# **Taller Bases de Datos Relacionales**

Juan Pablo Guzmán Martínez  
ID: 1036822

Corporación Universitaria Minuto de Dios  
Facultas Ingeniería de Sistemas  
NRC-10-73461

Bogotá, Colombia  
02 de Marzo de 2025

# **Introducción**

Este documento ayuda a presentar como es el desarrollo de una base de datos relacional, donde se incluyen su diagrama relacional, los scripts necesarios para su implementación y una explicación detallada de cada una de las realizaciones. El objetivo principal es poder demostrar el diseño y funcionamiento de una base de datos que es optimizada para su uso en entornos reales o en el mundo real. A lo largo del documento se muestran las relaciones entre las entidades, las estructuras de las tablas y los procedimientos necesarios para poder gestionar la información. Finalmente, se presentan las conclusiones basadas en la experiencia obtenida durante el desarrollo de este trabajo.

# **Tabla de contenido**

[**Taller Bases de Datos Relacionales** 1](#_Toc191844699)

[**Introducción** 2](#_Toc191844700)

[**Tabla de contenido** 3](#_Toc191844701)

[**Desarrollo** 4](#_Toc191844702)

[Diagrama Relacional de las diferentes bases de datos relacionales solicitadas. 4](#_Toc191844703)

[Scripts solicitados y necesarios para la defensa de la entrega. 5](#_Toc191844704)

[Script de mi base de datos db\_municipios: 5](#_Toc191844705)

[Script de mi base de datos databaseJumbo: 10](#_Toc191844706)

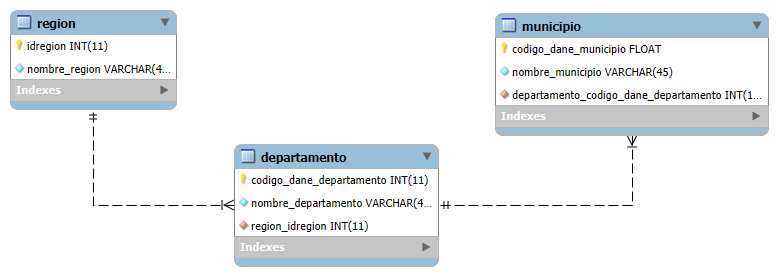
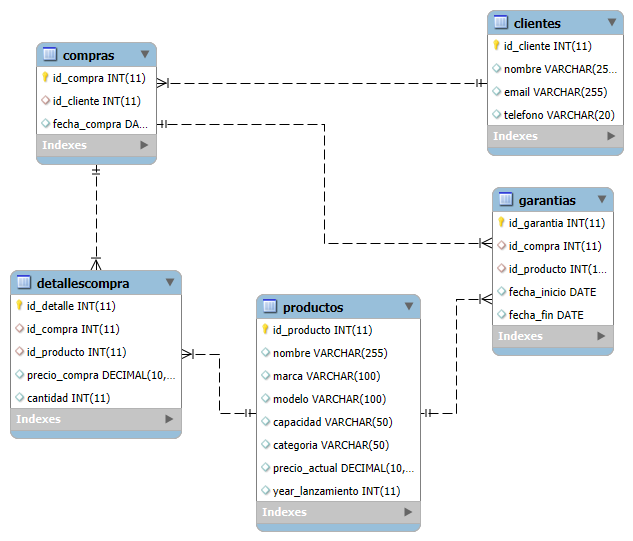
[Descripciones o explicaciones de los scripts solicitados. 13](#_Toc191844707)

[Conclusiones Scripts Load Data Infile 17](#_Toc191844708)

[Conclusiones 19](#_Toc191844709)

# **Desarrollo**

## Diagrama Relacional de las diferentes bases de datos relacionales solicitadas.

Base de Datos Relacional regiones, departamentos y municipios de Colombia:  
  
Base de Datos Relacional Procesos de Facturación:

## Scripts solicitados y necesarios para la defensa de la entrega.

### Script de mi base de datos db\_municipios:

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema fabrica

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema fabrica

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `fabrica` DEFAULT CHARACTER SET utf8mb4 ;

-- -----------------------------------------------------

-- Schema db\_municipios

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema db\_municipios

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `db\_municipios` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_spanish\_ci ;

USE `fabrica` ;

-- -----------------------------------------------------

-- Table `fabrica`.`instructores`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `fabrica`.`instructores` (

  `IdInstructor` INT(11) NOT NULL AUTO\_INCREMENT,

  `NombreInstructor` VARCHAR(255) NULL DEFAULT NULL,

  `ApellidoInstructor` VARCHAR(255) NULL DEFAULT NULL,

  `TipoIdentificacion` ENUM('CC', 'TI') NULL DEFAULT NULL,

  `NumeroIdentificacion` INT(11) NULL DEFAULT NULL,

  `CorreoInstructor` VARCHAR(255) NULL DEFAULT NULL,

  `CelularInstructor` INT(11) NULL DEFAULT NULL,

  PRIMARY KEY (`IdInstructor`))

ENGINE = InnoDB

AUTO\_INCREMENT = 3

DEFAULT CHARACTER SET = utf8mb4;

-- -----------------------------------------------------

-- Table `fabrica`.`productosgenerales`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `fabrica`.`productosgenerales` (

  `IdProducto` INT(11) NOT NULL AUTO\_INCREMENT,

  `NombreProducto` VARCHAR(225) NULL DEFAULT NULL,

  `DescripcionProducto` VARCHAR(225) NULL DEFAULT NULL,

  `TipoProducto` VARCHAR(20) NULL DEFAULT NULL,

  `CantidadProducto` INT(11) NULL DEFAULT NULL,

  `ObservacionesProducto` VARCHAR(225) NULL DEFAULT NULL,

  PRIMARY KEY (`IdProducto`))

ENGINE = InnoDB

AUTO\_INCREMENT = 6

DEFAULT CHARACTER SET = utf8mb4;

-- -----------------------------------------------------

-- Table `fabrica`.`prestamos`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `fabrica`.`prestamos` (

  `IdPrestamo` INT(11) NOT NULL AUTO\_INCREMENT,

  `IdInstructor` INT(11) NULL DEFAULT NULL,

  `IdProducto` INT(11) NULL DEFAULT NULL,

  `FechaHoraPrestamo` DATETIME NULL DEFAULT NULL,

  `CantidadPrestamo` INT(11) NULL DEFAULT NULL,

  `EstadoPrestamo` ENUM('En curso', 'Culminados') NULL DEFAULT NULL,

  `ObservacionesPrestamo` VARCHAR(225) NULL DEFAULT NULL,

  PRIMARY KEY (`IdPrestamo`),

  INDEX `IdInstructor` (`IdInstructor` ASC) VISIBLE,

  INDEX `IdProducto` (`IdProducto` ASC) VISIBLE,

  CONSTRAINT `prestamos\_ibfk\_1`

    FOREIGN KEY (`IdInstructor`)

    REFERENCES `fabrica`.`instructores` (`IdInstructor`),

  CONSTRAINT `prestamos\_ibfk\_2`

    FOREIGN KEY (`IdProducto`)

    REFERENCES `fabrica`.`productosgenerales` (`IdProducto`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4;

-- -----------------------------------------------------

-- Table `fabrica`.`devoluciones`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `fabrica`.`devoluciones` (

  `IdDevoluciones` INT(11) NOT NULL AUTO\_INCREMENT,

  `IdInstructor` INT(11) NULL DEFAULT NULL,

  `IdPrestamo` INT(11) NULL DEFAULT NULL,

  `IdProducto` INT(11) NULL DEFAULT NULL,

  `FechaHoraDevolucion` DATETIME NULL DEFAULT NULL,

  `EstadoDevolucion` ENUM('Bueno', 'Mal Estado') NULL DEFAULT NULL,

  `Observaciones` VARCHAR(225) NULL DEFAULT NULL,

  `ModoTiempoLugar` VARCHAR(225) NULL DEFAULT NULL,

  PRIMARY KEY (`IdDevoluciones`),

  INDEX `IdInstructor` (`IdInstructor` ASC) VISIBLE,

  INDEX `IdPrestamo` (`IdPrestamo` ASC) VISIBLE,

  INDEX `IdProducto` (`IdProducto` ASC) VISIBLE,

  CONSTRAINT `devoluciones\_ibfk\_1`

    FOREIGN KEY (`IdInstructor`)

    REFERENCES `fabrica`.`instructores` (`IdInstructor`),

  CONSTRAINT `devoluciones\_ibfk\_2`

    FOREIGN KEY (`IdPrestamo`)

    REFERENCES `fabrica`.`prestamos` (`IdPrestamo`),

  CONSTRAINT `devoluciones\_ibfk\_3`

    FOREIGN KEY (`IdProducto`)

    REFERENCES `fabrica`.`productosgenerales` (`IdProducto`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4;

-- -----------------------------------------------------

-- Table `fabrica`.`usuarios`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `fabrica`.`usuarios` (

  `IdUsuario` INT(11) NOT NULL AUTO\_INCREMENT,

  `NombreUsuario` VARCHAR(255) NULL DEFAULT NULL,

  `ApellidoUsuario` VARCHAR(255) NULL DEFAULT NULL,

  `TipoIdentificacion` ENUM('CC', 'TI') NULL DEFAULT NULL,

  `NumeroIdentificacion` INT(11) NULL DEFAULT NULL,

  `CorreoUsuario` VARCHAR(255) NULL DEFAULT NULL,

  `CelularUsuario` INT(11) NULL DEFAULT NULL,

  `ContrasenaUsuario` VARCHAR(50) NULL DEFAULT NULL,

  PRIMARY KEY (`IdUsuario`))

ENGINE = InnoDB

AUTO\_INCREMENT = 3

DEFAULT CHARACTER SET = utf8mb4;

USE `db\_municipios` ;

-- -----------------------------------------------------

-- Table `db\_municipios`.`region`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `db\_municipios`.`region` (

  `idregion` INT(11) NOT NULL AUTO\_INCREMENT,

  `nombre\_region` VARCHAR(45) NOT NULL,

  PRIMARY KEY (`idregion`),

  UNIQUE INDEX `nombre\_region` (`nombre\_region` ASC) VISIBLE)

ENGINE = InnoDB

AUTO\_INCREMENT = 1124

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_spanish\_ci;

-- -----------------------------------------------------

-- Table `db\_municipios`.`departamento`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `db\_municipios`.`departamento` (

  `codigo\_dane\_departamento` INT(11) NOT NULL AUTO\_INCREMENT,

  `nombre\_departamento` VARCHAR(45) NOT NULL,

  `region\_idregion` INT(11) NOT NULL,

  PRIMARY KEY (`codigo\_dane\_departamento`),

  INDEX `fk\_departamento\_region1\_idx` (`region\_idregion` ASC) VISIBLE,

  CONSTRAINT `fk\_departamento\_region1`

    FOREIGN KEY (`region\_idregion`)

    REFERENCES `db\_municipios`.`region` (`idregion`)

    ON DELETE CASCADE

    ON UPDATE CASCADE)

ENGINE = InnoDB

AUTO\_INCREMENT = 100

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_spanish\_ci;

-- -----------------------------------------------------

-- Table `db\_municipios`.`municipio`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `db\_municipios`.`municipio` (

  `codigo\_dane\_municipio` FLOAT NOT NULL AUTO\_INCREMENT,

  `nombre\_municipio` VARCHAR(45) NOT NULL,

  `departamento\_codigo\_dane\_departamento` INT(11) NOT NULL,

  PRIMARY KEY (`codigo\_dane\_municipio`),

  INDEX `fk\_municipio\_departamento\_idx` (`departamento\_codigo\_dane\_departamento` ASC) VISIBLE,

  CONSTRAINT `fk\_municipio\_departamento`

    FOREIGN KEY (`departamento\_codigo\_dane\_departamento`)

    REFERENCES `db\_municipios`.`departamento` (`codigo\_dane\_departamento`)

    ON DELETE CASCADE

    ON UPDATE CASCADE)

ENGINE = InnoDB

AUTO\_INCREMENT = 100

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_spanish\_ci;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

### Script de mi base de datos databaseJumbo:

CREATE DATABASE IF NOT EXISTS databaseJumbo;

USE databaseJumbo;

CREATE TABLE Productos (

    id\_producto INT AUTO\_INCREMENT PRIMARY KEY,

    nombre VARCHAR(255),

    marca VARCHAR(100),

    modelo VARCHAR(100),

    capacidad VARCHAR(50),

    categoria VARCHAR(50),

    precio\_actual DECIMAL(10,2),

    year\_lanzamiento INT

) ENGINE=InnoDB;

CREATE TABLE Clientes (

    id\_cliente INT AUTO\_INCREMENT PRIMARY KEY,

    nombre VARCHAR(255),

    email VARCHAR(255),

    telefono VARCHAR(20)

) ENGINE=InnoDB;

CREATE TABLE Compras (

    id\_compra INT AUTO\_INCREMENT PRIMARY KEY,

    id\_cliente INT,

    fecha\_compra DATE,

    FOREIGN KEY (id\_cliente) REFERENCES Clientes(id\_cliente)

) ENGINE=InnoDB;

CREATE TABLE DetallesCompra (

    id\_detalle INT AUTO\_INCREMENT PRIMARY KEY,

    id\_compra INT,

    id\_producto INT,

    precio\_compra DECIMAL(10,2),

    cantidad INT,

    FOREIGN KEY (id\_compra) REFERENCES Compras(id\_compra),

    FOREIGN KEY (id\_producto) REFERENCES Productos(id\_producto)

) ENGINE=InnoDB;

CREATE TABLE Garantias (

    id\_garantia INT AUTO\_INCREMENT PRIMARY KEY,

    id\_compra INT,

    id\_producto INT,

    fecha\_inicio DATE,

    fecha\_fin DATE,

    FOREIGN KEY (id\_compra) REFERENCES Compras(id\_compra) ON DELETE CASCADE,

    FOREIGN KEY (id\_producto) REFERENCES Productos(id\_producto) ON DELETE CASCADE

) ENGINE=InnoDB;

-- Inserciones a las tablasd

INSERT INTO Productos (nombre, marca, modelo, capacidad, categoria, precio\_actual, year\_lanzamiento)

VALUES

('Nevera Mabe 300L', 'Mabe', 'MB300', '300 Litros', 'Nevera', 1800000, 2023),

('Televisor Samsung 55"', 'Samsung', 'QLED55', '55 Pulgadas', 'Televisor', 3000000, 2023);

INSERT INTO Clientes (nombre, email, telefono)

VALUES

('Juan Pérez', 'juanperez@email.com', '3001234567');

INSERT INTO Compras (id\_cliente, fecha\_compra)

VALUES (1, '2023-06-15');

INSERT INTO DetallesCompra (id\_compra, id\_producto, precio\_compra, cantidad)

VALUES (1, 1, 1800000, 1);

INSERT INTO Garantias (id\_compra, id\_producto, fecha\_inicio, fecha\_fin)

VALUES (1, 1, '2023-06-15', '2025-06-15');

SELECT \* FROM Productos;

SELECT \* FROM Clientes;

SELECT \* FROM Compras;

SELECT \* FROM DetallesCompra;

SELECT \* FROM Garantias;

DELIMITER $$

CREATE PROCEDURE depreciar\_precios()

BEGIN

    UPDATE Productos

    SET precio\_actual = precio\_actual \* (1 - 0.10);

END$$

DELIMITER ;

CREATE EVENT IF NOT EXISTS depreciar\_precios\_event

ON SCHEDULE EVERY 1 YEAR

STARTS '2025-02-17 12:12:30'

DO

  CALL depreciar\_precios();

SELECT \* FROM Productos;

SELECT g.id\_garantia, c.id\_compra, p.nombre, p.marca, p.modelo, dc.precio\_compra

    AS precio\_original, p.precio\_actual AS precio\_actual

FROM Garantias g

JOIN Compras c ON g.id\_compra = c.id\_compra

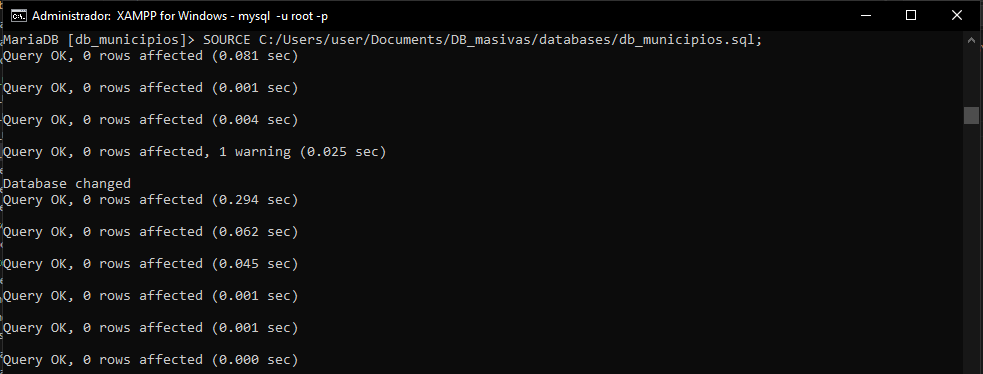
JOIN DetallesCompra dc ON c.id\_compra = dc.id\_compra

JOIN Productos p ON dc.id\_producto = p.id\_producto

WHERE c.id\_cliente = 1 AND p.id\_producto = 1;

## Descripciones o explicaciones de los scripts solicitados.

Tuve un problema en la instalación con el Docker en mi equipo, sin embargo, intenté hacerlo por medio de una maquina virtual con Docker en la nube de AWS, y tampoco fue posible.

De todos modos, tengo los scripts para indexar los datos a del archivo CSV a las tablas de “*db\_municipios*”:

Regiones:

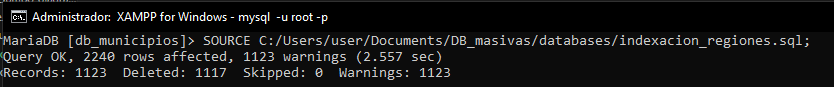
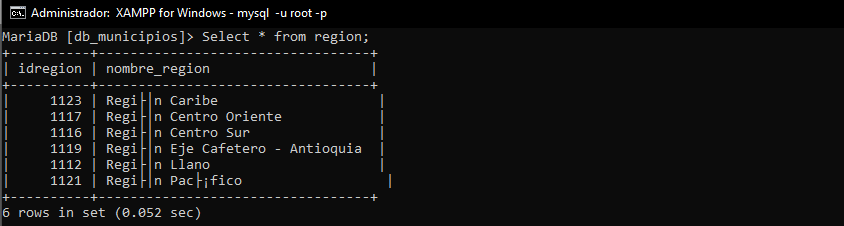
LOAD DATA INFILE 'C:/Users/user/Documents/DB\_masivas/datos\_municipios.csv'

REPLACE INTO TABLE region

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

(@nombre\_region) SET nombre\_region = TRIM(@nombre\_region);

Departamentos:

LOAD DATA INFILE 'C:/Users/user/Documents/DB\_masivas/datos\_municipios.csv'

REPLACE INTO TABLE departamento

FIELDS TERMINATED BY ','

OPTIONALLY ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

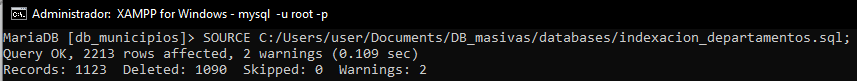
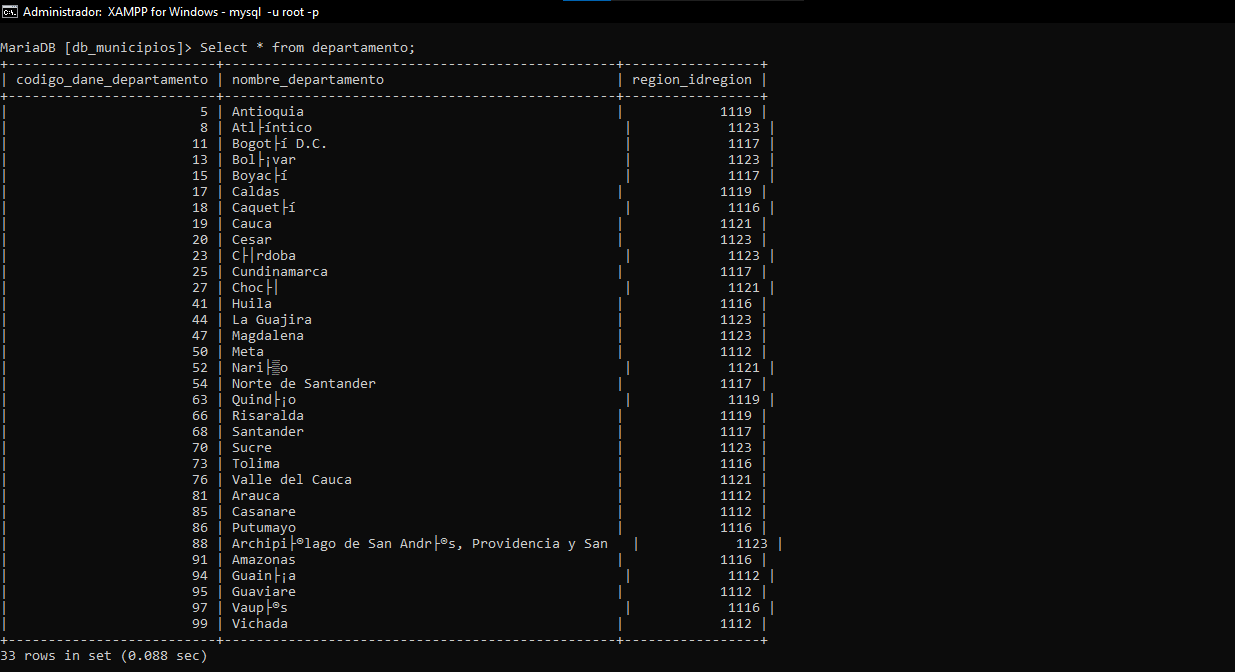
(@var1, @var2, @var3, @dummy1, @dummy2)

SET

    codigo\_dane\_departamento = TRIM(@var2),

    nombre\_departamento = TRIM(@var3),

    region\_idregion = (SELECT idregion FROM region WHERE TRIM(nombre\_region) = TRIM(@var1) LIMIT 1);



Municipios:

LOAD DATA INFILE 'C:/Users/user/Documents/DB\_masivas/datos\_municipios.csv'

REPLACE INTO TABLE municipio

FIELDS TERMINATED BY ','

OPTIONALLY ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

(@var1, @var2, @dummy1, @var4, @var5)

SET

    codigo\_dane\_municipio = TRIM(@var4),  -- Código DANE del municipio

    nombre\_municipio = TRIM(@var5),       -- Nombre del municipio

    departamento\_codigo\_dane\_departamento = (

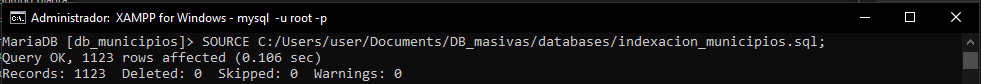
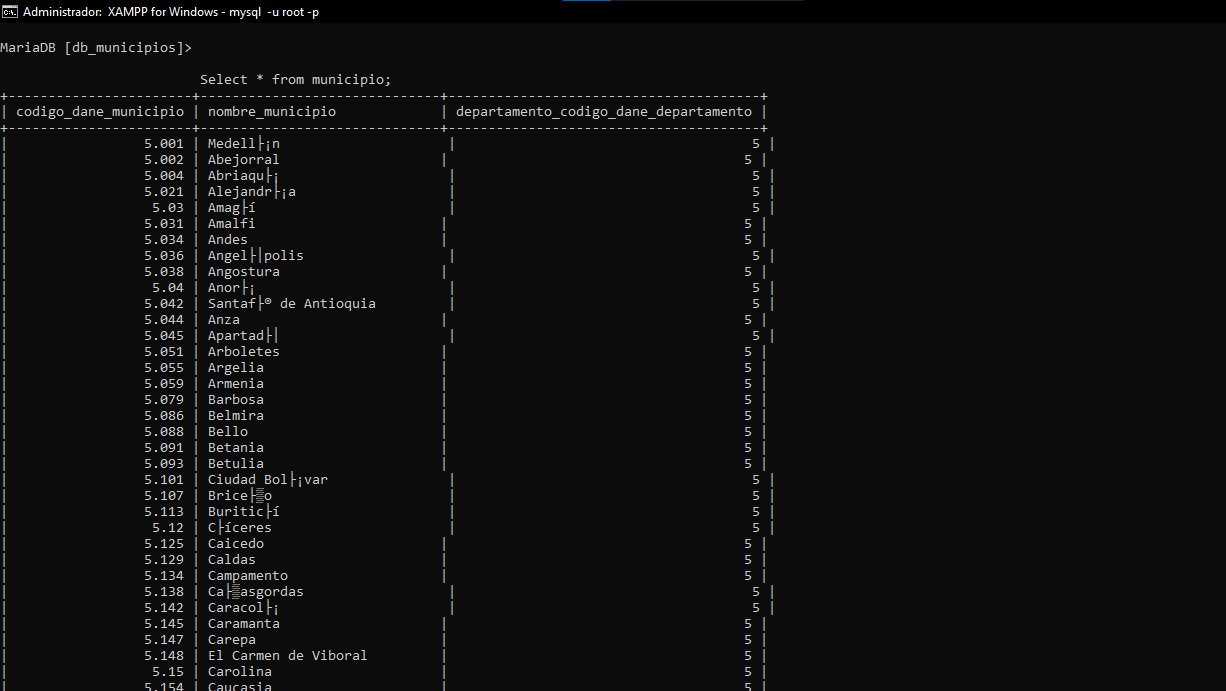
        SELECT codigo\_dane\_departamento

        FROM departamento

        WHERE TRIM(codigo\_dane\_departamento) = TRIM(@var2)

        LIMIT 1

    );

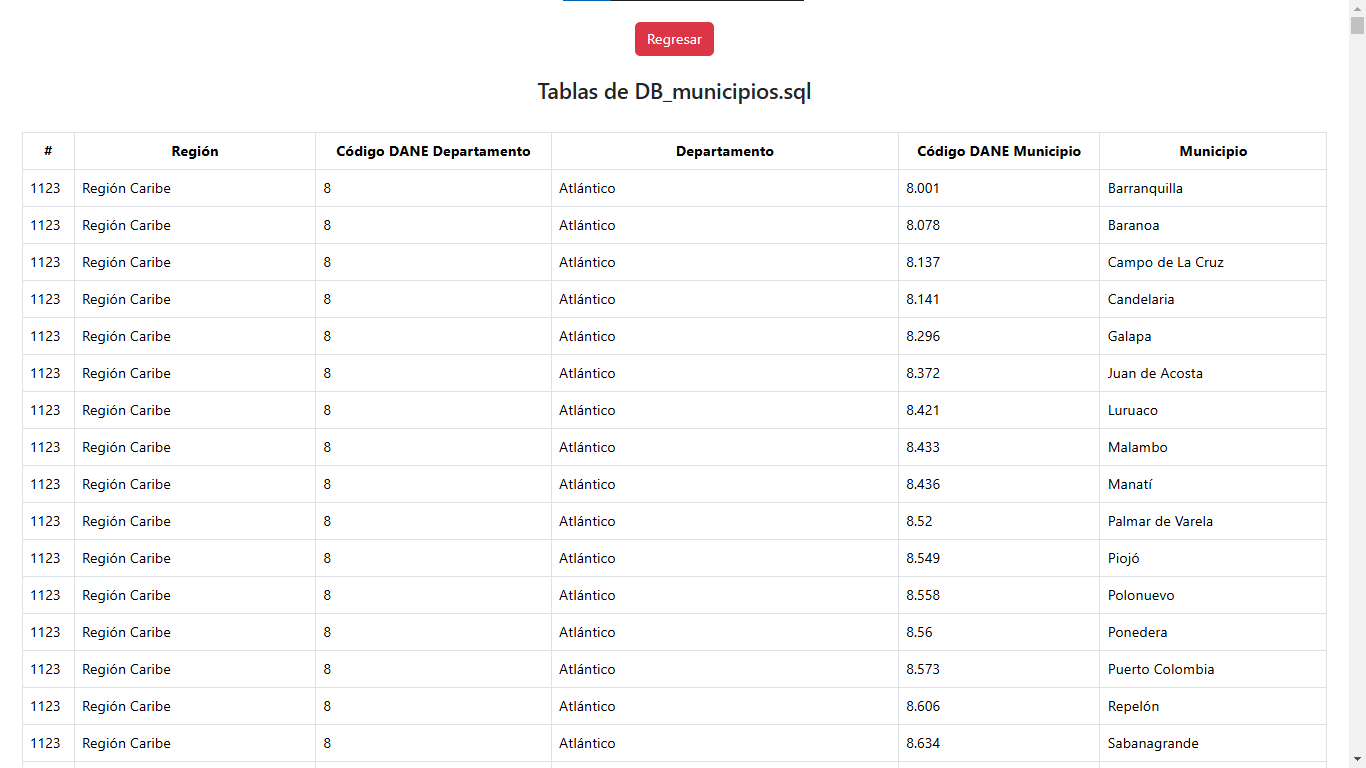


…

## Conclusiones Scripts Load Data Infile

Lo que hace este código de Load Data Infile es leer las filas que se escriben por mediante el script en la instrucción de *SELECT INTO OUTFILE*, después de esto ya puede volver a leer los archivos en las tablas por medio de la instrucción *LOAD DATA INFILE*. Las instrucciones *FIELDS* y *LINES* son las mismas instrucciones que de la misma manera predeterminada hace que se esperen que los campos terminen en tabulaciones y con nuevas líneas (*\n*). Cuando se ejecuta esta declaración se activan los *INSERT* en los activadores.

Se tiene que tener el privilegio y el permiso de *FILE* para que se pueda ejecutar el *LOAD DATA INFILE.* Esto nos ayuda a que los usuarios normales no puedan leer los archivos del sistema. *LOAD DATA LOCAL INFILE* no tienen ese requisito. De esta forma queda la indexación como una tabla igual que en el CSV:



Conclusiones Script databaseJumbo

En el script de mirar la garantía con precio de compra, precio original y precio actual

SELECT g.id\_garantia, c.id\_compra, p.nombre, p.marca, p.modelo, dc.precio\_compra

    AS precio\_original, p.precio\_actual AS precio\_actual

FROM Garantias g

JOIN Compras c ON g.id\_compra = c.id\_compra

JOIN DetallesCompra dc ON c.id\_compra = dc.id\_compra

JOIN Productos p ON dc.id\_producto = p.id\_producto

WHERE c.id\_cliente = 1 AND p.id\_producto = 1;

Nos da la posibilidad de mirar cómo se recupera la información sobre la garantía de un producto en específico que fué comprado por un cliente en particular. Si el cliente con id\_cliente = 1 compró un producto con id\_producto = 1, la consulta devolverá información sobre:

1. La garantía a la cual está asociada a esa compra
2. Los detalles del producto como el nombre, la marca y el modelo
3. El precio original al que se compró el producto
4. El precio actual que sirve para ver si ha cambiado desde que se hizo la compra

## Conclusiones

Es un trabajo orgánico que puede suceder el cualquier entorno real, el cuál nos enseña aún más a optimizar las tareas de datos por medio de entradas de SQL las cuales nos hacen la vida mucho más fácil, espero aprender mucho más de ello para poder implementar en un futuro el cuál sé que necesitaré implementarlo. Esto nos ha ayudad mucho a tener bases de datos masivas que permiten almacenar, procesar y analizar grandes cantidades de información de manera estructurada, lo cual no facilita mucho en la toma de decisiones en diferentes sectores como la industria tecnológica, en la salud y en el comercio.