisis y toma de decisiones.

# Conclusiones

El desarrollo de esta base de datos demuestra cómo una tienda virtual puede beneficiarse de una estructura organizada que permita gestionar de forma eficaz las operaciones diarias, como la gestión de inventario y la relación con proveedores y clientes.

# Recomendaciones

1. Implementar índices para mejorar el rendimiento de las consultas sobre grandes volúmenes de datos.
2. Agregar triggers para gestionar automáticamente la actualización de ciertos campos, como las fechas de creación y modificación.
3. Utilizar procedimientos almacenados para automatizar procesos como el registro de compras o la actualización de inventario.

# Referencias

* [Documentación oficial de MySQL](https://dev.mysql.com/doc/)
* W3Schools: SQL Tutorial