Proyecto gestión de la información

Daniel Andrés Pinzón Jay

Id: 819793

Yeimi Tatiana Corzo Lizarazo

Id: 815483

Programa: Ingeniería de sistemas

gestión de la información

Docente: Gustavo Adolfo Gómez Gómez

Universidad cooperativa de Colombia

Sede: Bucaramanga

2024

UNIVERSIDAD COOPERATIVA DE COLOMBIA

ÍNDICE

Introducción

Se ha propuesto un proyecto que deberá implementarse de la manera más eficiente posible, donde se manejen datos y se elimine su redundancia, en la materia gestión en la información, hemos decidido llevar a cabo un modelo de gestión de la información para gestionar mercancía agropecuaria en una empresa. Reforzaremos nuestras capacidades dándole paso a los diagramas entidad-relación y esquemas relacionales, en la cual se le permitirá a la empresa organizar, almacenar y distribuir la información de manera eficiente. una vez ingresados todos los productos, de la procedencia de las fincas se eliminarán sus redundancias, esto maximizará la retención de conocimiento organizacional ya que se mejorará la calidad y se disminuiría la reducción de riesgos de los datos, llevándose a cabo en MySQL.

Objetivos

➢ Objetivo general

diseñar e implementar un sistema de modelo de base de datos para una adecuada administración de la información de producción, transporte y venta de productos agrícolas. Mejorando la eficiencia y recolección de datos para realizar consultas, análisis y mejorar la toma de decisiones del sector agrícola.

➢ Objetivo Específicos

- identificaremos las necesidades de los trabajos que desarrolla la finca, a partir de identificar las necesidades se crearan los diagramas entidad-relación que representaran las entidades y las relaciones, esta manera visual nos permitirá tener más claro como los datos están relacionados entre sí.

- pasaremos a convertir el diseño conceptual a un esquema relacional, para poder definir cómo vamos a almacenar y a organizar los datos, estos nos ayudaran a tener las tablas definidas y las relaciones para una mejor compresión antes de pasar a la implementación, teniendo en cuenta las restricciones, las llaves primarias, las llaves foráneas y donde proporcionaremos una base sólida y estructurada para el manejo de los datos.

- en mariadb pasaremos a crear la base de datos y a exportar los datos, crear las tablas e insertar los datos en un paso fundamental.

- se documentará el resultado de las pruebas y se harán cambios según las necesidades de las fincas para hacer futuros ajustes y tener un modelo de base de datos sólida.

Glosario

- **Mayordomo:** que también puede conocerse como el capataz o el administrador de la finca, este se encarga de la gestión y de los procesos que ocurren en la finca, supervisando el trabajo del personal y que todo se haga de manera correcta.
- **Etapa desarrollo Animal:** en este caso, indica el estado en el cual se encuentra el animal registrado. El animal se puede registrar como en estado de crianza (primeras 8 semanas en caso de las gallinas y cerdos y 8 meses para las vacas), el estado de crecimiento y finalmente el estado de "sacrificio" que indica cuando el animal está en momento óptimo del sacrificio.
- **Certificación Alimento:** Actualmente los productos necesitan una certificación que reflejara la forma en la cual fue producido el alimento. Existen certificaciones como la "orgánica" para alimentos que fueron producidos de manera natural o están los productos con certificación "OMG" (Organismos modificados genéticamente).
- **Cartón de huevo:** este se divide en 4 categorías, que son los huevos, B, A, AA, AAA y se clasifican por su tamaño y categoría, su clasificación nos ayuda a organizar el producto y también su venta en el mercado.
- **Etapa de desarrollo:** esta etapa es en los animales, donde se pueden generar ingresos y podemos darnos cuenta si este animal nos sirve para producción.
- **Lote de producción:** puede estar compuesto por distintos alimentos que se envían de la fica al punto de venta, con el propósito de que se comercialice.
- **Raza:** la raza nos ayuda para distinguir el animal, ya que no todos tiene el mismo cuidado y precio, también que no todos son hechos para vender, si no para seguir produciendo alimentos de calidad, es importante ya que esto puede afectar la calidad del producto final o su rentabilidad.

Definición del problema

Actualmente la falta de un sistema para guardar y recopilar información que la organice, dificulta la administración de la mercancía, lo que conlleva a procesos ineficientes donde se pierde información clave, no se tiene un acceso directo a la coherencia de los datos, hay errores de redundancia y riesgos ya que la misma información puede ser ingresada desde varios lugares lo que provoca confusiones y discrepancia, también dificulta el seguimiento del inventario con proveedores y clientes, con estos fallos hay grandes limitaciones a la hora de analizar los datos y tomar decisiones

Marco teórico

la gestión de las fincas se ha implementado en un modelo de base de datos utilizaremos tecnologías como los modelos relacionales de datos.

1.Modelos Relacionales de Datos

Los modelos relacionales de datos están para la organización y gestión de la información en nuestra base de datos. Este modelo utiliza tablas de relaciones para representar los datos y sus relaciones, permitiendo operaciones como inserción, actualización, eliminación y consulta mediante SQL.

- Estructura organizada: Facilita la comprensión y manejo de datos complejos.
- Integridad de datos: Asegura que las relaciones entre tablas se mantengan correctas.
- Flexibilidad: Permite realizar operaciones complejas con facilidad.

2. Diagramas Entidad-Relación

Los diagramas ER son herramientas fundamentales para el diseño conceptual de la base de datos. Representan las entidades (objetos o conceptos del mundo real) y las relaciones entre ellas, proporcionando una visión clara de la estructura de la base de datos.

- Herramienta utilizada: Draw.io
- Visualización: Permite representar visualmente las entidades y sus relaciones y ayuda mucho en el trabajo en equipo.
- Comunicación: Facilita la comunicación con los miembros del equipo sobre la estructura de la base de datos.
- Base para implementación: Sirve como guía para la creación del esquema relacional en el sistema de gestión de base de datos.

3. Esquemas Relacionales

El diseño de esquemas relacionales se basa en los diagramas de entidad-relación se utiliza para crear la estructura lógica de la base de datos en términos de tablas, columnas y relaciones. para asegurarnos que la base de datos sea eficiente y funcional.

- Organización: Estructura lógica y organizada de los datos.
- Eficiencia: Optimiza el almacenamiento y acceso a los datos.
- Integridad: Mantiene la consistencia y exactitud de los datos.

4. MySQL

MySQL es un sistema de gestión de bases de datos relacional de código abierto, utilizado para gestionar grandes volúmenes de datos de manera eficiente. Es el motor de base de datos principal para nuestra aplicación de gestión de fincas.

- Gestión eficiente de datos: Permite realizar operaciones de inserción, actualización, eliminación y consulta de datos.
- Código abierto: Accesible y modificable según las necesidades del proyecto.
- SQL: es un lenguaje para interactuar con la base de datos, proporcionando una interfaz flexible para la gestión de datos.

5. Línea de Comando (CLI)

Utilizaremos la línea de comando (CLI) para ejecutar comandos y consultas directamente desde la terminal. esto incluye la ejecución de vistas, consultas anidadas y operadores avanzados (como subconsultas anidadas: ESCALAR, PERTENENCIA ALL, ANY, EXISTS y FROM).

- Precisión: Permite ejecutar comandos con exactitud.
- Eficiencia: Facilita la automatización de tareas repetitivas.
- Flexibilidad: Proporciona acceso directo a la base de datos para operaciones avanzadas.

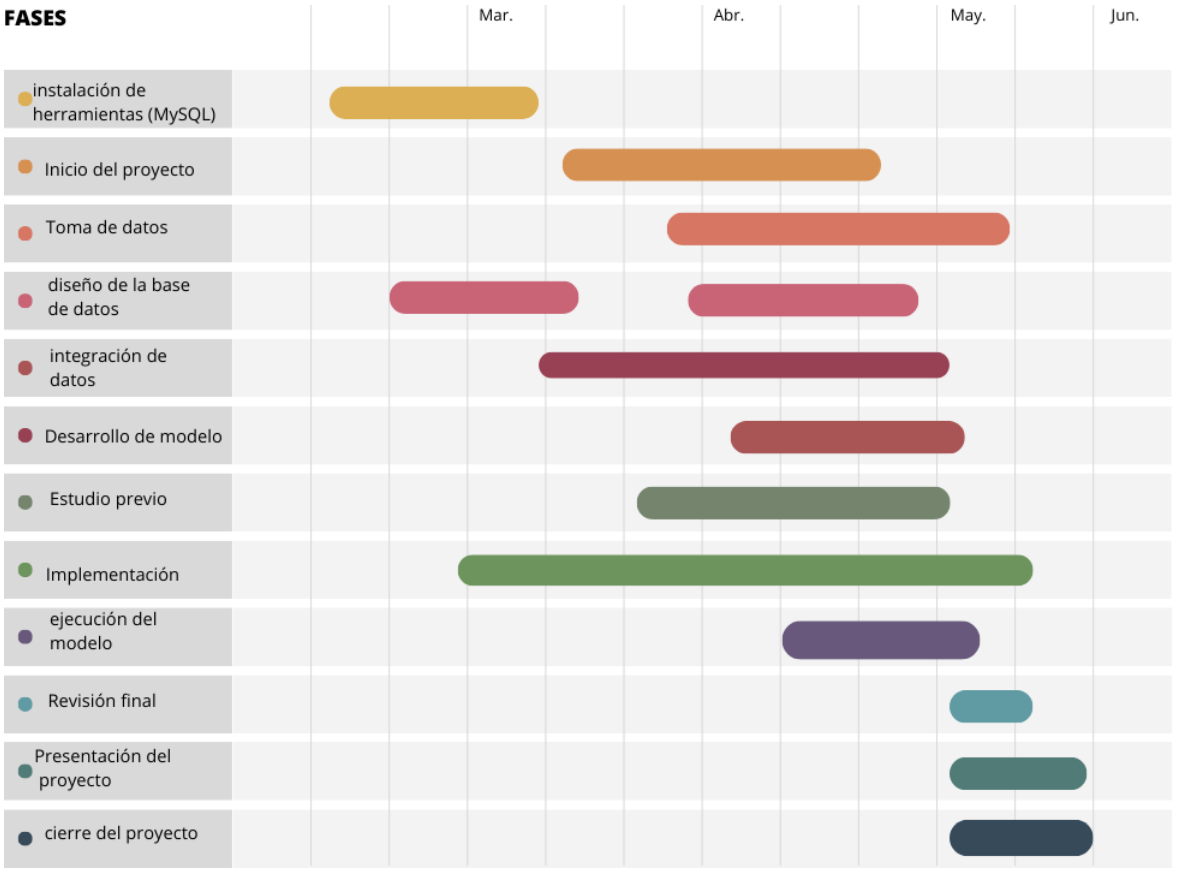
6. Visual Studio Code

Visual Studio Code es un editor de código fuente que utilizaremos para gestionar scripts y documentar comandos y consultas realizadas. Es especialmente útil para el trabajo en equipo y la comprensión de la estructura de la base de datos.

- Facilita el trabajo en equipo mediante herramientas de control de versiones como Git.
- Documentación: Permite mantener una documentación detallada y organizada de la base de datos.
- Extensibilidad: Soporta una amplia gama de extensiones para mejorar la funcionalidad y llevar un control de los datos ingresados.

UNIVERSIDAD COOPERATIVA DE COLOMBIA

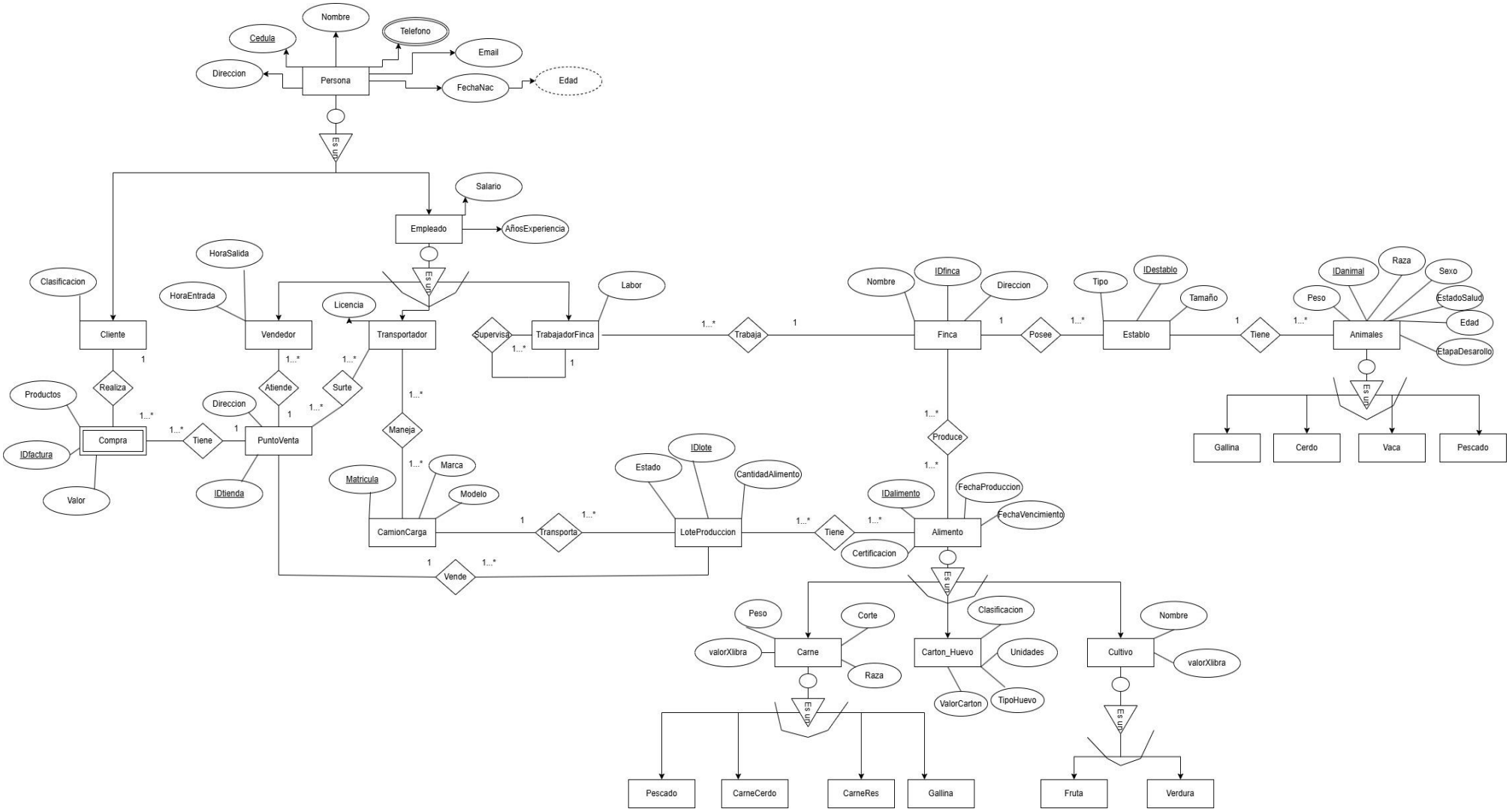
Cronograma



UNIVERSIDAD COOPERATIVA DE COLOMBIA

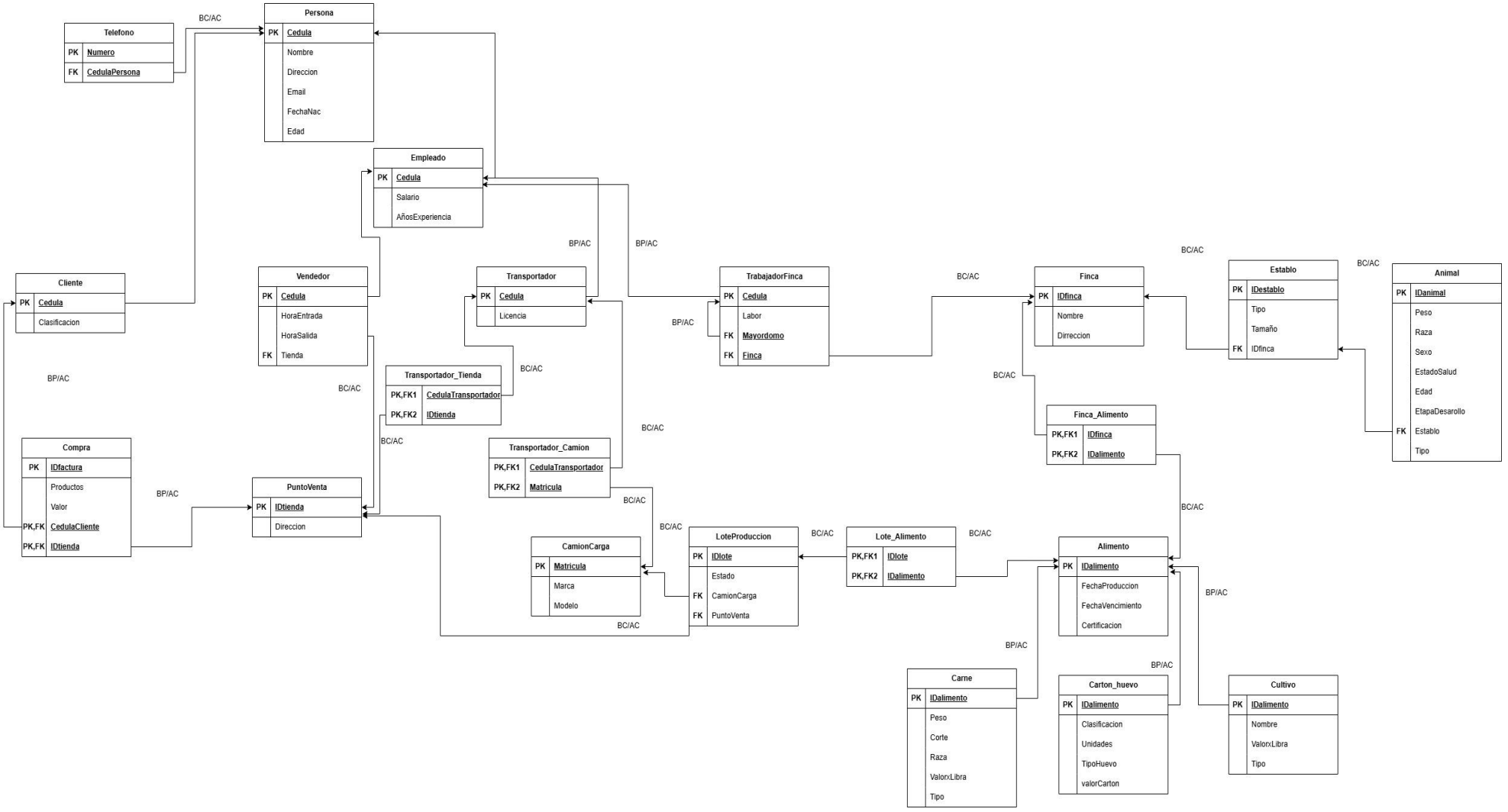
Diseño

Entidad-Relación



UNIVERSIDAD COOPERATIVA DE COLOMBIA

Esquema relacional



Nota: Se adjuntará los diagramas por aparte para una mejor visualización.

DESAROLLO

Código SQL DDL por cada tabla.

Tabla PuntoVenta:

```
CREATE TABLE puntoVenta (
    IDtienda INT NOT NULL,
    direccion VARCHAR(50),
    PRIMARY KEY(IDtienda));
```

En esta tabla registraremos los puntos de venta de productos agrícolas con los que cuentan la empresa.

Tabla persona:

```
CREATE TABLE persona (
    cedula INT NOT NULL,
    nombre VARCHAR(100),
    fechaNac DATE,
    edad INT,
    direccion VARCHAR(100),
    genero CHAR(1),
    PRIMARY KEY(cedula),
    CHECK(genero in("F","M")));
```

La tabla persona será en la cual registremos los atributos que tienen en común tanto los clientes como los empleados (cedula, nombre, etc.). Esta tabla tendrá la tabla cliente y la tabla Empleado como subtipos.

Tabla Cliente:

```
CREATE TABLE cliente (
    cedula INT NOT NULL,
    clasificacion VARCHAR(30),
    PRIMARY KEY(cedula),
    FOREIGN KEY (cedula) REFERENCES persona(cedula),
    CHECK(clasificacion in("Minorista", "Mayorista")));
```

La tabla cliente será en la que guardemos los datos de nuestros clientes, esto es útil en la vida real para realizar procesos de facturación.

Tabla Empleado:

```
CREATE TABLE empleado (
    cedula INT NOT NULL,
    salario INT,
    añosExperiencia INT,
    PRIMARY KEY(cedula),
    FOREIGN KEY (cedula) REFERENCES persona(cedula));
```

La table Empleado será en la cual guardemos los datos propios de los empleados de la empresa, contara con 3 subtipos que serán los siguientes:

Tabla Vendedor:

```
CREATE TABLE vendedor (
    cedula INT NOT NULL,
    HoraEntrada TIME,
    HoraSalida TIME,
    tienda INT,
    PRIMARY KEY(cedula),
    FOREIGN KEY (cedula) REFERENCES Empleado(cedula),
    FOREIGN KEY(tienda) REFERENCES puntoVenta(IDtienda));
```

La tabla vendedor subtipo de empleado donde se registrarán los datos de los vendedores del punto de venta.

Tabla Transportador:

```
CREATE TABLE transportador (
    cedula INT NOT NULL,
    licencia INT,
    PRIMARY KEY(cedula),
    FOREIGN KEY (cedula) REFERENCES Empleado(cedula));
```

La table transportador guardaremos los choferes registrados en la empresa.

Tabla TrabajadorFinca:

```
CREATE TABLE trabajadorFinca(
    cedula INT NOT NULL,
    Labor VARCHAR(28),
    mayordomo INT,
    finca INT,
    PRIMARY KEY(cedula),
    FOREIGN KEY (cedula) REFERENCES Empleado(Cedula),
    FOREIGN KEY (mayordomo) REFERENCES trabajadorFinca(cedula),
    FOREIGN KEY (finca) REFERENCES finca(IDfinca));
```

En esta tabla registraremos a todos los trabajadores agrícolas de la empresa.

Tabla Compra:

```
CREATE TABLE compra (
    IDfactura INT NOT NULL,
    productos INT,
    valor INT,
    cedula_cliente INT NOT NULL,
    IDtienda INT NOT NULL,
    PRIMARY KEY(IDfactura, cedula_cliente, IDtienda),
    FOREIGN KEY(cedula_cliente) REFERENCES cliente(Cedula),
    FOREIGN KEY(IDtienda) REFERENCES puntoVenta(IDtienda));
```

La tabla compra registrara todas las compras realizadas en algún punto de venta registrando igualmente el cliente.

Tabla Tranportador_Tienda:

```
CREATE TABLE transportador_tienda (
    cedula_transportador INT NOT NULL,
    IDtienda INT NOT NULL,
    PRIMARY KEY(cedula_transportador, IDtienda),
    FOREIGN KEY(cedula_transportador) REFERENCES transportador(cedula),
    FOREIGN KEY(IDtienda) REFERENCES puntoVenta(IDtienda));
```

Tabla en la cual se guardara la relación entre el transportador y la tienda a la que surte.

Tabla Tranportador_Tienda:

```
CREATE TABLE CamionCarga (
    matricula INT NOT NULL,
    marca VARCHAR(20),
    modelo VARCHAR(20),
    PRIMARY KEY(matricula));
```

Tabla en la cual registraremos todos los camiones de la empresa.

Tabla Tranportador_Tienda:

```
CREATE TABLE transportador_camion (
    cedula_transportador INT NOT NULL,
    matricula INT NOT NULL,
    PRIMARY KEY(cedula_transportador, matricula),
    FOREIGN KEY(cedula_transportador) REFERENCES transportador (cedula),
    FOREIGN KEY(matricula) REFERENCES CamionCarga (matricula));
```

Tabla en la cual registraremos tanto el transportador como el camión el cual maneja.

Tabla Finca:

```
CREATE TABLE finca (
    IDfinca INT NOT NULL,
    nombre VARCHAR(25),
    direccion VARCHAR(100),
    PRIMARY KEY(IDfinca));
```

Tabla en la cual registraremos todas las fincas de la empresa para la producción agrícola.

Tabla Alimento:

```
CREATE TABLE alimento (
    IDalimento INT NOT NULL,
    fechaProduccion VARCHAR(20),
    fechaVencimiento VARCHAR(20),
    certificacion VARCHAR(30) CHECK (certificacion IN ('Orgánico', 'OMG')),
    PRIMARY KEY (IDalimento));
```

Tabla en la cual registraremos el alimento producido por la empresa.

Tabla Carne:

```
CREATE TABLE carne (
    IDalimento INT NOT NULL,
    peso INT,
    corte VARCHAR(20),
    raza VARCHAR(20),
    valorXlibra INT,
    tipo VARCHAR(20) CHECK (tipo IN ('Res', 'Cerdo', 'Pescado', 'Gallina')),
    PRIMARY KEY(IDalimento),
    FOREIGN KEY(IDalimento) REFERENCES alimento(IDalimento));
```

Subtipo de la tabla alimento en la cual guardaremos toda la producción realizada de carnes.

Tabla CartonHuevo:

```
CREATE TABLE cartonHuevo (
    IDalimento INT NOT NULL,
    clasificacion VARCHAR(28) CHECK (clasificacion IN ('B', 'A', 'AA', 'AAA')),
    unidades INT,
    tipoHuevo VARCHAR(15) CHECK (tipoHuevo IN ('Tradicional', 'Criollo')),
    ValorCarton INT,
    PRIMARY KEY(IDalimento),
    FOREIGN KEY(IDalimento) REFERENCES alimento(IDalimento)
);
```

Subtipo de la tabla alimento en la cual guardaremos toda la producción realizada de cartones de huevo.

Tabla Cultivo:

```
CREATE TABLE cultivo (
    IDalimento INT NOT NULL,
    nombre VARCHAR(20),
    valorXlibra INT,
    tipo VARCHAR(28),
    PRIMARY KEY(IDalimento),
    FOREIGN KEY(IDalimento) REFERENCES alimento(IDalimento));
```

Subtipo de la tabla alimento en la cual guardaremos toda la producción realizada de cultivos.

Tabla Cultivo:

```
CREATE TABLE finca_alimento (
    IDfinca INT NOT NULL,
    IDalimento INT NOT NULL,
    PRIMARY KEY(IDfinca, IDalimento),
    FOREIGN KEY(IDfinca) REFERENCES finca(IDfinca),
    FOREIGN KEY (IDalimento) REFERENCES alimento(IDalimento));
```

Tabla en la cual guardaremos la relación entre la finca y el alimento producido. Se crea una tabla por aparte ya que una finca produce mucho alimento y un alimento es producido por muchas fincas.

Tabla LoteProduccion:

```
CREATE TABLE loteProduccion (
    IDlote INT NOT NULL,
    estado VARCHAR(20),
    camionCarga INT NOT NULL,
    puntoVenta INT NOT NULL,
    PRIMARY KEY(IDlote),
    FOREIGN KEY(camionCarga) REFERENCES camionCarga(matricula),
    FOREIGN KEY(puntoVenta) REFERENCES puntoVenta(IDtienda));
```

Tabla en la cual guardaremos los datos de los lotes de producción. Un lote de producción es un conjunto de distintos alimentos que se envía de la finca a el punto de venta.

Tabla lote_alimento:

```
CREATE TABLE lote_alimento (
    IDlote INT NOT NULL,
    IDalimento INT NOT NULL,
    PRIMARY KEY(IDlote, IDalimento),
    FOREIGN KEY(IDlote) REFERENCES loteProduccion(IDlote),
    FOREIGN KEY(IDalimento) REFERENCES alimento(IDalimento));
```

Tabla en la cual guardaremos la relación entre el lote de producción y el alimento. Se debe hacer en una tabla por aparte ya que un lote tiene muchos alimentos y un alimento puede estar en muchos lotes.

Tabla establo:

```
CREATE TABLE establo (
    IDestablo INT NOT NULL,
    tipo VARCHAR(25),
    tamaño INT,
    IDfinca INT NOT NULL,
    PRIMARY KEY(IDestablo),
    FOREIGN KEY(IDfinca) REFERENCES finca(IDfinca));
```

Tabla en la cual registraremos todos los establos de las distintas fincas.

Tabla Animal:

```
CREATE TABLE animal (
    IDanimal INT NOT NULL,
    peso INT,
    raza VARCHAR(20),
    sexo VARCHAR(5),
    estadoSalud VARCHAR(25),
    edad INT,
    etapaDesarrollo VARCHAR(28),
    establo INT,
    tipo VARCHAR (28) CHECK (tipo IN ('Vaca', 'Cerdo', 'Pescado', 'Gallina')),
    CHECK(sexo in("F","M")),
    PRIMARY KEY(IDanimal),
    FOREIGN KEY(establo) REFERENCES establo(IDestablo));
```

Tabla en la cual registraremos todos los animales de las fincas en sus respectivos establos.

VISTAS

Vista 1: la primera vista nos permitirá la visualización tanto del mayordomo de cada finca junto con todos los trabajadores de esta.

```
CREATE VIEW Finca_Mayordomo_Trabajadores AS
SELECT
   f.nombre AS Finca, p.nombre AS Mayordomo,
   (
      SELECT GROUP_CONCAT(p.nombre)
      FROM trabajadorFinca tf2
      JOIN Persona p ON tf2.Cedula = p.Cedula
      WHERE tf2.finca = tf.finca AND tf2.mayordomo IS NOT NULL
   ) AS Trabajadores
FROM trabajadorFinca tf
JOIN Persona p ON tf.Cedula = p.Cedula
JOIN finca f ON tf.finca = f.IDfinca
WHERE tf.mayordomo IS NULL
ORDER BY f.IDfinca;
```

```
MariaDB [Proyecto]> select * from Finca_Mayordomo_Trabajadores;
+----------+----------------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| Finca    | Mayordomo      | Trabajadores                                                                                                                                                                  |
+----------+----------------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| Napoleon | Gustavo Mendez | Mateo Sanchez,Ricardo Gonzalez,Daniela Rodriguez,Juliana Gomez,Sebastian Ramirez,Carolina Diaz,Sara Hernandez,Valentina Garcia,Alejandro Martinez                              |
| Maria    | Fabian Castro  | Manuel Alvarez,Monica Perez,Camilo Rojas,Juanita Ruiz,David Jimenez,Estefania Lopez,Andrea Lopez,Paola Garcia,Nicolas Gomez,Angela Ramirez                                    |
| Esperanza | Roberto Castillo | Gabriel Lopez,Adriana Ortiz,Lucia Marin,Felipe Duque,Eduardo Gomez,Tatiana Moreno                                                                                             |
| Rosal    | Marcela Guerrero | Catalina Gomez,Hernan Torres,Laura Alvarez,Juan Pablo Castro,Diego Rodriguez                                                                                                  |
| Antonio  | Sofia Diaz     | Luis Rodriguez,Carolina Molina,Daniel Gomez,Valentina Marin                                                                                                                   |
+----------+----------------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
```

Vista 2: la segunda vista nos permitirá la visualización de todos las tiendas y los productos (su inventario) que posee en el registro de datos.

```
CREATE VIEW ProductoPorTienda AS
SELECT lp.puntoVenta AS Tienda, pv.direccion AS DireccionTienda,
   CASE
      WHEN a.IDalimento IN (SELECT IDalimento FROM carne) THEN c.tipo
      WHEN a.IDalimento IN (SELECT IDalimento FROM cartonHuevo) THEN "Carton Huevo"
      WHEN a.IDalimento IN (SELECT IDalimento FROM cultivo cu) THEN cu.tipo
      ELSE "Desconocido"
   END AS TipoProducto,
   la.numeroLibras, a.fechaProduccion AS FechaProduccion, a.fechaVencimiento AS
FechaVencimiento
FROM loteProduccion lp
JOIN puntoVenta pv ON lp.puntoVenta = pv.IDtienda
JOIN lote_alimento la ON lp.IDlote = la.IDlote
JOIN alimento a ON la.IDalimento = a.IDalimento
LEFT JOIN carne c ON a.IDalimento = c.IDalimento
LEFT JOIN cultivo cu ON a.IDalimento = cu.IDalimento;
```

```
MariaDB [Proyecto]> select * from ProductoPorTienda;
+--------+-----------------+--------------+--------------+-----------------+------------------+
| Tienda | DireccionTienda | TipoProducto | numeroLibras | FechaProduccion | FechaVencimiento |
+--------+-----------------+--------------+--------------+-----------------+------------------+
|  10097 | Calle 52 #16-88 | Grano        |          500 | 2024-04-30      | 2024-05-30       |
|  10097 | Calle 52 #16-88 | Vegetal      |          700 | 2024-04-28      | 2024-05-28       |
|  10097 | Calle 52 #16-88 | Tubérculo    |          300 | 2024-04-22      | 2024-05-22       |
|  10097 | Calle 52 #16-88 | Carton Huevo |          100 | 2024-04-18      | 2024-05-18       |
|  10097 | Calle 52 #16-88 | Carne cerdo  |          400 | 2024-04-06      | 2024-06-06       |
|  10097 | Calle 52 #16-88 | Gallina      |          200 | 2024-03-28      | 2024-06-27       |
|  10097 | Calle 52 #16-88 | Pescado      |          600 | 2024-03-17      | 2024-05-16       |
|  10097 | Calle 52 #16-88 | Vegetal      |          400 | 2024-04-23      | 2024-05-23       |
|  10097 | Calle 52 #16-88 | Carton Huevo |          500 | 2024-04-13      | 2024-05-28       |
|  10097 | Calle 52 #16-88 | Carne res    |          300 | 2024-04-07      | 2024-06-07       |
|  10097 | Calle 52 #16-88 | Carton Huevo |          200 | 2024-04-01      | 2024-06-01       |
|  10097 | Calle 52 #16-88 | Gallina      |          700 | 2024-03-23      | 2024-05-22       |
|  10097 | Calle 52 #16-88 | Pescado      |          600 | 2024-03-12      | 2024-06-11       |
|  10098 | Calle 23 #34-23 | Grano        |          600 | 2024-04-29      | 2024-05-29       |
|  10098 | Calle 23 #34-23 | Tubérculo    |          400 | 2024-04-27      | 2024-06-27       |
|  10098 | Calle 23 #34-23 | Vegetal      |          500 | 2024-04-21      | 2024-05-21       |
|  10098 | Calle 23 #34-23 | Carton Huevo |          200 | 2024-04-17      | 2024-07-17       |
|  10098 | Calle 23 #34-23 | Carne cerdo  |          300 | 2024-04-05      | 2024-06-05       |
|  10098 | Calle 23 #34-23 | Gallina      |          700 | 2024-03-27      | 2024-05-26       |
|  10098 | Calle 23 #34-23 | Pescado      |          800 | 2024-03-16      | 2024-06-15       |
|  10098 | Calle 23 #34-23 | Grano        |          800 | 2024-04-30      | 2024-05-30       |
|  10098 | Calle 23 #34-23 | Vegetal      |          600 | 2024-04-21      | 2024-05-21       |
|  10098 | Calle 23 #34-23 | Carne res    |          500 | 2024-04-12      | 2024-05-28       |
|  10098 | Calle 23 #34-23 | Carton Huevo |          400 | 2024-03-31      | 2024-05-30       |
|  10098 | Calle 23 #34-23 | Carton Huevo |          200 | 2024-03-30      | 2024-06-29       |
|  10098 | Calle 23 #34-23 | Gallina      |          300 | 2024-03-22      | 2024-06-21       |
|  10099 | Calle 46 #32-21 | Leguminosa   |          800 | 2024-04-26      | 2024-05-26       |
|  10099 | Calle 46 #32-21 | Carton Huevo |          400 | 2024-04-20      | 2024-05-20       |
|  10099 | Calle 46 #32-21 | Carton Huevo |          300 | 2024-04-16      | 2024-07-16       |
|  10099 | Calle 46 #32-21 | Carne res    |          500 | 2024-04-10      | 2024-05-10       |
```

UNIVERSIDAD COOPERATIVA DE COLOMBIA

CONSULTAS

1. Se requiere saber el numero de ventas por cada punto de venta

SELECT pv.IDtienda, COUNT(c.IDfactura) AS NumeroDeVentas

FROM puntoVenta pv

LEFT JOIN compra c ON pv.IDtienda = c.IDtienda

GROUP BY pv.IDtienda;

```
MariaDB [Proyecto]> SELECT pv.IDtienda, COUNT(c.IDfactura) AS NumeroDeVentas
    -> FROM puntoVenta pv
    -> LEFT JOIN compra c ON pv.IDtienda = c.IDtienda
    -> GROUP BY pv.IDtienda;
+----------+----------------+
| IDtienda | NumeroDeVentas |
+----------+----------------+
|    10097 |              5 |
|    10098 |              5 |
|    10099 |              5 |
|   100100 |              5 |
|   100101 |              5 |
+----------+----------------+
5 rows in set (0.003 sec)
```

2. Se requiere saber los la etapa de desarrollo de los animales en la finca Maria

SELECT a.IDanimal, a.tipo, a.EtapaDesarrollo, e.IDestablo as Establo

FROM Animal a

JOIN establo e ON a.Establo = e.IDestablo

JOIN finca f ON e.IDfinca = f.IDfinca

WHERE f.Nombre = "Maria";

```
MariaDB [Proyecto]> SELECT a.IDanimal, a.tipo, a.EtapaDesarrollo, e.IDestablo as Establo
    -> FROM Animal a
    -> JOIN establo e ON a.Establo = e.IDestablo
    -> JOIN finca f ON e.IDfinca = f.IDfinca
    -> WHERE f.Nombre = "Maria";
+----------+--------+----------------+---------+
| IDanimal | tipo   | EtapaDesarrollo | Establo |
+----------+--------+----------------+---------+
|        7 | Vaca   | Crecimiento    |      26 |
|       23 | Vaca   | Crecimiento    |      26 |
|        8 | Cerdo  | Crianza        |      27 |
|       24 | Cerdo  | Crianza        |      27 |
|        9 | Gallina| Crianza        |      28 |
|       25 | Gallina| Crianza        |      28 |
|       10 | Pescado| Crecimiento    |      29 |
|       26 | Pescado| Crecimiento    |      29 |
|       11 | Vaca   | Crecimiento    |     210 |
|       27 | Vaca   | Crecimiento    |     210 |
+----------+--------+----------------+---------+
10 rows in set (0.001 sec)
```

3. Se requiere saber el promedio de salarios divido por labor en la finca Napoleon.

```
SELECT tf.Labor, COUNT(*) AS TotalTrabajadores,
    (SELECT AVG(salario)
     FROM empleado e
     WHERE e.cedula = tf.cedula
    ) AS SalarioPromedio
FROM trabajadorFinca tf
WHERE tf.finca = 41
GROUP BY tf.Labor;
```

```
MariaDB [Proyecto]> SELECT tf.Labor, COUNT(*) AS TotalTrabajadores,
    ->        (SELECT AVG(salario)
    ->         FROM empleado e
    ->         WHERE e.cedula = tf.cedula
    ->        ) AS SalarioPromedio
    -> FROM trabajadorFinca tf
    -> WHERE tf.finca = 41
    -> GROUP BY tf.Labor;
+----------------+-------------------+-----------------+
| Labor          | TotalTrabajadores | SalarioPromedio |
+----------------+-------------------+-----------------+
| Cuidado Ganado |                 4 |    1950000.0000 |
| Cultivador     |                 2 |    1800000.0000 |
| Limpieza       |                 2 |    2000000.0000 |
| Mantenimiento  |                 1 |    1650000.0000 |
| Mayordomo      |                 1 |    3750000.0000 |
+----------------+-------------------+-----------------+
5 rows in set (0.002 sec)
```

4. Se requiere conocer las horas que trabajan al día los vendedores

```
SELECT p.nombre AS NombreVendedor,
    v.cedula AS CedulaVendedor,
    SUM(TIMESTAMPDIFF(HOUR, v.HoraEntrada, v.HoraSalida)) AS TotalHorasTrabajadas
FROM vendedor v
JOIN persona p ON v.cedula = p.cedula
GROUP BY v.cedula, p.nombre;
```

```
MariaDB [Proyecto]> SELECT p.nombre AS NombreVendedor,
    ->        v.cedula AS CedulaVendedor,
    ->        SUM(TIMESTAMPDIFF(HOUR, v.HoraEntrada, v.HoraSalida)) AS TotalHorasTrabajadas
    -> FROM vendedor v
    -> JOIN persona p ON v.cedula = p.cedula
    -> GROUP BY v.cedula, p.nombre;
+-------------------+---------------+---------------------+
| NombreVendedor    | CedulaVendedor | TotalHorasTrabajadas |
+-------------------+---------------+---------------------+
| Diego Castro      |         63142 |                   9 |
| Raul Alvarez      |         65781 |                   5 |
| Oscar Ramirez     |        137245 |                   6 |
| Laura Herrera     |        196020 |                   5 |
| Gabriela Fernandez |       1006447 |                   9 |
| Fernanda Torres   |       1009645 |                   9 |
| Valeria Martinez  |       1098537 |                   6 |
+-------------------+---------------+---------------------+
7 rows in set (0.001 sec)
```

Nota: (TIMESTAMPDIFF calcula la diferencia entre dos variables tipo DATE o tipo Time)

5. Se requiere saber todos los datos de las personas que trabajan en la finca Antonio

```
SELECT * FROM persona WHERE cedula IN (
    SELECT cedula
    FROM empleado
    WHERE cedula IN (
        SELECT cedula
        FROM trabajadorFinca
        WHERE finca = 45));
```

```
MariaDB [Proyecto]> SELECT * FROM persona WHERE cedula IN (
    ->     SELECT cedula
    ->     FROM empleado
    ->     WHERE cedula IN (
    ->         SELECT cedula
    ->         FROM trabajadorFinca
    ->         WHERE finca = 45));
+---------+-----------------+------------+------+-------------------+--------+
| cedula  | nombre          | fechaNac   | edad | direccion         | genero |
+---------+-----------------+------------+------+-------------------+--------+
|   63541 | Luis Rodriguez  | 1980-10-18 |   44 | Calle 55 #16-22   | M      |
| 1000245 | Carolina Molina | 2000-05-20 |   24 | Carrera 20 #18-23 | F      |
| 1025486 | Sofia Diaz      | 1989-04-03 |   35 | Carrera 28 #17-22 | F      |
| 1096255 | Daniel Gomez    | 1996-01-08 |   28 | Carrera 38 #12-18 | M      |
| 1455747 | Valentina Marin | 1993-12-05 |   31 | Carrera 50 #20-25 | F      |
+---------+-----------------+------------+------+-------------------+--------+
5 rows in set (0.001 sec)
```

6. Se requiere saber cuántos animales hay en los establos, que animales hay y el id de los establos

```
SELECT e.IDestablo, e.tipo AS "Tipo_Establo", a.Tipo_Animal, a.Numero_Animales
FROM establo e
JOIN (
    SELECT establo, MIN(tipo) AS "Tipo_Animal", COUNT(*) AS "Numero_Animales"
    FROM animal
    GROUP BY establo
) a ON e.IDestablo = a.establo;
```

```
MariaDB [Proyecto]> SELECT e.IDestablo, e.tipo AS "Tipo_Establo", a.Tipo_Animal, a.Numero_Animales
    -> FROM establo e
    -> JOIN (
    ->     SELECT establo, MIN(tipo) AS "Tipo_Animal", COUNT(*) AS "Numero_Animales"
    ->     FROM animal
    ->     GROUP BY establo
    -> ) a ON e.IDestablo = a.establo;
+-----------+--------------+-------------+-----------------+
| IDestablo | Tipo_Establo | Tipo_Animal | Numero_Animales |
+-----------+--------------+-------------+-----------------+
|        11 | Grande       | Vaca        |               4 |
|        12 | Grande       | Vaca        |               3 |
|        13 | Grande       | Cerdo       |               2 |
|        14 | Grande       | Gallina     |               2 |
|        15 | Grande       | Pescado     |               2 |
|        26 | Grande       | Vaca        |               2 |
|        27 | Grande       | Cerdo       |               2 |
|        28 | Grande       | Gallina     |               2 |
|        29 | Grande       | Pescado     |               2 |
|       210 | Grande       | Vaca        |               2 |
|       311 | Pequeño      | Cerdo       |               2 |
|       312 | Mediano      | Gallina     |               2 |
|       313 | Grande       | Cerdo       |               3 |
|       414 | Pequeño      | Gallina     |               3 |
|       415 | Mediano      | Cerdo       |               3 |
|       516 | Pequeño      | Gallina     |               2 |
+-----------+--------------+-------------+-----------------+
16 rows in set (0.020 sec)
```

7. Se necesita consultar la cantidad de compras que ha hecho cada cliente

```
SELECT cedula, clasificacion, IFNULL(cantidad_compras, 0) AS cantidad_compras
FROM cliente
LEFT JOIN (
    SELECT cedula_cliente, COUNT(*) AS cantidad_compras
    FROM compra
    GROUP BY cedula_cliente
) AS compras_por_cliente ON cliente.cedula = compras_por_cliente.cedula_cliente;
```

```
MariaDB [Proyecto]> SELECT cedula, clasificacion, IFNULL(cantidad_compras, 0) AS cantidad_compras
    -> FROM cliente
    -> LEFT JOIN (
    ->     SELECT cedula_cliente, COUNT(*) AS cantidad_compras
    ->     FROM compra
    ->     GROUP BY cedula_cliente
    -> ) AS compras_por_cliente ON cliente.cedula = compras_por_cliente.cedula_cliente;
+-----------+--------------+------------------+
| cedula    | clasificacion | cantidad_compras |
+-----------+--------------+------------------+
|      1346 | Minorista    |                0 |
|      1365 | Minorista    |                0 |
|     63210 | Minorista    |                0 |
|   1002578 | Minorista    |                5 |
|   1005478 | Minorista    |                0 |
|   1007345 | Minorista    |                5 |
|   1097584 | Minorista    |                0 |
|   1098523 | Minorista    |                5 |
|   1098543 | Minorista    |                0 |
| 222222222 | Minorista    |               10 |
+-----------+--------------+------------------+
10 rows in set (0.001 sec)
```

8. consultar cantidad disponible de huevos por su clasificación y tipo

```
SELECT cf.clasificacion AS "Clasificacion",
    cf.tipoHuevo AS "Tipo_Huevo",
    SUM(cf.unidades) AS "Cantidad_Disponible"
FROM CartonHuevo cf
LEFT JOIN loteProduccion lp ON cf.IDalimento = lp.IDlote
WHERE lp.estado IS NULL OR lp.estado = "En punto de venta"
GROUP BY cf.clasificacion, cf.tipoHuevo;
```

```
MariaDB [Proyecto]> SELECT cf.clasificacion AS "Clasificacion",
    ->     cf.tipoHuevo AS "Tipo_Huevo",
    ->     SUM(cf.unidades) AS "Cantidad_Disponible"
    -> FROM CartonHuevo cf
    -> LEFT JOIN loteProduccion lp ON cf.IDalimento = lp.IDlote
    -> WHERE lp.estado IS NULL OR lp.estado = "En punto de venta"
    -> GROUP BY cf.clasificacion, cf.tipoHuevo;
+---------------+-------------+----------------------+
| Clasificacion | Tipo_Huevo  | Cantidad_Disponible  |
+---------------+-------------+----------------------+
| A             | Criollo     |                   75 |
| AA            | Criollo     |                   15 |
| AA            | tradicional |                   30 |
| AAA           | criollo     |                   38 |
| AAA           | Tradicional |                    8 |
| B             | tradicional |                   61 |
+---------------+-------------+----------------------+
6 rows in set (0.001 sec)
```

9. Consultar los vendedores junto con su horario y salida y la dirección de la tienda en la que fueron asignados.

```
SELECT vendedor.cedula, vendedor.HoraEntrada, vendedor.HoraSalida,
   (SELECT direccion FROM puntoVenta WHERE IDtienda = vendedor.tienda) AS direccion_tienda
FROM vendedor;
```

```
MariaDB [Proyecto]> SELECT vendedor.cedula, vendedor.HoraEntrada, vendedor.HoraSalida,
    ->     (SELECT direccion FROM puntoVenta WHERE IDtienda = vendedor.tienda) AS direccion_tienda
    -> FROM vendedor;
+---------+-------------+-----------+------------------+
| cedula  | HoraEntrada | HoraSalida | direccion_tienda |
+---------+-------------+-----------+------------------+
|   63142 | 08:30:00    | 17:30:00  | Carrera 15wa     |
|   65781 | 12:00:00    | 17:00:00  | Calle 52 #16-88  |
|  137245 | 14:30:00    | 20:30:00  | Calle 46 #32-21  |
|  196020 | 08:00:00    | 13:00:00  | Calle 46 #32-21  |
| 1006447 | 10:30:00    | 19:30:00  | Carrera 238v     |
| 1009645 | 09:00:00    | 18:00:00  | Calle 23 #34-23  |
| 1098537 | 06:00:00    | 12:00:00  | Calle 52 #16-88  |
+---------+-------------+-----------+------------------+
7 rows in set (0.001 sec)
```

10. Consultar los tipos de carnes junto con su corte que tiene cada lote de produccion (si es que tienen)

```
SELECT la.IDlote AS ID_lote_produccion,
    (SELECT tipo FROM carne WHERE IDalimento = la.IDalimento) AS tipo_carne,
    (SELECT corte FROM carne WHERE IDalimento = la.IDalimento) AS tipo_corte
FROM lote_alimento la
WHERE EXISTS (SELECT * FROM carne WHERE IDalimento = la.IDalimento);
```

```
MariaDB [Proyecto]> SELECT la.IDlote AS ID_lote_produccion,
    ->        (SELECT tipo FROM carne WHERE IDalimento = la.IDalimento) AS tipo_carne,
    ->        (SELECT corte FROM carne WHERE IDalimento = la.IDalimento) AS tipo_corte
    -> FROM lote_alimento la
    -> WHERE EXISTS (SELECT * FROM carne WHERE IDalimento = la.IDalimento);
+--------------------+-------------+------------+
| ID_lote_produccion | tipo_carne  | tipo_corte |
+--------------------+-------------+------------+
|               1007 | Carne res   | Filete     |
|               1008 | Carne res   | Costilla   |
|               1003 | Carne res   | Lomo       |
|               1004 | Carne res   | Chuletón   |
|               1005 | Carne res   | Solomillo  |
|               1006 | Carne res   | Picaña     |
|               1001 | Carne cerdo | Lomo       |
|               1002 | Carne cerdo | Costilla   |
|               1003 | Carne cerdo | Jamón      |
|               1004 | Carne cerdo | Panceta    |
|               1001 | Gallina     | Pechuga    |
|               1008 | Gallina     | Pechuga    |
|               1002 | Gallina     | Muslo      |
|               1003 | Gallina     | Ala        |
|               1004 | Gallina     | Cadera     |
|               1005 | Gallina     | Espalda    |
|               1006 | Gallina     | Pata       |
|               1007 | Gallina     | Cuello     |
|               1008 | Gallina     | Hígado     |
|               1001 | Pescado     | Filete     |
|               1002 | Pescado     | Lomo       |
|               1003 | Pescado     | Cola       |
|               1004 | Pescado     | Rodaja     |
|               1005 | Pescado     | Cabeza     |
|               1006 | Pescado     | Vísceras   |
+--------------------+-------------+------------+
```

UNIVERSIDAD COOPERATIVA DE COLOMBIA

Gestión de base de datos

1.Documentacion de Roles y permisos

Almacen_Datos: Este rol será el destinado para usar por el administrador de la base de datos, por lo tanto, contará con todos los privilegios existentes. Incluido contará con la opción Grant option con la cual podrá otorgar o revocar privilegios a los demás usuarios. Como dijimos este rol está destinado a ser usador por el administrador de la base de datos, lo recomendado seria que solo un usuario haga uso de ese rol ya que poseerá muchos privilegios que si se les da un mal uso podría dañar o borrar la base de datos.

```
MariaDB [(none)]> CREATE ROLE Almacen_Datos;
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]>
MariaDB [(none)]> grant all
    -> on Proyecto.*
    -> to Almacen_Datos
    -> with grant option;
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]>
MariaDB [(none)]> grant all
    -> on AuditoriaProyecto.*
    -> to Almacen_Datos
    -> with grant option;
Query OK, 0 rows affected (0.002 sec)
```

Desarrollador: Como su mismo nombre lo indica este rol está diseñado para ser usado por los desarrolladores de software que trabajan dentro de la base de datos. Por lo tanto, es un rol que contara con los privilegios de las tablas (alter, create, delete, drop, index, insert, select, update) dentro de la base de datos Proyecto (todas las tablas). Dentro de la base de datos de la auditoria contara con los permisos de crear tablas y seleccionar datos. Este rol puede ser usado por los N desarrolladores que participan dentro del proyecto.

```
MariaDB [(none)]> CREATE ROLE Desarrollador;
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]>
MariaDB [(none)]> grant ALTER, CREATE, DELETE, DROP, INDEX, INSERT, SELECT, UPDATE
    ->  on Proyecto.*
    ->  to Desarrollador;
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]>
MariaDB [(none)]> grant CREATE, SELECT
    -> on AuditoriaProyecto.*
    -> to Desarrollador;
Query OK, 0 rows affected (0.002 sec)
```

APP_Principal: Este rol será el destinado a usar a todos los empleados de la empresa que usan la base de datos. Por lo tanto, únicamente podrán agregar, modificar, eliminar y seleccionar datos dentro de las tablas. Mas no podrán modificar ninguna tabla. Tampoco tendrán acceso a la base de datos de Auditoria Proyecto.

```
MariaDB [(none)]> CREATE ROLE APP_Principal;
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]>
MariaDB [(none)]> grant INSERT, UPDATE, DELETE, SELECT
    -> on Proyecto.*
    -> to APP_Principal;
Query OK, 0 rows affected (0.002 sec)
```

Auditor: Por último, contamos con un auditor, por lo general dentro de las empresas el auditor es un ente externo que cada cierto tiempo realiza auditoria en los procesos de la base de datos. Por lo tanto, el auditor únicamente contara con la opción de ver (Select) los datos dentro de la base de datos Proyecto y la base de datos AuditoriaProyecto mas no tiene poder ni de insertar ni modificar las tablas o los datos.

```
MariaDB [(none)]> CREATE ROLE Auditor;
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]>
MariaDB [(none)]> grant SELECT
    -> on Proyecto.*
    -> to Auditor;
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]>
MariaDB [(none)]> grant SELECT
    -> on AuditoriaProyecto.*
    -> to Auditor;
Query OK, 0 rows affected (0.003 sec)
```

2.Documentacion de Usuarios

Administrador: El usuario Administrador será aquel que se le asigne el rol Almacen_Datos y por lo tanto obtendrá todos los privilegios disponibles. Así como la capacidad de asignar o quitar privilegios a los demás usuarios. Por temas de seguridad solo tendremos un desarrollador.

Usuario: Administrador - Contraseña: A1823m

```
MariaDB [(none)]> CREATE USER 'Administrador'@'localhost' IDENTIFIED BY 'A1823m';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> GRANT Almacen_Datos TO 'Administrador'@'localhost';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> SET DEFAULT ROLE Almacen_Datos FOR 'Administrador'@'localhost';
Query OK, 0 rows affected (0.003 sec)
```

Desarrolladores: A diferencia del administrador aquí podemos crear todos los desarrolladores que queramos y se les otorgara el rol de desarrollador creado anteriormente.

Usuario: Daniel - Contraseña: 1

```
MariaDB [(none)]> CREATE USER 'Daniel'@'localhost' IDENTIFIED BY '1';
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]> GRANT Desarrollador TO 'Daniel'@'localhost';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> SET DEFAULT ROLE Desarrollador FOR 'Daniel'@'localhost';
Query OK, 0 rows affected (0.002 sec)
```

Usuario: Yeimy - Contraseña: 2

```
MariaDB [(none)]> CREATE USER 'Yeimy'@'localhost' IDENTIFIED BY '2';
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]> GRANT Desarrollador TO 'Yeimy'@'localhost';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> SET DEFAULT ROLE Desarrollador FOR 'Yeimy'@'localhost';
Query OK, 0 rows affected (0.002 sec)
```

Usuario: Gustavo - Contraseña: 3

```
MariaDB [(none)]> CREATE USER 'Gustavo'@'localhost' IDENTIFIED BY '3';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> GRANT Desarrollador TO 'Gustavo'@'localhost';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> SET DEFAULT ROLE Desarrollador FOR 'Gustavo'@'localhost';
Query OK, 0 rows affected (0.003 sec)
```

Empleado: En empleado crearemos a todos los empleados de la empresa que harán uso de la base de datos, se les otorgara el rol de APP_Principal.

Usuario: Empleado - Contraseña: A12024

```
MariaDB [(none)]> CREATE USER 'Empleado'@'localhost' IDENTIFIED BY 'A12024';
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]> GRANT APP_Principal TO 'Empleado'@'localhost';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> SET DEFAULT ROLE APP_Principal FOR 'Empleado'@'localhost';
Query OK, 0 rows affected (0.003 sec)
```

Auditor: El auditor será aquel que realice auditorias periódicas, por lo que se le otorgará el rol de su mismo nombre “Auditor”.

Usuario: Auditor - Contraseña: 123

```
MariaDB [(none)]> CREATE USER 'Auditor'@'localhost' IDENTIFIED BY '123';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> GRANT Auditor TO 'Auditor'@'localhost';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> SET DEFAULT ROLE Auditor FOR 'Auditor'@'localhost';
Query OK, 0 rows affected (0.004 sec)
```

Pruebas

Prueba Administrador:

```
PS C:\Users\danie> mysql -u Administrador -p
Enter password: ******
MariaDB [Proyecto]> use auditoriaProyecto;
Database changed
MariaDB [auditoriaProyecto]> use Proyecto;
Database changed
```

Lo primero que podemos observar es que podemos movernos libremente en ambas bases de datos

```
MariaDB [Proyecto]> CREATE TABLE TablaPrueba (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     atributo VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.009 sec)

MariaDB [Proyecto]> INSERT INTO TablaPrueba (atributo) VALUES ('Dato de prueba');
Query OK, 1 row affected (0.003 sec)

MariaDB [Proyecto]> UPDATE TablaPrueba SET atributo = 'Nuevo dato' WHERE id = 1;
Query OK, 1 row affected (0.003 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [Proyecto]> DELETE FROM TablaPrueba WHERE id = 1;
Query OK, 1 row affected (0.003 sec)

MariaDB [Proyecto]> DROP TABLE TablaPrueba;
Query OK, 0 rows affected (0.011 sec)
```

Tambien podemos obersvar que tiene todos los privilegios en funcionamiento, esto debido a que le asignamos el ALL PRIVILEGES.

Prueba Desarrollador (Daniel):

```
MariaDB [(none)]> use auditoriaProyecto;
Database changed
MariaDB [auditoriaProyecto]> CREATE TABLE TablaPrueba (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     atributo VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.011 sec)

MariaDB [auditoriaProyecto]> DROP TABLE TablaPrueba;
ERROR 1142 (42000): DROP command denied to user 'Daniel'@'localhost' for table `auditoriaproyecto`.`tablaprueba`
MariaDB [auditoriaProyecto]> select * from TablaPrueba;
Empty set (0.001 sec)
```

Como podemos ver en un usuario con Rol Desarrollador podemos tener acceso a la BD auditoriaProyecto, en el cual podemos crear tablas, mas no podremos borrarla ya que el desarrollador solo tiene permisos de “CREATE” y “SELECT” en la base de datos auditoriaProyecto.

```
MariaDB [Proyecto]> CREATE TABLE TablaPrueba (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     atributo VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.010 sec)

MariaDB [Proyecto]> INSERT INTO TablaPrueba (atributo) VALUES ('Dato de prueba');
Query OK, 1 row affected (0.001 sec)

MariaDB [Proyecto]> UPDATE TablaPrueba SET atributo = 'Nuevo dato' WHERE id = 1;
Query OK, 1 row affected (0.001 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [Proyecto]> SELECT * FROM TablaPrueba;
+----+------------+
| id | atributo   |
+----+------------+
|  1 | Nuevo dato |
+----+------------+
1 row in set (0.000 sec)

MariaDB [Proyecto]> DELETE FROM TablaPrueba WHERE id = 1;
Query OK, 1 row affected (0.001 sec)

MariaDB [Proyecto]> DROP TABLE TablaPrueba;
Query OK, 0 rows affected (0.011 sec)
```

Por otro lado en la base de datos Proyecto tenemos todos los permisos relacionados con las tablas (CREATE, INSERT, UPDATE, SELECT, DELETE, DROP).

Prueba Empleado:

```
PS C:\Users\danie> mysql -u Empleado -p
Enter password: ******
```

```
MariaDB [(none)]> use auditoriaProyecto;
ERROR 1044 (42000): Access denied for user 'Empleado'@'localhost' to database 'auditoriaproyecto'
```

Lo primero que podemos observar es que Empleado no tiene acceso a auditoriaProyecto.

```
MariaDB [(none)]> use proyecto;
Database changed
MariaDB [proyecto]> CREATE TABLE TablaPrueba (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     atributo VARCHAR(50)
    -> );
ERROR 1142 (42000): CREATE command denied to user 'Empleado'@'localhost' for table `proyecto`.`tablaprueba`
MariaDB [proyecto]>
```

De igual manera vemos que no poseemos la capacidad de crear tablas.

```
MariaDB [proyecto]> ALTER TABLE persona MODIFY direccion VARCHAR(150);
ERROR 1142 (42000): ALTER command denied to user 'Empleado'@'localhost' for table `proyecto`.`persona`
MariaDB [proyecto]> DROP TABLE persona;
ERROR 1142 (42000): DROP command denied to user 'Empleado'@'localhost' for table `proyecto`.`persona`
```

Tampoco poseeremos la capacidad de alterar o borrar las tablas.

```
MariaDB [proyecto]> INSERT INTO Persona (Cedula, Nombre, FechaNac, Edad, Direccion, Genero) VALUES
    -> (1, "Prueba", NULL, NULL, NULL, NUll);
Query OK, 1 row affected (0.003 sec)

MariaDB [proyecto]> UPDATE Persona
    -> SET Nombre = 'NuevoNombre'
    -> WHERE Cedula = 1;
Query OK, 1 row affected (0.003 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [proyecto]> DELETE FROM Persona WHERE Cedula = 1;
Query OK, 1 row affected (0.003 sec)

MariaDB [proyecto]> select * from persona;
+-----------+--------------------+------------+-------+-----------------------+---------+
| cedula    | nombre             | fechaNac   | edad | direccion             | genero |
+-----------+--------------------+------------+-------+-----------------------+---------+
|      1346 | Carlos Martinez    | 1982-06-30 |   42 | Calle 50 #28-35       | M      |
|      1365 | Miguel Gonzalez    | 1972-09-08 |   52 | Calle 55 #12-50       | M      |
```

Por el contrario, si tenemos la capacidad de usar el INSERT, UPDATE, DELETE y SELECT.

Prueba Auditor:

```
PS C:\Users\danie> mysql -u Auditor -p
Enter password: ***
```

```
MariaDB [AuditoriaProyecto]> CREATE TABLE TablaPrueba (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     atributo VARCHAR(50)
    -> );
ERROR 1142 (42000): CREATE command denied to user 'Auditor'@'localhost' for table `auditoriaproyecto`.`tablaprueba`
MariaDB [AuditoriaProyecto]> INSERT INTO TablaPrueba (atributo) VALUES ('Dato de prueba');
ERROR 1142 (42000): INSERT command denied to user 'Auditor'@'localhost' for table `auditoriaproyecto`.`tablaprueba`
MariaDB [AuditoriaProyecto]> UPDATE TablaPrueba SET atributo = 'Nuevo dato' WHERE id = 1;
ERROR 1142 (42000): UPDATE command denied to user 'Auditor'@'localhost' for table `auditoriaproyecto`.`tablaprueba`
MariaDB [AuditoriaProyecto]> DELETE FROM TablaPrueba WHERE id = 1;
ERROR 1142 (42000): DELETE command denied to user 'Auditor'@'localhost' for table `auditoriaproyecto`.`tablaprueba`
MariaDB [AuditoriaProyecto]> DROP TABLE TablaPrueba;
ERROR 1142 (42000): DROP command denied to user 'Auditor'@'localhost' for table `auditoriaproyecto`.`tablaprueba`
```

En el auditor no tenemos derecho a usar ningun comando relacionado con las tablas, excepto por el SELECT

```
MariaDB [AuditoriaProyecto]> select * from auditoriaPersona;
+-------------+---------------------+----------+---------+--------------------+-----------+
| IDauditoria | fecha               | IDobjeto | Tabla   | Usuario            | Operacion |
+-------------+---------------------+----------+---------+--------------------+-----------+
|           1 | 2024-05-23 15:55:16 | 12345678 | Persona | DanielJay@localhost | INSERT    |
|           2 | 2024-05-23 15:55:16 | 12345678 | Persona | DanielJay@localhost | UPDATE    |
|           3 | 2024-05-23 15:55:16 | 12345678 | Persona | DanielJay@localhost | DELETE    |
|           4 | 2024-05-23 15:55:53 | 12345678 | Persona | DanielJay@localhost | INSERT    |
|           5 | 2024-05-23 15:55:53 | 12345678 | Persona | DanielJay@localhost | UPDATE    |
```

De igual manera en la BD proyecto solo podemos usar el select en todas las tablas.

NOTA: CREACION DE ROLES, USUARIOS Y ASIGNACION DE ROLES A USUARIOS HACERLOS DESDE USUARIO ROOT.

3.Glosario de Permisos

Grant All: Conceden todos los privilegios dentro de una base de datos especifica. Este permiso solo lo poseerá el rol **Almacen_Datos.**

Grant Option: Permite conceder o revocar permisos a los demás usuarios. Este permiso solo lo poseerá el rol **Almacen_Datos.**

Create: Permite crear tablas dentro de la base de datos. Permiso que tendrán los roles **Almacen_Datos** y **Desarrollador.**

Alter: Permite modificar tablas dentro de la base de datos. Permiso que tendrán los roles **Almacen_Datos** y **Desarrollador.**

Drop: Permite Borrar tablas dentro de la base de datos. Permiso que tendrán los roles **Almacen_Datos** y **Desarrollador.**

Index: Permite crear índices dentro de las tablas en la base de datos. Permiso que tendrán los roles **Almacen_Datos** y **Desarrollador.**

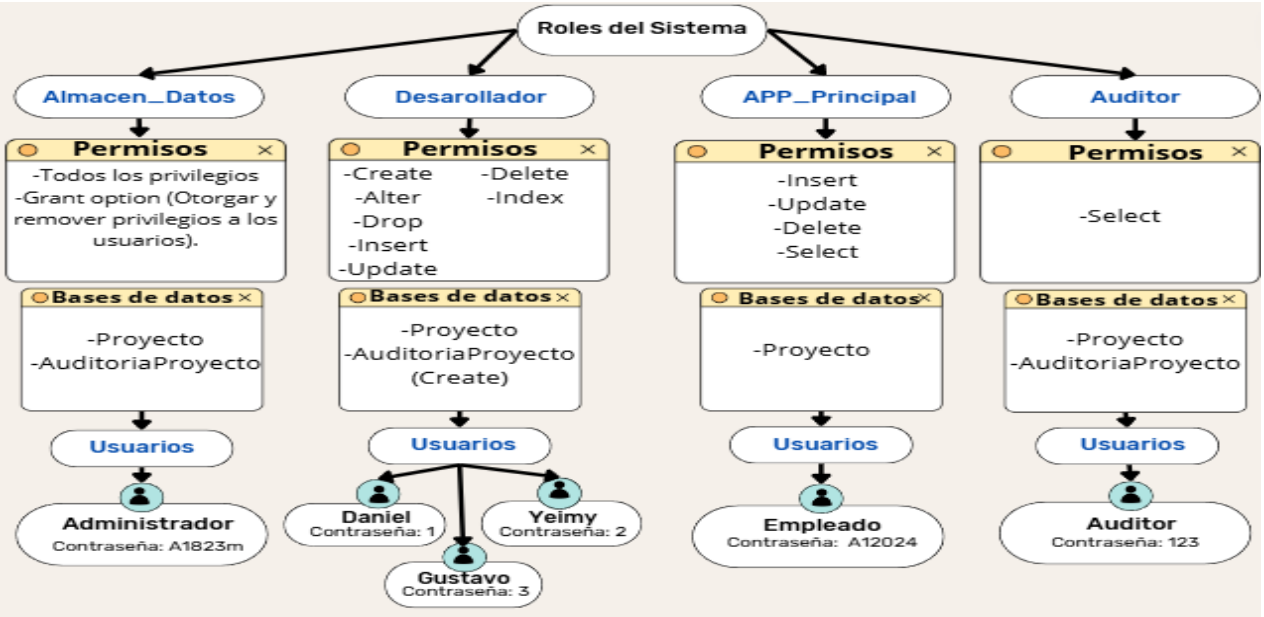
Insert: Permite insertar datos dentro de las tablas en la base de datos. Permiso que tendrán los roles **Almacen_Datos**, **Desarrollador** y **APP_Principal.**

Update: Permite Actualizar los datos dentro de las tablas en la base de datos. Permiso que tendrán los roles **Almacen_Datos**, **Desarrollador** y **APP_Principal.**

Delete: Permite borrar los datos dentro de las tablas en la base de datos. Permiso que tendrán los roles **Almacen_Datos**, **Desarrollador** y **APP_Principal.**

Select: Permite visualizar los datos dentro de las tablas en la base de datos. Permiso que tendrán los roles **Almacen_Datos, Desarrollador, APP_Principal** y **Auditor.**

4.Esquema de roles y usuarios

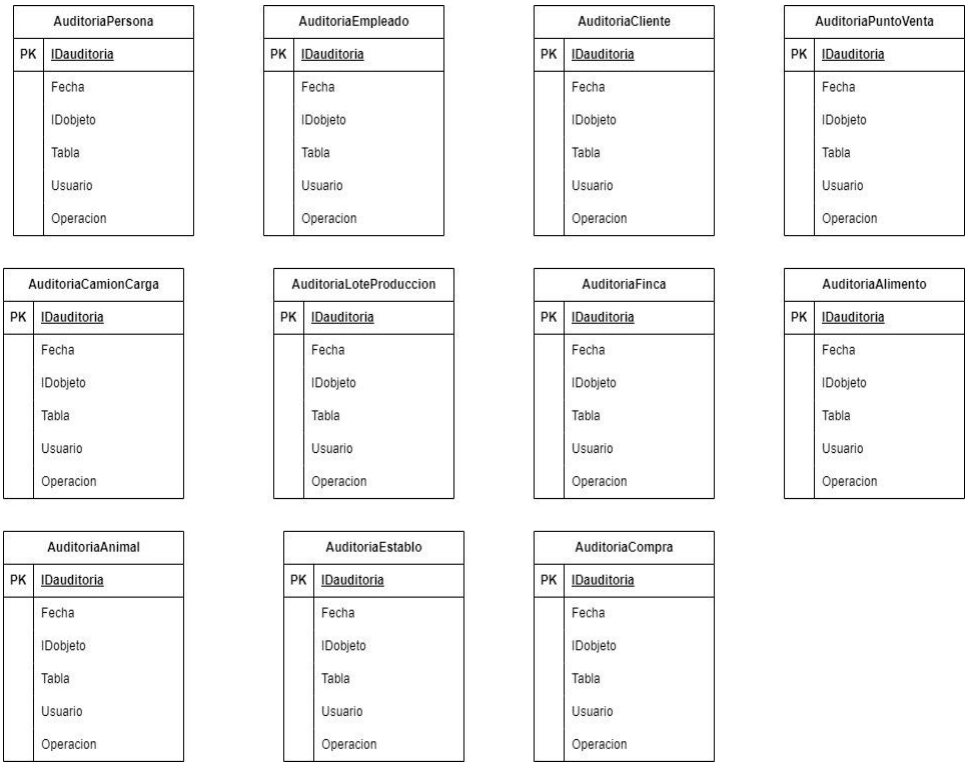


UNIVERSIDAD COOPERATIVA DE COLOMBIA

Creación de modelo de auditoria

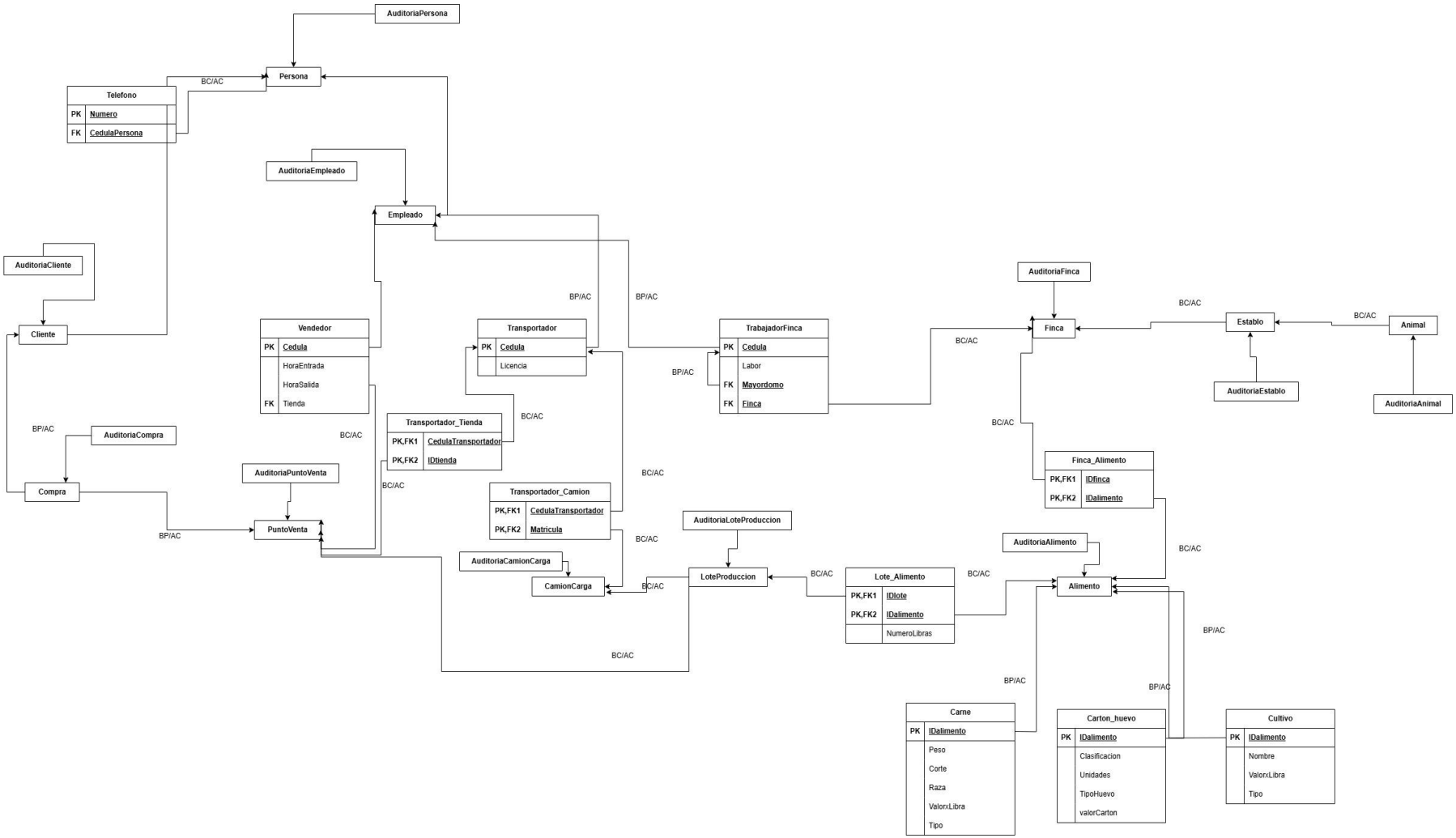
1. Esquema gráfico de auditoria

1. Esquema Relacional individual



Cada tabla de auditoria guardara los datos insertados, modificados y eliminados de la tabla a la cual su nombre hace referencia, exceptuando la AuditoriaEmpleado y la AuditoriaAlimento. En estas dos tablas también se guardarán los datos insertados, modificados y eliminados también de sus respectivos subtipos.

Esquema Relacional conjunto



2. Documentación creación de modelo

1. Creación de base de datos

```
MariaDB [Proyecto]> create database auditoriaProyecto;
Query OK, 1 row affected (0.002 sec)
```

2. Creación de tablas

Auditoria tabla Persona:

```
MariaDB [proyecto]> CREATE TABLE auditoriaPersona (
    ->  IDauditoria INT AUTO_INCREMENT PRIMARY KEY,
    ->  fecha DATETIME NOT NULL,
    ->  IDobjeto INT,
    ->  Tabla VARCHAR(30),
    ->  Usuario VARCHAR(100),
    ->  Operacion CHAR(6) NOT NULL,
    -> CHECK(Operacion = "INSERT" or Operacion="DELETE" or Operacion="UPDATE"));
Query OK, 0 rows affected (0.010 sec)
```

Auditoria tabla Cliente:

```
MariaDB [proyecto]> CREATE TABLE auditoriaEmpleado (
    ->  IDauditoria INT AUTO_INCREMENT PRIMARY KEY,
    ->  fecha DATETIME NOT NULL,
    ->  IDobjeto INT,
    ->  Tabla VARCHAR(30),
    ->  Usuario VARCHAR(100),
    ->  Operacion CHAR(6) NOT NULL,
    -> CHECK(Operacion = "INSERT" or Operacion="DELETE" or Operacion="UPDATE"));
Query OK, 0 rows affected (0.009 sec)
```

Auditoria tabla Empleado:

```
MariaDB [proyecto]> CREATE TABLE auditoriaEmpleado (
    ->  IDauditoria INT AUTO_INCREMENT PRIMARY KEY,
    ->  fecha DATETIME NOT NULL,
    ->  IDobjeto INT,
    ->  Tabla VARCHAR(30),
    ->  Usuario VARCHAR(100),
    ->  Operacion CHAR(6) NOT NULL,
    -> CHECK(Operacion = "INSERT" or Operacion="DELETE" or Operacion="UPDATE"));
Query OK, 0 rows affected (0.009 sec)
```

Auditoria tabla Compra:

```
MariaDB [proyecto]> CREATE TABLE auditoriaCompra(
    ->  IDauditoria INT AUTO_INCREMENT PRIMARY KEY,
    ->  fecha DATETIME NOT NULL,
    ->  IDobjeto INT,
    ->  Tabla VARCHAR(30),
    ->  Usuario VARCHAR(100),
    ->  Operacion CHAR(6) NOT NULL,
    -> CHECK(Operacion = "INSERT" or Operacion="DELETE" or Operacion="UPDATE"));
Query OK, 0 rows affected (0.006 sec)
```

Auditoria tabla PuntoVenta:

```
MariaDB [proyecto]> CREATE TABLE auditoriaPuntoVenta (
    ->  IDauditoria INT AUTO_INCREMENT PRIMARY KEY,
    ->  fecha DATETIME NOT NULL,
    ->  IDobjeto INT,
    ->  Tabla VARCHAR(30),
    ->  Usuario VARCHAR(100),
    ->  Operacion CHAR(6) NOT NULL,
    -> CHECK(Operacion = "INSERT" or Operacion="DELETE" or Operacion="UPDATE"));
Query OK, 0 rows affected (0.010 sec)
```

Auditoria tabla CamionCarga:

```
MariaDB [proyecto]> CREATE TABLE auditoriaCamionCarga (
    ->  IDauditoria INT AUTO_INCREMENT PRIMARY KEY,
    ->  fecha DATETIME NOT NULL,
    ->  IDobjeto INT,
    ->  Tabla VARCHAR(30),
    ->  Usuario VARCHAR(100),
    ->  Operacion CHAR(6) NOT NULL,
    -> CHECK(Operacion = "INSERT" or Operacion="DELETE" or Operacion="UPDATE"));
Query OK, 0 rows affected (0.009 sec)
```

Auditoria tabla LoteProduccion:

```
MariaDB [proyecto]> CREATE TABLE auditoriaLoteProduccion (
    ->  IDauditoria INT AUTO_INCREMENT PRIMARY KEY,
    ->  fecha DATETIME NOT NULL,
    ->  IDobjeto INT,
    ->  Tabla VARCHAR(30),
    ->  Usuario VARCHAR(100),
    ->  Operacion CHAR(6) NOT NULL,
    -> CHECK(Operacion = "INSERT" or Operacion="DELETE" or Operacion="UPDATE"));
Query OK, 0 rows affected (0.008 sec)
```

Auditoria tabla Finca:

```
MariaDB [proyecto]> CREATE TABLE auditoriaFinca (
    ->  IDauditoria INT AUTO_INCREMENT PRIMARY KEY,
    ->  fecha DATETIME NOT NULL,
    ->  IDobjeto INT,
    ->  Tabla VARCHAR(30),
    ->  Usuario VARCHAR(100),
    ->  Operacion CHAR(6) NOT NULL,
    -> CHECK(Operacion = "INSERT" or Operacion="DELETE" or Operacion="UPDATE"));
Query OK, 0 rows affected (0.009 sec)
```

Auditoria tabla Alimento:

```
MariaDB [proyecto]> CREATE TABLE auditoriaAlimento (
    ->  IDauditoria INT AUTO_INCREMENT PRIMARY KEY,
    ->  fecha DATETIME NOT NULL,
    ->  IDobjeto INT,
    ->  Tabla VARCHAR(30),
    ->  Usuario VARCHAR(100),
    ->  Operacion CHAR(6) NOT NULL,
    -> CHECK(Operacion = "INSERT" or Operacion="DELETE" or Operacion="UPDATE"));
Query OK, 0 rows affected (0.006 sec)
```

Auditoria tabla Establo:

```
MariaDB [proyecto]> CREATE TABLE auditoriaEstablo (
    ->  IDauditoria INT AUTO_INCREMENT PRIMARY KEY,
    ->  fecha DATETIME NOT NULL,
    ->  IDobjeto INT,
    ->  Tabla VARCHAR(30),
    ->  Usuario VARCHAR(100),
    ->  Operacion CHAR(6) NOT NULL,
    -> CHECK(Operacion = "INSERT" or Operacion="DELETE" or Operacion="UPDATE"));
Query OK, 0 rows affected (0.007 sec)
```

Auditoria tabla Animal:

```
MariaDB [proyecto]> CREATE TABLE auditoriaAnimal  (
    ->  IDauditoria INT AUTO_INCREMENT PRIMARY KEY,
    ->  fecha DATETIME NOT NULL,
    ->  IDobjeto INT,
    ->  Tabla VARCHAR(30),
    ->  Usuario VARCHAR(100),
    ->  Operacion CHAR(6) NOT NULL,
    -> CHECK(Operacion = "INSERT" or Operacion="DELETE" or Operacion="UPDATE"));
Query OK, 0 rows affected (0.007 sec)
```

Auditoria tabla Establo:

```
MariaDB [proyecto]> CREATE TABLE auditoriaEstablo (
    ->  IDauditoria INT AUTO_INCREMENT PRIMARY KEY,
    ->  fecha DATETIME NOT NULL,
    ->  IDobjeto INT,
    ->  Tabla VARCHAR(30),
    ->  Usuario VARCHAR(100),
    ->  Operacion CHAR(6) NOT NULL,
    -> CHECK(Operacion = "INSERT" or Operacion="DELETE" or Operacion="UPDATE"));
Query OK, 0 rows affected (0.007 sec)
```

Auditoria tabla Animal:

```
MariaDB [proyecto]> CREATE TABLE auditoriaAnimal  (
    ->  IDauditoria INT AUTO_INCREMENT PRIMARY KEY,
    ->  fecha DATETIME NOT NULL,
    ->  IDobjeto INT,
    ->  Tabla VARCHAR(30),
    ->  Usuario VARCHAR(100),
    ->  Operacion CHAR(6) NOT NULL,
    -> CHECK(Operacion = "INSERT" or Operacion="DELETE" or Operacion="UPDATE"));
Query OK, 0 rows affected (0.007 sec)
```

3. Creación de triggers.

Triggers auditoriaPersona: Dentro de esta auditoria solo se guardará los datos insertados/modificados/eliminados de la tabla persona.

```
MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Insertar_Persona
    -> AFTER INSERT ON proyecto.Persona
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaPersona (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.cedula,"Persona", USER(), "INSERT");
    -> END;
    -> //
Query OK, 0 rows affected (0.043 sec)

MariaDB [Proyecto]>
MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Actualizar_Persona
    -> AFTER UPDATE ON proyecto.Persona
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaPersona (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.cedula, "Persona", USER(), "UPDATE");
    -> END;
    -> //
Query OK, 0 rows affected (0.024 sec)

MariaDB [Proyecto]>
MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Borrar_Persona
    -> AFTER DELETE ON proyecto.Persona
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaPersona (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), OLD.cedula, "Persona", USER(), "DELETE");
    -> END;
    -> //
Query OK, 0 rows affected (0.039 sec)
```

Triggers auditoriaCliente: Dentro de esta auditoria solo se guardará los datos insertados/modificados/eliminados de la tabla Cliente.

```
MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Insertar_Cliente
    -> AFTER INSERT ON proyecto.Cliente
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaCliente   (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.cedula,"Cliente", USER(), "INSERT");
    -> END;
    -> //
Query OK, 0 rows affected (0.035 sec)

MariaDB [Proyecto]>
MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Actualizar_Cliente
    -> AFTER UPDATE ON proyecto.Cliente
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaCliente   (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.cedula, "Cliente", USER(), "UPDATE");
    -> END;
    -> //
Query OK, 0 rows affected (0.022 sec)

MariaDB [Proyecto]>
MariaDB [Proyecto]>
MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Borrar_Cliente
    -> AFTER DELETE ON proyecto.Cliente
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaCliente   (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), OLD.cedula, "Cliente", USER(), "DELETE");
    -> END;
    -> //
Query OK, 0 rows affected (0.021 sec)
```

Triggers auditoriaEmpleado: Dentro de esta auditoria solo se guardará los datos insertados/modificados/eliminados de las tablas: Empleado/Vendedor/Transportador/TrabajadorFinca.

Empleado:

```
MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Insertar_Empleado
    -> AFTER INSERT ON proyecto.Empleado
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaEmpleado  (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.cedula,"Empleado", USER(), "INSERT");
    -> END;
    -> //
Query OK, 0 rows affected (0.034 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Actualizar_Empleado
    -> AFTER UPDATE ON proyecto.Empleado
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaEmpleado  (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.cedula, "Empleado", USER(), "UPDATE");
    -> END;
    -> //
Query OK, 0 rows affected (0.022 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Borrar_Empleado
    -> AFTER DELETE ON proyecto.Empleado
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaEmpleado  (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), OLD.cedula, "Empleado", USER(), "DELETE");
    -> END;
    -> //
Query OK, 0 rows affected (0.040 sec)
```

Vendedor:

```
MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Insertar_Vendedor
    -> AFTER INSERT ON proyecto.Vendedor
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaEmpleado (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.cedula,"Vendedor", USER(), "INSERT");
    -> END;
    -> //
Query OK, 0 rows affected (0.038 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Actualizar_Vendedor
    -> AFTER UPDATE ON proyecto.Vendedor
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaEmpleado (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.cedula, "Vendedor", USER(), "UPDATE");
    -> END;
    -> //
Query OK, 0 rows affected (0.023 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Borrar_Vendedor
    -> AFTER DELETE ON proyecto.Vendedor
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaEmpleado (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), OLD.cedula, "Vendedor", USER(), "DELETE");
    -> END;
    -> //
Query OK, 0 rows affected (0.040 sec)
```

Transportador:

```
MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Insertar_Transportador
    -> AFTER INSERT ON proyecto.Transportador
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaEmpleado (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.cedula,"Transportador", USER(), "INSERT");
    -> END;
    -> //
Query OK, 0 rows affected (0.036 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Actualizar_Transportador
    -> AFTER UPDATE ON proyecto.Transportador
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaEmpleado (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.cedula, "Transportador", USER(), "UPDATE");
    -> END;
    -> //
Query OK, 0 rows affected (0.024 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Borrar_Transportador
    -> AFTER DELETE ON proyecto.Transportador
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaEmpleado  (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), OLD.cedula, "Transportador", USER(), "DELETE");
    -> END;
    -> //
Query OK, 0 rows affected (0.041 sec)
```

TrabajadorFinca:

```
MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Insertar_TrabajadorFinca
    -> AFTER INSERT ON proyecto.TrabajadorFinca
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaEmpleado   (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.cedula,"TrabajadorFinca", USER(), "INSERT");
    -> END;
    -> //
Query OK, 0 rows affected (0.036 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Actualizar_TrabajadorFinca
    -> AFTER UPDATE ON proyecto.TrabajadorFinca
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaEmpleado (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.cedula, "TrabajadorFinca", USER(), "UPDATE");
    -> END;
    -> //
Query OK, 0 rows affected (0.023 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Borrar_TrabajadorFinca
    -> AFTER DELETE ON proyecto.TrabajadorFinca
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaEmpleado  (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), OLD.cedula, "TrabajadorFinca", USER(), "DELETE");
    -> END;
    -> //
Query OK, 0 rows affected (0.043 sec)
```

Triggers auditoriaCompra: Dentro de esta auditoria solo se guardará los datos insertados/modificados/eliminados de la tabla Compra.

```
MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Insertar_Compra
    -> AFTER INSERT ON proyecto.Compra
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaCompra (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.IDfactura,"Compra", USER(), "INSERT");
    -> END;
    -> //
Query OK, 0 rows affected (0.035 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Actualizar_Compra
    -> AFTER UPDATE ON proyecto.Compra
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaCompra (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.IDfactura, "Compra", USER(), "UPDATE");
    -> END;
    -> //
Query OK, 0 rows affected (0.023 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Borrar_Compra
    -> AFTER DELETE ON proyecto.Compra
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaCompra (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), OLD.IDfactura, "Compra", USER(), "DELETE");
    -> END;
    -> //
Query OK, 0 rows affected (0.042 sec)
```

Triggers auditoriaPuntoventa: Dentro de esta auditoria solo se guardará los datos insertados/modificados/eliminados de la tabla PuntoVenta.

```
MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Insertar_PuntoVenta
    -> AFTER INSERT ON proyecto.PuntoVenta
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaPuntoVenta   (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.IDtienda,"PuntoVenta", USER(), "INSERT");
    -> END;
    -> //
Query OK, 0 rows affected (0.037 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Actualizar_PuntoVenta
    -> AFTER UPDATE ON proyecto.PuntoVenta
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaPuntoVenta  (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.IDtienda, "PuntoVenta", USER(), "UPDATE");
    -> END;
    -> //
Query OK, 0 rows affected (0.023 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Borrar_PuntoVenta
    -> AFTER DELETE ON proyecto.PuntoVenta
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaPuntoVenta  (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), OLD.IDtienda, "PuntoVenta", USER(), "DELETE");
    -> END;
    -> //
Query OK, 0 rows affected (0.041 sec)
```

Triggers auditoriaFinca: Dentro de esta auditoria solo se guardará los datos insertados/modificados/eliminados de la tabla Finca.

```
MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Insertar_Finca
    -> AFTER INSERT ON proyecto.Finca
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaFinca (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.IDfinca, "Finca", USER(), "INSERT");
    -> END;
    -> //
Query OK, 0 rows affected (0.034 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Actualizar_Finca
    -> AFTER UPDATE ON proyecto.Finca
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaFinca(fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.IDfinca, "Finca", USER(), "UPDATE");
    -> END;
    -> //
Query OK, 0 rows affected (0.025 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Borrar_Finca
    -> AFTER DELETE ON proyecto.Finca
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaFinca (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), OLD.IDfinca, "Finca", USER(), "DELETE");
    -> END;
    -> //
Query OK, 0 rows affected (0.024 sec)
```

Triggers auditoria Establo: Dentro de esta auditoria solo se guardará los datos insertados/modificados/eliminados de la tabla Establo.

```
MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Insertar_Establo
    -> AFTER INSERT ON proyecto.Establo
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaEstablo (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.IDestablo,"Establo", USER(), "INSERT");
    -> END;
    -> //
Query OK, 0 rows affected (0.035 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Actualizar_Establo
    -> AFTER UPDATE ON proyecto.Establo
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaEstablo(fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.IDestablo, "Establo", USER(), "UPDATE");
    -> END;
    -> //
Query OK, 0 rows affected (0.023 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Borrar_Establo
    -> AFTER DELETE ON proyecto.Establo
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaEstablo (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), OLD.IDestablo, "Establo", USER(), "DELETE");
    -> END;
    -> //
Query OK, 0 rows affected (0.031 sec)
```

Triggers auditoria Finca: Dentro de esta auditoria solo se guardará los datos insertados/modificados/eliminados de la tabla Finca.

```
MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Insertar_Finca
    -> AFTER INSERT ON proyecto.Finca
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaFinca (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.IDfinca, "Finca", USER(), "INSERT");
    -> END;
    -> //
Query OK, 0 rows affected (0.034 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Actualizar_Finca
    -> AFTER UPDATE ON proyecto.Finca
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaFinca(fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.IDfinca, "Finca", USER(), "UPDATE");
    -> END;
    -> //
Query OK, 0 rows affected (0.025 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Borrar_Finca
    -> AFTER DELETE ON proyecto.Finca
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaFinca (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), OLD.IDfinca, "Finca", USER(), "DELETE");
    -> END;
    -> //
Query OK, 0 rows affected (0.024 sec)
```

Triggers auditoriaEstablo: Dentro de esta auditoria solo se guardará los datos insertados/modificados/eliminados de la tabla Establo.

```
MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Insertar_Establo
    -> AFTER INSERT ON proyecto.Establo
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaEstablo (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.IDestablo,"Establo", USER(), "INSERT");
    -> END;
    -> //
Query OK, 0 rows affected (0.035 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Actualizar_Establo
    -> AFTER UPDATE ON proyecto.Establo
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaEstablo(fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.IDestablo, "Establo", USER(), "UPDATE");
    -> END;
    -> //
Query OK, 0 rows affected (0.023 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Borrar_Establo
    -> AFTER DELETE ON proyecto.Establo
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaEstablo (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), OLD.IDestablo, "Establo", USER(), "DELETE");
    -> END;
    -> //
Query OK, 0 rows affected (0.031 sec)
```

Triggers auditoriaCamionCarga: Dentro de esta auditoria solo se guardará los datos insertados/modificados/eliminados de la tabla CamionCarga.

```
MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Insertar_CamionCarga
    -> AFTER INSERT ON proyecto.CamionCarga
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaCamionCarga (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.matricula,"CamionCarga", USER(), "INSERT");
    -> END;
    -> //
Query OK, 0 rows affected (0.038 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Actualizar_CamionCarga
    -> AFTER UPDATE ON proyecto.CamionCarga
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaCamionCarga(fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.matricula, "CamionCarga", USER(), "UPDATE");
    -> END;
    -> //
Query OK, 0 rows affected (0.022 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Borrar_CamionCarga
    -> AFTER DELETE ON proyecto.CamionCarga
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaCamionCarga (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), OLD.matricula, "CamionCarga", USER(), "DELETE");
    -> END;
    -> DELIMITER//
Query OK, 0 rows affected (0.040 sec)
```

Triggers auditoriaAlimento: Dentro de esta auditoria solo se guardará los datos insertados/modificados/eliminados de la tabla Alimento/Carne/CartonHuevo/Cultivo.

```
MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Insertar_Alimento
    -> AFTER INSERT ON proyecto.Alimento
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaAlimento (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.IDalimento,"Alimento", USER(), "INSERT");
    -> END;
    -> //
Query OK, 0 rows affected (0.037 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Actualizar_Alimento
    -> AFTER UPDATE ON proyecto.Alimento
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaAlimento(fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.IDalimento, "Alimento", USER(), "UPDATE");
    -> END;
    -> //
Query OK, 0 rows affected (0.035 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Borrar_Alimento
    -> AFTER DELETE ON proyecto.Alimento
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaAlimento (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), OLD.IDalimento, "Alimento", USER(), "DELETE");
    -> END;
    -> //
Query OK, 0 rows affected (0.041 sec)
```

Carne:

```
MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Insertar_Carne
    -> AFTER INSERT ON proyecto.Carne
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaAlimento (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.IDalimento,"Carne", USER(), "INSERT");
    -> END;
    -> //
Query OK, 0 rows affected (0.018 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Actualizar_Carne
    -> AFTER UPDATE ON proyecto.Carne
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaAlimento(fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.IDalimento, "Carne", USER(), "UPDATE");
    -> END;
    -> //
Query OK, 0 rows affected (0.024 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Borrar_Carne
    -> AFTER DELETE ON proyecto.Carne
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaAlimento (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), OLD.IDalimento, "Carne", USER(), "DELETE");
    -> END;
    -> //
Query OK, 0 rows affected (0.041 sec)
```

CartonHuevo:

```
MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Insertar_CartonHuevo
    -> AFTER INSERT ON proyecto.CartonHuevo
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaAlimento (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.IDalimento,"CartonHuevo", USER(), "INSERT");
    -> END;
    -> //
Query OK, 0 rows affected (0.050 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Actualizar_CartonHuevo
    -> AFTER UPDATE ON proyecto.CartonHuevo
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaAlimento(fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.IDalimento, "CartonHuevo", USER(), "UPDATE");
    -> END;
    -> //
Query OK, 0 rows affected (0.022 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Borrar_CartonHuevo
    -> AFTER DELETE ON proyecto.CartonHuevo
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaAlimento (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), OLD.IDalimento, "CartonHuevo", USER(), "DELETE");
    -> END;
    -> //
Query OK, 0 rows affected (0.024 sec)
```

Cultivo:

```
MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Insertar_Cultivo
    -> AFTER INSERT ON proyecto.Cultivo
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaAlimento (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.IDalimento,"Cultivo", USER(), "INSERT");
    -> END;
    -> //
Query OK, 0 rows affected (0.027 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Actualizar_Cultivo
    -> AFTER UPDATE ON proyecto.Cultivo
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaAlimento(fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), NEW.IDalimento, "Cultivo", USER(), "UPDATE");
    -> END;
    -> //
Query OK, 0 rows affected (0.022 sec)

MariaDB [Proyecto]> DELIMITER //
MariaDB [Proyecto]> CREATE TRIGGER Borrar_Cultivo
    -> AFTER DELETE ON proyecto.Cultivo
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO AuditoriaProyecto.auditoriaAlimento (fecha, IDobjeto, Tabla, Usuario, Operacion)
    -> VALUES (NOW(), OLD.IDalimento, "Cultivo", USER(), "DELETE");
    -> END;
    -> //
Query OK, 0 rows affected (0.040 sec)
```

3. Documentación Pruebas al modelo

Dentro de cada tabla insertaremos, modificaremos y eliminaremos un dato y luego revisaremos su respectiva tabla de auditoría.

Prueba Persona:

```
MariaDB [Proyecto]> INSERT INTO persona (cedula, nombre, fechaNac, edad, direccion, genero)
    -> VALUES (12345678, 'Juan Perez', '1985-06-15', 38, 'Calle Falsa 123', 'M');
Query OK, 1 row affected (0.021 sec)

MariaDB [Proyecto]>
MariaDB [Proyecto]> UPDATE persona
    -> SET direccion = 'Avenida Siempre Viva 742'
    -> WHERE cedula = 12345678;
Query OK, 1 row affected (0.002 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [Proyecto]>
MariaDB [Proyecto]> DELETE FROM persona
    -> WHERE cedula = 12345678;
Query OK, 1 row affected (0.020 sec)
```

```
MariaDB [AuditoriaProyecto]> select * from AuditoriaPersona;
+-------------+---------------------+----------+---------+---------------------+-----------+
| IDauditoria | fecha               | IDobjeto | Tabla   | Usuario             | Operacion |
+-------------+---------------------+----------+---------+---------------------+-----------+
|           1 | 2024-05-23 15:55:16 | 12345678 | Persona | DanielJay@localhost | INSERT    |
|           2 | 2024-05-23 15:55:16 | 12345678 | Persona | DanielJay@localhost | UPDATE    |
|           3 | 2024-05-23 15:55:16 | 12345678 | Persona | DanielJay@localhost | DELETE    |
```

Prueba Cliente:

```
MariaDB [Proyecto]> INSERT INTO cliente (cedula, clasificacion)
    -> VALUES (9101112, 'Minorista');
Query OK, 1 row affected (0.020 sec)

MariaDB [Proyecto]> UPDATE cliente
    -> SET clasificacion = 'Mayorista'
    -> WHERE cedula = 9101112;
Query OK, 1 row affected (0.021 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [Proyecto]> DELETE FROM cliente
    -> WHERE cedula = 9101112;
Query OK, 1 row affected (0.021 sec)
```

Auditoria:

```
MariaDB [auditoriaProyecto]> select * from auditoriaCliente
    -> ;
+-------------+---------------------+----------+---------+---------------------+-----------+
| IDauditoria | fecha               | IDobjeto | Tabla   | Usuario             | Operacion |
+-------------+---------------------+----------+---------+---------------------+-----------+
|           1 | 2024-05-23 16:02:18 |  9101112 | Cliente | DanielJay@localhost | INSERT    |
|           2 | 2024-05-23 16:02:32 |  9101112 | Cliente | DanielJay@localhost | UPDATE    |
|           3 | 2024-05-23 16:02:38 |  9101112 | Cliente | DanielJay@localhost | DELETE    |
+-------------+---------------------+----------+---------+---------------------+-----------+
3 rows in set (0.001 sec)
```

Prueba Empleado:

```
MariaDB [proyecto]> INSERT INTO empleado (cedula, salario, añosExperiencia)
    -> VALUES
    -> (12345679, 35000, 10),
    -> (12345680, 45000, 20),
    -> (12345681, 40000, 15);
Query OK, 3 rows affected (0.023 sec)
Records: 3  Duplicates: 0  Warnings: 0

MariaDB [proyecto]> INSERT INTO vendedor (cedula, HoraEntrada, HoraSalida, tienda)
    -> VALUES (12345679, '09:00:00', '17:00:00', 10097);
Query OK, 1 row affected (0.022 sec)

MariaDB [proyecto]> INSERT INTO transportador (cedula, licencia)
    -> VALUES (12345680, 56789);
Query OK, 1 row affected (0.023 sec)

MariaDB [proyecto]> INSERT INTO trabajadorFinca (cedula, Labor, mayordomo, finca)
    -> VALUES (12345681, 'Cosecha de frutas', NULL, 41);
Query OK, 1 row affected (0.022 sec)

MariaDB [proyecto]>
MariaDB [proyecto]> DELETE FROM trabajadorFinca
    -> WHERE cedula = 12345681;
Query OK, 1 row affected (0.020 sec)

MariaDB [proyecto]> UPDATE transportador
    -> SET licencia = 98765
    -> WHERE cedula = 12345680;
Query OK, 1 row affected (0.021 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

En empleado ingresamos 3 empleados, lo metemos cada uno a un subtipo (vendedor, transportador y trabajadorfinca) y luego borramos al trabajador finca y actualizamos los datos del transportador

Auditoria:

```
MariaDB [auditoriaproyecto]> select * from auditoriaEmpleado;
+-------------+---------------------+----------+-----------------+---------------------+-----------+
| IDauditoria | fecha               | IDobjeto | Tabla           | Usuario             | Operacion |
+-------------+---------------------+----------+-----------------+---------------------+-----------+
|           1 | 2024-05-23 16:05:27 | 12345679 | Empleado        | DanielJay@localhost | INSERT    |
|           2 | 2024-05-23 16:05:27 | 12345680 | Empleado        | DanielJay@localhost | INSERT    |
|           3 | 2024-05-23 16:05:27 | 12345681 | Empleado        | DanielJay@localhost | INSERT    |
|           4 | 2024-05-23 16:06:38 | 12345679 | Vendedor        | DanielJay@localhost | INSERT    |
|           5 | 2024-05-23 16:07:39 | 12345680 | Transportador   | DanielJay@localhost | INSERT    |
|           6 | 2024-05-23 16:08:21 | 12345681 | TrabajadorFinca | DanielJay@localhost | INSERT    |
|           7 | 2024-05-23 16:08:44 | 12345681 | TrabajadorFinca | DanielJay@localhost | DELETE    |
|           8 | 2024-05-23 16:08:47 | 12345680 | Transportador   | DanielJay@localhost | UPDATE    |
+-------------+---------------------+----------+-----------------+---------------------+-----------+
8 rows in set (0.001 sec)
```

Prueba Compra: Para la prueba de compra vamos a borrar y actualizar una de las compras ya existentes.

```
MariaDB [proyecto]> DELETE FROM compra
    -> WHERE IDfactura = 25;
Query OK, 1 row affected (0.023 sec)

MariaDB [proyecto]>
MariaDB [proyecto]> -- Modificar una compra en la tabla compra
MariaDB [proyecto]> UPDATE compra
    -> SET productos = 1, valor = 7000
    -> WHERE IDfactura = 24;
Query OK, 1 row affected (0.002 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Auditoria:

```
MariaDB [auditoriaProyecto]> select * from auditoriacompra;
+-------------+---------------------+----------+---------+---------------------+-----------+
| IDauditoria | fecha               | IDobjeto | Tabla   | Usuario             | Operacion |
+-------------+---------------------+----------+---------+---------------------+-----------+
|           1 | 2024-05-23 16:13:12 |       25 | Compra | DanielJay@localhost | DELETE    |
|           2 | 2024-05-23 16:13:12 |       24 | Compra | DanielJay@localhost | UPDATE    |
+-------------+---------------------+----------+---------+---------------------+-----------+
2 rows in set (0.001 sec)
```

Prueba Punto de venta:

```
MariaDB [proyecto]> INSERT INTO puntoVenta (IDtienda, direccion)
    -> VALUES (100102, 'Calle Principal 789');
Query OK, 1 row affected (0.006 sec)

MariaDB [proyecto]> UPDATE puntoVenta
    -> SET direccion = 'Avenida Secundaria 456'
    -> WHERE IDtienda = 100102;
Query OK, 1 row affected (0.002 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [proyecto]> DELETE FROM puntoVenta
    -> WHERE IDtienda = 100102;
Query OK, 1 row affected (0.022 sec)
```

Auditoria:

```
MariaDB [auditoriaProyecto]> select * from auditoriapuntoventa;
+-------------+---------------------+----------+------------+---------------------+-----------+
| IDauditoria | fecha               | IDobjeto | Tabla      | Usuario             | Operacion |
+-------------+---------------------+----------+------------+---------------------+-----------+
|           1 | 2024-05-23 16:15:39 |   100102 | PuntoVenta | DanielJay@localhost | INSERT    |
|           2 | 2024-05-23 16:15:47 |   100102 | PuntoVenta | DanielJay@localhost | UPDATE    |
|           3 | 2024-05-23 16:15:55 |   100102 | PuntoVenta | DanielJay@localhost | DELETE    |
+-------------+---------------------+----------+------------+---------------------+-----------+
3 rows in set (0.001 sec)
```

Prueba Finca:

```
MariaDB [proyecto]> INSERT INTO finca (IDfinca, nombre, direccion)
    -> VALUES (1001, 'Finca La Esperanza', 'Calle Principal 123');
Query OK, 1 row affected (0.022 sec)

MariaDB [proyecto]>
MariaDB [proyecto]> UPDATE finca
    -> SET direccion = 'Avenida Secundaria 456'
    -> WHERE IDfinca = 1001;
Query OK, 1 row affected (0.002 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [proyecto]>
MariaDB [proyecto]> DELETE FROM finca
    -> WHERE IDfinca = 1001;
Query OK, 1 row affected (0.021 sec)
```

Auditoria:

```
MariaDB [auditoriaProyecto]> select * from auditoriafinca;
+-------------+---------------------+----------+-------+---------------------+-----------+
| IDauditoria | fecha               | IDobjeto | Tabla | Usuario             | Operacion |
+-------------+---------------------+----------+-------+---------------------+-----------+
|           1 | 2024-05-23 16:17:58 |     1001 | Finca | DanielJay@localhost | INSERT    |
|           2 | 2024-05-23 16:17:58 |     1001 | Finca | DanielJay@localhost | UPDATE    |
|           3 | 2024-05-23 16:17:58 |     1001 | Finca | DanielJay@localhost | DELETE    |
|           4 | 2024-05-23 16:19:00 |     1001 | Finca | DanielJay@localhost | INSERT    |
+-------------+---------------------+----------+-------+---------------------+-----------+
4 rows in set (0.001 sec)
```

Prueba Establo:

```
MariaDB [proyecto]> INSERT INTO establo (IDestablo, tipo, tamaño, IDfinca)
    -> VALUES (2001, 'Establo A', 50, 1001);
Query OK, 1 row affected (0.023 sec)

MariaDB [proyecto]>
MariaDB [proyecto]> UPDATE establo
    -> SET tamaño = 60
    -> WHERE IDestablo = 2001;
Query OK, 1 row affected (0.002 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [proyecto]>
MariaDB [proyecto]> DELETE FROM establo
    -> WHERE IDestablo = 2001;
Query OK, 1 row affected (0.022 sec)
```

Auditoria:

```
MariaDB [auditoriaProyecto]> select * from auditoriaestablo;
+-------------+---------------------+----------+---------+---------------------+-----------+
| IDauditoria | fecha               | IDobjeto | Tabla   | Usuario             | Operacion |
+-------------+---------------------+----------+---------+---------------------+-----------+
|           1 | 2024-05-23 16:19:28 |     2001 | Establo | DanielJay@localhost | INSERT    |
|           2 | 2024-05-23 16:19:28 |     2001 | Establo | DanielJay@localhost | UPDATE    |
|           3 | 2024-05-23 16:19:28 |     2001 | Establo | DanielJay@localhost | DELETE    |
|           4 | 2024-05-23 16:20:14 |     2001 | Establo | DanielJay@localhost | INSERT    |
+-------------+---------------------+----------+---------+---------------------+-----------+
4 rows in set (0.001 sec)
```

Prueba Animal:

```
MariaDB [proyecto]> INSERT INTO animal (IDanimal, peso, raza, sexo, estadoSalud, edad, etapaDesarrollo, establo, tipo)
    -> VALUES (3001, 250, 'Holstein', 'M', 'Saludable', 3, 'Adulto', 2001, 'Vaca');
Query OK, 1 row affected (0.023 sec)

MariaDB [proyecto]>
MariaDB [proyecto]> UPDATE animal
    -> SET peso = 280
    -> WHERE IDanimal = 3001;
Query OK, 1 row affected (0.002 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [proyecto]>
MariaDB [proyecto]> DELETE FROM animal
    -> WHERE IDanimal = 3001;
Query OK, 1 row affected (0.021 sec)
```

Auditoria:

```
MariaDB [auditoriaProyecto]> select * from auditoriaanimal;
+-------------+---------------------+----------+--------+---------------------+-----------+
| IDauditoria | fecha               | IDobjeto | Tabla  | Usuario             | Operacion |
+-------------+---------------------+----------+--------+---------------------+-----------+
|           1 | 2024-05-23 16:20:50 |     3001 | Animal | DanielJay@localhost | INSERT    |
|           2 | 2024-05-23 16:20:50 |     3001 | Animal | DanielJay@localhost | UPDATE    |
|           3 | 2024-05-23 16:20:51 |     3001 | Animal | DanielJay@localhost | DELETE    |
+-------------+---------------------+----------+--------+---------------------+-----------+
3 rows in set (0.001 sec)
```

Prueba LoteProduccion:

```
MariaDB [proyecto]>
MariaDB [proyecto]> DELETE FROM loteProduccion
    -> WHERE IDlote = 4001;
Query OK, 1 row affected (0.007 sec)
```

Auditoria:

```
MariaDB [auditoriaProyecto]> select * from auditorialoteproduccion;
+-------------+---------------------+----------+----------------+---------------------+-----------+
| IDauditoria | fecha               | IDobjeto | Tabla          | Usuario             | Operacion |
+-------------+---------------------+----------+----------------+---------------------+-----------+
|           1 | 2024-05-23 16:23:39 |     4001 | LoteProduccion | DanielJay@localhost | DELETE    |
+-------------+---------------------+----------+----------------+---------------------+-----------+
1 row in set (0.001 sec)
```

Prueba CamionCarga:

```
MariaDB [proyecto]> INSERT INTO CamionCarga (matricula, marca, modelo)
    -> VALUES (1, 'Volvo', 'VNL64T780');
Query OK, 1 row affected (0.022 sec)

MariaDB [proyecto]>
MariaDB [proyecto]> UPDATE CamionCarga
    -> SET marca = 'Mercedes-Benz'
    -> WHERE matricula = 1;
Query OK, 1 row affected (0.002 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [proyecto]>
MariaDB [proyecto]> DELETE FROM CamionCarga
    -> WHERE matricula = 1;
Query OK, 1 row affected (0.002 sec)
```

Auditoria:

```
MariaDB [auditoriaProyecto]> select * from auditoriacamioncarga;
+-------------+---------------------+----------+-------------+---------------------+-----------+
| IDauditoria | fecha               | IDobjeto | Tabla       | Usuario             | Operacion |
+-------------+---------------------+----------+-------------+---------------------+-----------+
|           1 | 2024-05-23 16:24:35 |     1001 | CamionCarga | DanielJay@localhost | UPDATE    |
|           2 | 2024-05-23 16:24:58 |        1 | CamionCarga | DanielJay@localhost | INSERT    |
|           3 | 2024-05-23 16:24:58 |        1 | CamionCarga | DanielJay@localhost | UPDATE    |
|           4 | 2024-05-23 16:24:58 |        1 | CamionCarga | DanielJay@localhost | DELETE    |
+-------------+---------------------+----------+-------------+---------------------+-----------+
4 rows in set (0.001 sec)
```

Prueba Alimento: De igual manera de como empleado, para la prueba crearemos alimentos y luego los repartiremos entre sus subtipos para realizar las pruebas.

```
MariaDB [proyecto]> INSERT INTO alimento (IDalimento, fechaProduccion, fechaVencimiento, valorXlibra, certificacion)
    -> VALUES
    -> (1, '2024-05-01', '2024-06-30', 300, 'orgánico'),
    -> (2, '2024-05-05', '2024-07-05', 300, 'orgánico'),
    -> (3, '2024-05-10', '2024-08-10', 300, 'orgánico');
Query OK, 3 rows affected (0.023 sec)
Records: 3  Duplicates: 0  Warnings: 0

MariaDB [proyecto]> INSERT INTO cultivo (IDalimento, nombre, tipo, hectareaCultivo)
    -> VALUES (1, 'Manzanas', 'Fruta', 301);
Query OK, 1 row affected (0.022 sec)

MariaDB [proyecto]> INSERT INTO carne (IDalimento, peso, corte, raza, tipo)
    -> VALUES (2, 500, 'Filete', 'Angus', 'Vacuno');
Query OK, 1 row affected (0.022 sec)

MariaDB [proyecto]> INSERT INTO CartonHuevo (IDalimento, clasificacion, unidades, tipoHuevo)
    -> VALUES (3, 'A', 12, 'Criollo');
Query OK, 1 row affected (0.023 sec)

MariaDB [proyecto]>
MariaDB [proyecto]> UPDATE carne
    -> SET peso = 550
    -> WHERE IDalimento = 2;
Query OK, 1 row affected (0.021 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [proyecto]> DELETE FROM CartonHuevo
    -> WHERE IDalimento = 3;
Query OK, 1 row affected (0.002 sec)
```

Auditoria:

```
MariaDB [auditoriaProyecto]> select * from auditoriaalimento;
+-------------+---------------------+----------+-------------+---------------------+-----------+
| IDauditoria | fecha               | IDobjeto | Tabla       | Usuario             | Operacion |
+-------------+---------------------+----------+-------------+---------------------+-----------+
|           1 | 2024-05-23 16:28:29 |        1 | Alimento    | DanielJay@localhost | INSERT    |
|           2 | 2024-05-23 16:28:29 |        2 | Alimento    | DanielJay@localhost | INSERT    |
|           3 | 2024-05-23 16:28:29 |        3 | Alimento    | DanielJay@localhost | INSERT    |
|           4 | 2024-05-23 16:29:00 |        1 | Cultivo     | DanielJay@localhost | INSERT    |
|           5 | 2024-05-23 16:29:27 |        2 | Carne       | DanielJay@localhost | INSERT    |
|           6 | 2024-05-23 16:30:04 |        3 | CartonHuevo | DanielJay@localhost | INSERT    |
|           7 | 2024-05-23 16:30:24 |        2 | Carne       | DanielJay@localhost | UPDATE    |
|           8 | 2024-05-23 16:30:29 |        3 | CartonHuevo | DanielJay@localhost | DELETE    |
+-------------+---------------------+----------+-------------+---------------------+-----------+
8 rows in set (0.001 sec)
```

Conclusiones

La implementación de un modelo de base de datos para la administración de la mercancía de una finca agropecuaria trae múltiples beneficios y nos ayuda abordar ciertas ventajas a la hora de una buena administración, permitiéndonos recopilar una gran cantidad de información y almacenarlos electrónicamente, la implementación de métodos específicos como diagramas de entidad y esquemas relacionales ilustra cómo el modelo puede ser extendido para incluir funcionalidades específicas del dominio, en este modelo se abordó los aspectos más importantes de la finca, entre las entidades y las relaciones se refleja la complejidad y la interconexión de los elementos presentes en la finca agropecuaria ya qu**e** nos permite hacer búsquedas o encontrar cualquier información de una manera rápida, ya que optimiza la distribución de información y da una mayor precisión y consistencia a lo que buscamos. Al disponer de una base de datos bien estructurada se puede analizar el estado de la finca para identificar puntos importantes o tomar decisiones informadas a futuro que puedan mejorar su viabilidad, la implementación de MYSQL un sistema de gestión de base de datos nos da la posibilidad de que podamos modificar los datos según las necesidades de la finca y realizar operaciones complejas eliminando redundancias, esto minimiza los errores.

La base de datos es con el fin del uso de implementar la gestión de bases de datos, mejorando las competencias técnicas del equipo ya que esto permitiría agilizar los procesos operativos, la calidad del uso de datos y mejora la eficiencia y reducción de riesgos.