Los Ambientales

Campers:

Robinson Mosquera Cubides

Eduar Damián Chanaga González

Campuslands

Grupo:

P1

Trainer:

Pedro Felipe Gómez Bonilla

10 de julio 2024

Contenido

Introducción

El Ministerio del Medio Ambiente ha iniciado un proyecto innovador para mejorar la gestión de los datos e información sobre los parques naturales que administra cada departamento. El objetivo es desarrollar un software sólido y funcional que organice y analice toda la información relacionada con los parques naturales, desde sus características físicas hasta los datos de las especies y el personal que los gestiona.

En esta primera fase, se han definido varias características clave para el sistema. Cada departamento puede tener múltiples parques, y cada parque puede abarcar áreas de varios departamentos. Las entidades responsables de los parques, las especies que habitan en diferentes áreas, y el personal que trabaja en los parques también son aspectos fundamentales del sistema. Además, se tendrá en cuenta la estructura y funcionamiento del personal, dividido en diferentes roles y funciones, así como la información detallada de los visitantes.

Para desarrollar este proyecto, utilizaremos un repositorio privado en GitHub donde documentaremos y seguiremos los pasos necesarios para la creación del sistema de información. Estos pasos incluyen la extrapolación del caso de estudio a un modelo conceptual, la construcción y normalización del modelo lógico y físico, y finalmente, la implementación del modelo físico en SQL.

Este proyecto no solo es una oportunidad para aplicar nuestros conocimientos en bases de datos, sino también un desafío para contribuir a la conservación y gestión eficiente de los parques naturales. Esperamos que el resultado de nuestro esfuerzo sea una herramienta útil y efectiva para el Ministerio del Medio Ambiente y para la preservación de nuestro patrimonio natural.

Caso de estudio

La gestión de los parques naturales ha enfrentado desafíos significativos debido a la falta de un sistema eficiente para organizar y analizar la información. Con cinco parques naturales, cada uno con características únicas y una gran biodiversidad, la entidad responsable ha tenido dificultades para registrar y monitorear las áreas protegidas, las especies que las habitan y el personal encargado de la conservación y vigilancia. Además, la gestión de la información de los visitantes y el uso de los alojamientos disponibles también ha sido problemática. El objetivo de este proyecto es desarrollar un sistema de información que permita gestionar de manera más eficiente estos datos, facilitando el registro y actualización de información sobre los parques, las entidades responsables, las especies, el personal y los visitantes. Con esta solución, se espera mejorar la eficiencia en la gestión de los parques naturales, optimizar la toma de decisiones para la conservación y el uso sostenible de estos espacios, y promover la preservación de la biodiversidad y el ecoturismo en la región.

Instalación

1. Instalación de MySQL Workbench

Para instalar MySQL Workbench, sigue estos pasos:

1. **Descarga MySQL Workbench**:
    - Ve a la página oficial de MySQL en https://dev.mysql.com/downloads/workbench/.
    - Selecciona tu sistema operativo (Windows, macOS, Linux) y descarga la versión correspondiente de MySQL Workbench.
2. **Instala MySQL Workbench**:

- Ejecuta el archivo descargado y sigue las instrucciones del asistente de instalación.
- Acepta los términos y condiciones, y elige las configuraciones predeterminadas a menos que necesites realizar ajustes específicos.

3. **Inicia MySQL Workbench**:
   - Una vez instalada, abre MySQL Workbench desde el menú de inicio o desde el ícono en tu escritorio.
   - Configura una conexión a tu servidor MySQL ingresando los detalles de tu servidor (host, puerto, nombre de usuario, contraseña).

2. Creación y Configuración de la Base de Datos

Una vez que MySQL Workbench esté instalado y configurado, puedes crear la base de datos siguiendo estos pasos:

1. **Conéctate al Servidor MySQL**:
   - En MySQL Workbench, haz clic en el ícono de conexión que configuraste previamente para conectarte al servidor MySQL.
2. **Abre y Ejecuta Archivos SQL**:
   - Para facilitar la creación de la base de datos y las tablas, escribe todos los comandos SQL en archivos separados y ejecútalos de manera ordenada en MySQL Workbench. Los archivos y el orden recomendado son los siguientes:
     1. **creacion_base_de_datos_y_tablas.sql**:
        - Ve a File y selecciona Open SQL Script... para abrir el archivo creacion_base_de_datos_y_tablas.sql que contiene los comandos para crear la base de datos y las tablas.
        - Haz clic en el ícono de rayo amarillo (Execute) para ejecutar todos los comandos en este archivo.

2. **eventos_triggers.sql**:
    - Finalmente, abre el archivo eventos_triggers.sql para configurar los eventos y triggers en la base de datos.
    - Ejecuta estos comandos para completar la configuración.
3. **usuarios_permisos.sql**:
    - A continuación, abre el archivo usuarios_permisos.sql para configurar los usuarios y sus permisos en la base de datos.
    - Ejecuta los comandos utilizando el ícono de rayo amarillo.
4. **funciones_procedimientos.sql**:
    - Luego, abre el archivo funciones_procedimientos.sql para crear las funciones y procedimientos almacenados necesarios.
    - Ejecuta los comandos correspondientes.
5. **inserts.sql**:
    - Después de crear la base de datos y las tablas, abre el archivo inserts.sql que contiene los comandos para insertar los datos iniciales.
    - Nuevamente, haz clic en el ícono de rayo amarillo para ejecutar los comandos de inserción.

Planificación

Equipo de Trabajo:

- Robinson Mosquera (Documentación)
- Eduar Chanaga (Implementación)

Objetivo del Proyecto

Desarrollar una base de datos para gestionar la información de los parques naturales administrados por el Ministerio del Medio Ambiente. La base de datos debe incluir tablas para visitantes, proyectos de investigación, entidades responsables, especies, vehículos, alojamientos, parques naturales, departamentos, áreas, personal, inventarios de especies, y relaciones entre estas entidades.

Estructura de la Base de Datos

1. Base de Datos

- Crear la base de datos ParquesNaturalesColombia y seleccionarla para su uso.

2. Tablas

- **Tablas sin Foreign Key:**
  - **Visitantes:** Para almacenar información básica sobre los visitantes de los parques.
  - **Proyectos de investigación:** Para gestionar los proyectos de investigación que se llevan a cabo en los parques naturales.
  - **Entidades responsables:** Para registrar las entidades encargadas de la gestión de los parques.
  - **Especies:** Para clasificar las especies presentes en los parques naturales.

        - **Vehículos:** Para registrar los vehículos utilizados en la vigilancia de los parques.
        - **Alojamientos:** Para gestionar los alojamientos disponibles en los parques para los visitantes.
        - **Parques naturales:** Para almacenar información general sobre cada parque natural.
    - **Tablas con Foreign Key:**
        - **Departamentos:** Para registrar los departamentos que gestionan los parques y sus entidades responsables.
        - **Áreas:** Para dividir los parques en áreas específicas, cada una con su extensión.
    - **Tablas con más de una Foreign Key:**
        - **Inventario de especies:** Para llevar un registro del número de individuos de cada especie en cada área del parque.
        - **Personal de gestión:** Para detallar el personal encargado de la gestión de los visitantes en las entradas de los parques.
        - **Personal de investigación:** Para registrar los investigadores y sus titulaciones.
        - **Personal de conservación:** Para gestionar el personal encargado de la conservación de las áreas del parque y sus especialidades.
        - **Personal de vigilancia:** Para registrar el personal encargado de la vigilancia del parque, sus áreas asignadas y los vehículos utilizados.
    - **Tablas de Relaciones:**
        - **Relación entre departamento y parque:** Para gestionar qué parques son administrados por cada departamento.

- **Relación entre investigador y proyecto:** Para registrar los proyectos de investigación y los investigadores involucrados.
- **Relación entre visitante y alojamiento:** Para llevar un registro de los alojamientos ocupados por los visitantes.

Justificación de las Tablas

- **Visitantes:** Necesaria para identificar a las personas que visitan los parques y gestionar su estancia.
- **Proyectos de investigación:** Fundamental para organizar los proyectos que buscan preservar y estudiar la biodiversidad.
- **Entidades responsables:** Importante para saber quiénes son los encargados de la gestión de los parques.
- **Especies:** Permite catalogar la biodiversidad presente en los parques, facilitando su estudio y conservación.
- **Vehículos:** Esencial para el control y asignación de los vehículos utilizados por el personal de vigilancia.
- **Alojamientos:** Facilita la gestión de los alojamientos disponibles para los visitantes.
- **Parques naturales:** Base para todas las operaciones y consultas relacionadas con los parques.
- **Departamentos:** Clave para identificar la administración de los parques a nivel regional.
- **Áreas:** Necesaria para una organización detallada dentro de cada parque.
- **Inventario de especies:** Vital para el seguimiento y conservación de la fauna y flora en los parques.
- **Personal de gestión:** Ayuda a organizar el personal encargado de la entrada y gestión de visitantes.

- **Personal de investigación:** Facilita la administración del personal que realiza investigaciones en los parques.
- **Personal de conservación:** Importante para mantener y preservar las áreas de los parques.
- **Personal de vigilancia:** Crucial para la seguridad y vigilancia dentro de los parques.
- **Relación entre departamento y parque:** Esencial para entender la administración de los parques.
- **Relación entre investigador y proyecto:** Permite organizar los proyectos de investigación y los investigadores asociados.
- **Relación entre visitante y alojamiento:** Necesaria para gestionar las estancias de los visitantes en los alojamientos disponibles.

Ejecución

Construcción del modelo conceptual

El modelo conceptual de la base de datos ParquesNaturalesColombia está diseñado para gestionar eficientemente toda la información relacionada con los parques naturales administrados por el Ministerio del Medio Ambiente. A continuación, describo el propósito y las relaciones de cada tabla en el sistema:

1. **Visitantes**
    - **Descripción:** Almacena información básica sobre los visitantes de los parques naturales.
    - **Atributos:** ID, Nombre, Dirección, Profesión.
2. **Proyectos de Investigación**
    - **Descripción:** Gestiona los proyectos de investigación que se llevan a cabo en los parques.

- **Atributos:** ID, Nombre, Presupuesto, Fecha de Inicio, Fecha de Fin.

3. **Entidad Responsable**
    - **Descripción:** Registra las entidades encargadas de la gestión de los parques naturales.
    - **Atributos:** ID, Nombre, Dirección, Teléfono.
4. **Especies**
    - **Descripción:** Clasifica las especies presentes en los parques naturales.
    - **Atributos:** ID, Nombre Científico, Nombre Vulgar, Categoría (Vegetal, Animal, Mineral).
5. **Vehículos**
    - **Descripción:** Registra los vehículos utilizados en la vigilancia de los parques.
    - **Atributos:** ID, Tipo, Marca.
6. **Alojamientos**
    - **Descripción:** Gestiona los alojamientos disponibles en los parques para los visitantes.
    - **Atributos:** ID, Nombre, Capacidad, Categoría (Hotel, Cabaña).
7. **Parques Naturales**
    - **Descripción:** Almacena información general sobre cada parque natural.
    - **Atributos:** ID, Nombre, Fecha de Declaración.
8. **Departamentos**
    - **Descripción:** Registra los departamentos que gestionan los parques y sus entidades responsables.
    - **Atributos:** ID, Nombre, Entidad Responsable (FK).

9. **Áreas**
    - **Descripción:** Divide los parques en áreas específicas, cada una con su extensión.
    - **Atributos:** ID, Nombre, Extensión, Parque (FK).
10. **Personal**
    - **Descripción:** Almacena información sobre el personal que trabaja en los parques naturales.
    - **Atributos:** ID, Documento, Nombre, Dirección, Teléfono, Sueldo, Tipo (Personal de Gestión, Personal de Vigilancia, Personal de Conservación, Personal Investigador), Parque (FK).
11. **Inventario de Especies**
    - **Descripción:** Lleva un registro del número de individuos de cada especie en cada área del parque.
    - **Atributos:** ID, Número de Individuos, Área (FK), Especie (FK).
12. **Personal de Gestión**
    - **Descripción:** Detalla el personal encargado de la gestión de los visitantes en las entradas de los parques.
    - **Atributos:** ID (FK), Nombre, Puerta.
13. **Personal de Investigación**
    - **Descripción:** Registra los investigadores y sus titulaciones.
    - **Atributos:** ID (FK), Nombre, Titulación.
14. **Personal de Conservación**
    - **Descripción:** Gestiona el personal encargado de la conservación de las áreas del parque y sus especialidades.

- **Atributos:** ID (FK), Nombre, Área (FK), Especialidad.

15. **Personal de Vigilancia**

    - **Descripción:** Registra el personal encargado de la vigilancia del parque, sus áreas asignadas y los vehículos utilizados.
    - **Atributos:** ID (FK), Nombre, Área (FK), Vehículo (FK).

16. **Relación entre Departamento y Parque**

    - **Descripción:** Gestiona qué parques son administrados por cada departamento.
    - **Atributos:** Departamento (FK), Parque (FK).

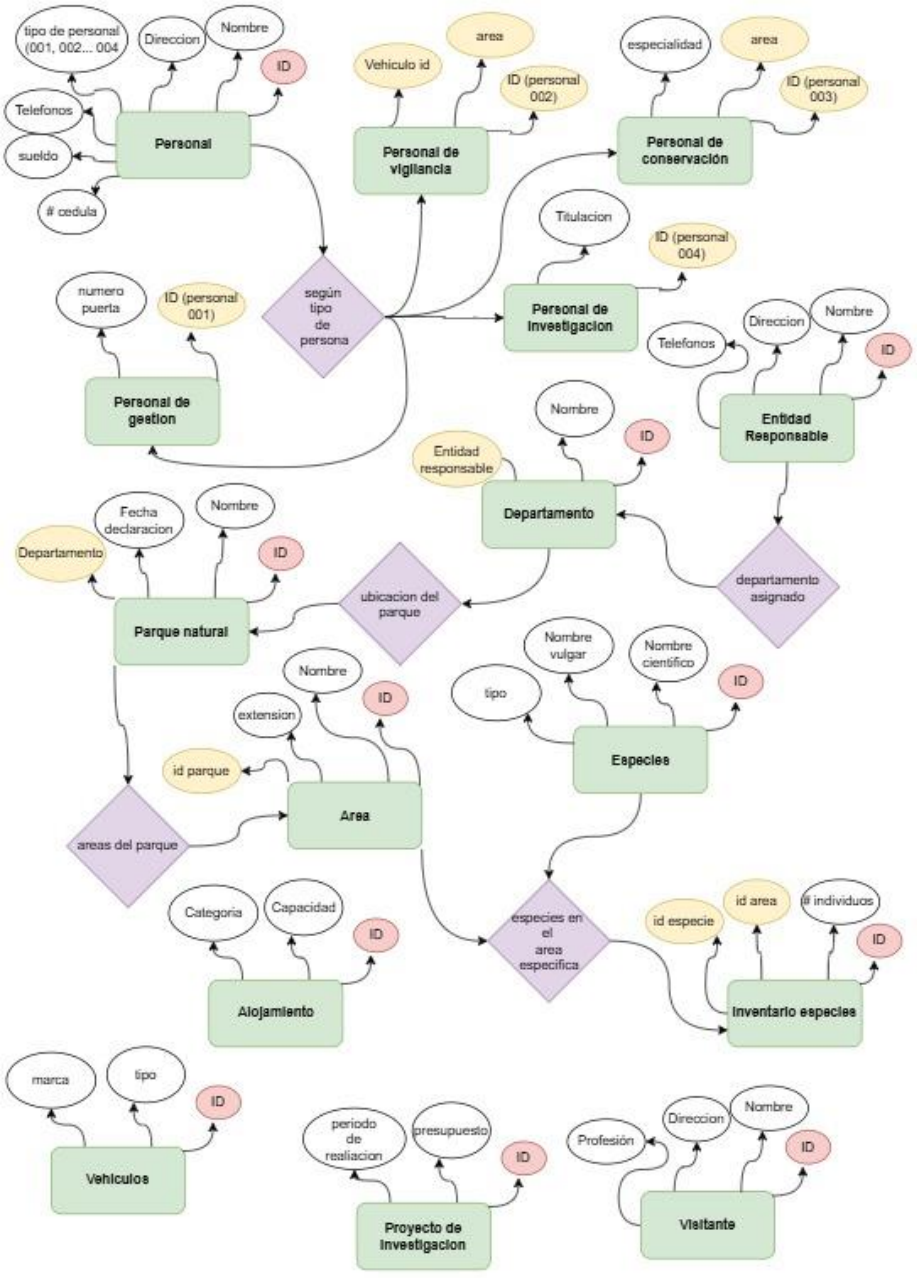
17. **Relación entre Investigador y Proyecto**

    - **Descripción:** Registra los proyectos de investigación y los investigadores involucrados.
    - **Atributos:** Investigador (FK), Proyecto (FK).

18. **Relación entre Visitante y Alojamiento**

    - **Descripción:** Lleva un registro de los alojamientos ocupados por los visitantes.
    - **Atributos:** Visitante (FK), Alojamiento (FK).

Grafico:


tipo de personal (001, 002... 004
Direccion
Nombre
ID
Telefonos
sueldo
cedula
Personal
Vehiculo id
area
ID (personal 002)
Personal de vigilancia
especialidad
area
ID (personal 003)
Personal de conservación
Titulacion
ID (personal 004)
Personal de investigacion
numero puerta
ID (personal 001)
Personal de gestion
según tipo de persona
Telefonos
Direccion
Nombre
ID
Entidad Responsable
Nombre
ID
Entidad responsable
Departamento
Departamento
Fecha declaracion
Nombre
ID
Parque natural
ubicacion del parque
departamento asignado
Nombre
ID
extension
id parque
Area
areas del parque
Nombre vulgar
Nombre cientifico
tipo
ID
Especies
Categoria
Capacidad
ID
Alojamiento
especies en el area especifica
id especie
id area
individuos
ID
Inventario especies
marca
tipo
ID
Vehiculos
periodo de realiacion
presupuesto
ID
Proyecto de Investigacion
Profesión
Direccion
Nombre
ID
Visitante

Construcción del modelo lógico

El modelo lógico de la base de datos ParquesNaturalesColombia es una representación más detallada y técnica del modelo conceptual, en la que se especifican las relaciones entre las tablas y se definen los tipos de datos de cada atributo. A continuación, se presenta una descripción del modelo lógico de las tablas y sus relaciones:

1. **Visitantes**
    - **Descripción:** Almacena información básica sobre los visitantes de los parques naturales.
    - **Atributos:**
        - id INT PRIMARY KEY AUTO_INCREMENT
        - nombre VARCHAR(50) NOT NULL
        - direccion VARCHAR(50) NOT NULL
        - profesion VARCHAR(50)
2. **Proyectos de Investigación**
    - **Descripción:** Gestiona los proyectos de investigación que se llevan a cabo en los parques.
    - **Atributos:**
        - id INT PRIMARY KEY AUTO_INCREMENT
        - nombre VARCHAR(50) NOT NULL
        - presupuesto DECIMAL(15,2) NOT NULL
        - fecha_inicio DATE NOT NULL
        - fecha_fin DATE NOT NULL

3. **Entidad Responsable**
    - **Descripción:** Registra las entidades encargadas de la gestión de los parques naturales.
    - **Atributos:**
        - id INT PRIMARY KEY AUTO_INCREMENT
        - nombre VARCHAR(50) NOT NULL
        - direccion VARCHAR(50) NOT NULL
        - telefono INT
4. **Especies**
    - **Descripción:** Clasifica las especies presentes en los parques naturales.
    - **Atributos:**
        - id INT PRIMARY KEY AUTO_INCREMENT
        - nombre_cientifico VARCHAR(50) NOT NULL
        - nombre_vulgar VARCHAR(50) NOT NULL
        - categoria ENUM('vegetal', 'animal', 'mineral') NOT NULL
5. **Vehículos**
    - **Descripción:** Registra los vehículos utilizados en la vigilancia de los parques.
    - **Atributos:**
        - id INT PRIMARY KEY AUTO_INCREMENT
        - tipo VARCHAR(25) NOT NULL
        - marca VARCHAR(25) NOT NULL

6. **Alojamientos**
    - **Descripción:** Gestiona los alojamientos disponibles en los parques para los visitantes.
    - **Atributos:**
        - id INT PRIMARY KEY AUTO_INCREMENT
        - nombre VARCHAR(25) NOT NULL
        - capacidad INT NOT NULL
        - categoria ENUM('hotel', 'cabaña') NOT NULL
7. **Parques Naturales**
    - **Descripción:** Almacena información general sobre cada parque natural.
    - **Atributos:**
        - id INT PRIMARY KEY AUTO_INCREMENT
        - nombre VARCHAR(25) NOT NULL
        - fecha_declaracion DATE NOT NULL
8. **Departamentos**
    - **Descripción:** Registra los departamentos que gestionan los parques y sus entidades responsables.
    - **Atributos:**
        - id INT PRIMARY KEY AUTO_INCREMENT
        - nombre VARCHAR(25) NOT NULL
        - entidad_responsable INT, FOREIGN KEY(entidad_responsable) REFERENCES entidad_responsable(id)

9. **Áreas**
    - **Descripción:** Divide los parques en áreas específicas, cada una con su extensión.
    - **Atributos:**
        - id INT PRIMARY KEY AUTO_INCREMENT
        - nombre VARCHAR(25) NOT NULL
        - extension DECIMAL(10,2) NOT NULL
        - parque INT, FOREIGN KEY(parque) REFERENCES parques_naturales(id)
10. **Personal**
    - **Descripción:** Almacena información sobre el personal que trabaja en los parques naturales.
    - **Atributos:**
        - id INT PRIMARY KEY AUTO_INCREMENT
        - documento INT NOT NULL
        - nombre VARCHAR(50) NOT NULL
        - direccion VARCHAR(50) NOT NULL
        - telefono VARCHAR(15) NOT NULL
        - sueldo DECIMAL(10,2) NOT NULL
        - tipo ENUM('Personal de Gestión', 'Personal de Vigilancia', 'Personal de Conservación', 'Personal Investigador') NOT NULL
        - parque INT, FOREIGN KEY(parque) REFERENCES parques_naturales(id)

11. **Inventario de Especies**

    - **Descripción:** Lleva un registro del número de individuos de cada especie en cada área del parque.
    - **Atributos:**
        - id INT PRIMARY KEY AUTO_INCREMENT
        - numero_individuos INT NOT NULL
        - area INT, FOREIGN KEY(area) REFERENCES area(id)
        - especie INT, FOREIGN KEY(especie) REFERENCES especies(id)

12. **Personal de Gestión**

    - **Descripción:** Detalla el personal encargado de la gestión de los visitantes en las entradas de los parques.
    - **Atributos:**
        - id INT, PRIMARY KEY(id), FOREIGN KEY(id) REFERENCES personal(id)
        - nombre VARCHAR(50) NOT NULL
        - puerta INT

13. **Personal de Investigación**

    - **Descripción:** Registra los investigadores y sus titulaciones.
    - **Atributos:**
        - id INT, PRIMARY KEY(id), FOREIGN KEY(id) REFERENCES personal(id)
        - nombre VARCHAR(50) NOT NULL
        - titulacion VARCHAR(25)

14. **Personal de Conservación**

- **Descripción:** Gestiona el personal encargado de la conservación de las áreas del parque y sus especialidades.
- **Atributos:**
  - id INT, PRIMARY KEY(id), FOREIGN KEY(id) REFERENCES personal(id)
  - nombre VARCHAR(50) NOT NULL
  - area INT, FOREIGN KEY(area) REFERENCES area(id)
  - especialidad VARCHAR(25)

15. **Personal de Vigilancia**

- **Descripción:** Registra el personal encargado de la vigilancia del parque, sus áreas asignadas y los vehículos utilizados.
- **Atributos:**
  - id INT, PRIMARY KEY(id), FOREIGN KEY(id) REFERENCES personal(id)
  - nombre VARCHAR(50) NOT NULL
  - area INT, FOREIGN KEY(area) REFERENCES area(id)
  - vehiculo INT, FOREIGN KEY(vehiculo) REFERENCES vehiculos(id)

16. **Relación entre Departamento y Parque**

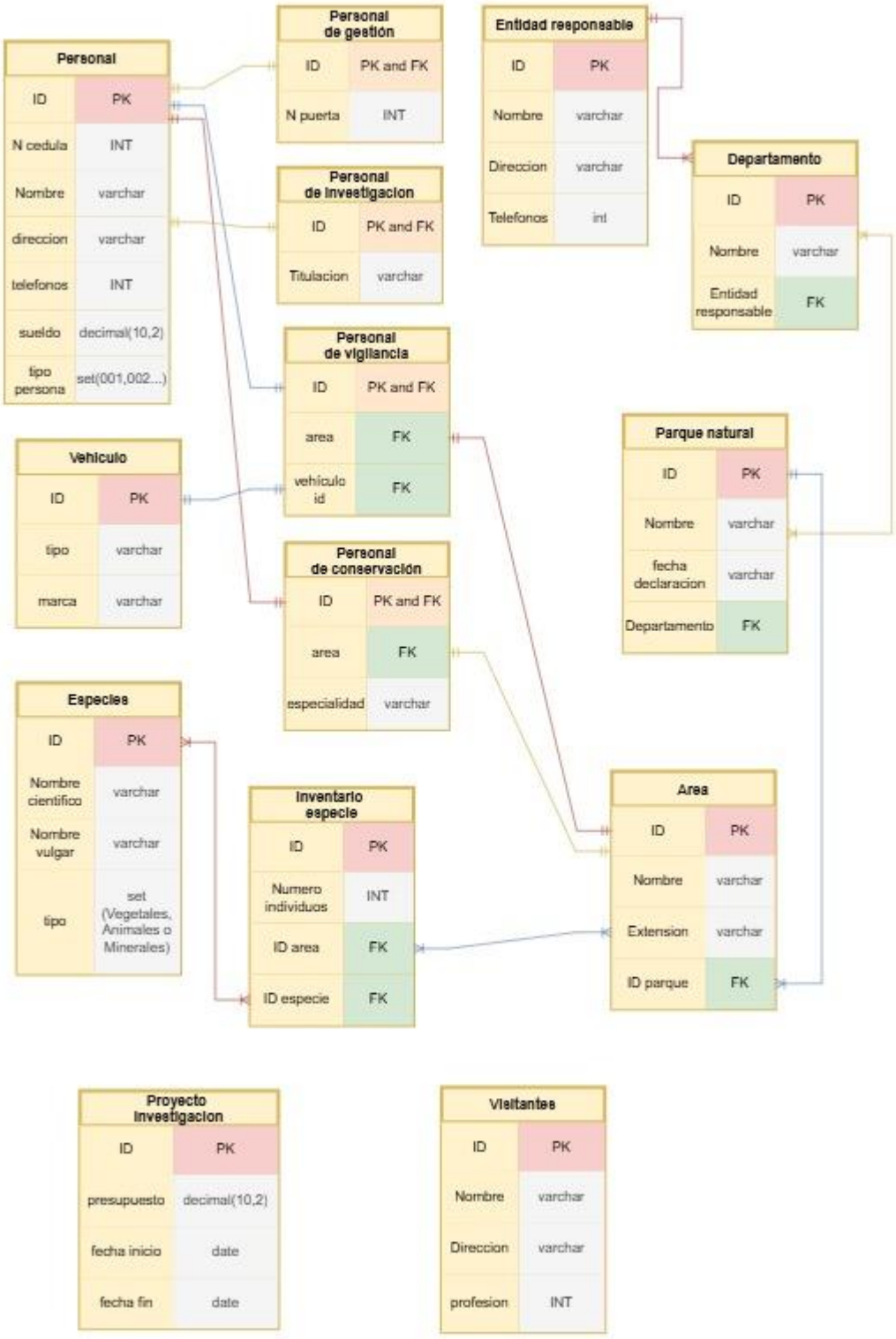
- **Descripción:** Gestiona qué parques son administrados por cada departamento.
- **Atributos:**
  - departamento INT, FOREIGN KEY(departamento) REFERENCES departamentos(id)

        - parque INT, FOREIGN KEY(parque) REFERENCES parques_naturales(id)
        - PRIMARY KEY(departamento, parque)

17. **Relación entre Investigador y Proyecto**
    - **Descripción:** Registra los proyectos de investigación y los investigadores involucrados.
    - **Atributos:**
        - investigador INT, FOREIGN KEY(investigador) REFERENCES personal_investigacion(id)
        - proyecto INT, FOREIGN KEY(proyecto) REFERENCES proyectos_investigacion(id)
        - PRIMARY KEY(investigador, proyecto)

18. **Relación entre Visitante y Alojamiento**
    - **Descripción:** Lleva un registro de los alojamientos ocupados por los visitantes.
    - **Atributos:**
        - visitante INT, FOREIGN KEY(visitante) REFERENCES visitantes(id)
        - alojamiento INT, FOREIGN KEY(alojamiento) REFERENCES alojamientos(id)
        - PRIMARY KEY(visitante, alojamiento)

Grafico:

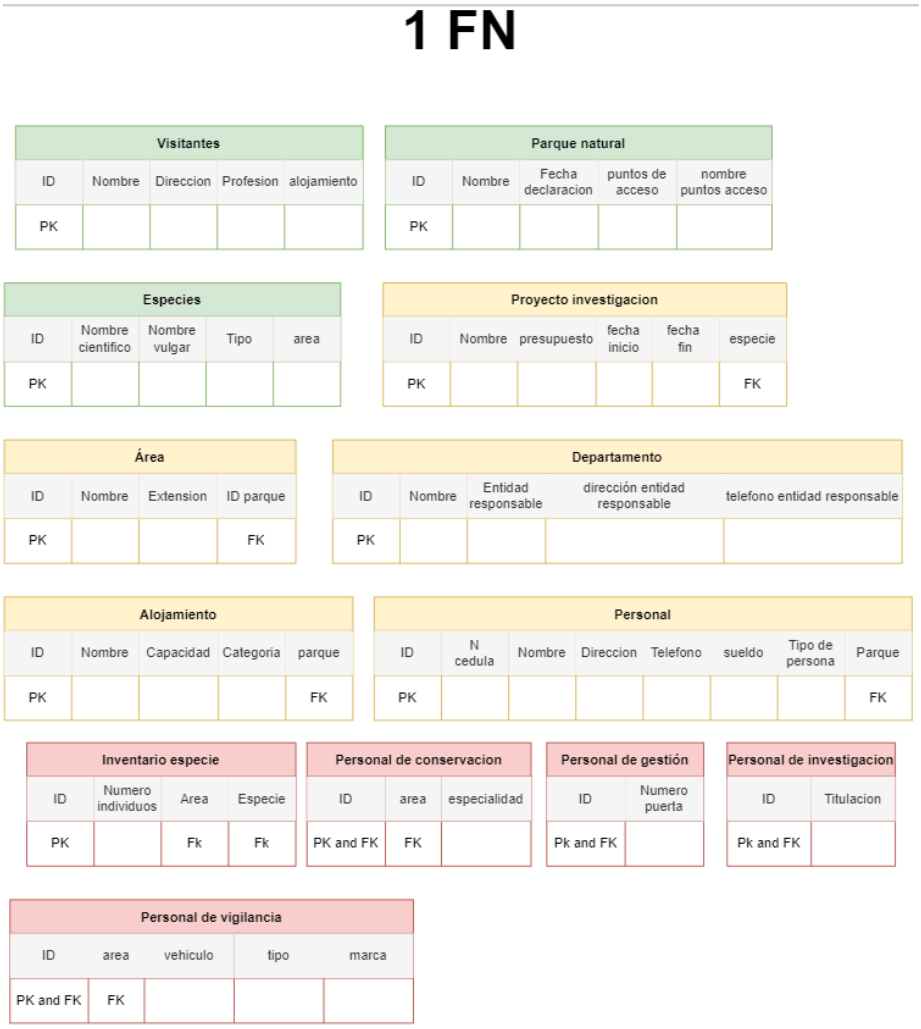

Personal
ID PK
N cedula INT
Nombre varchar
direccion varchar
telefonos INT
sueldo decimal(10,2)
tipo persona set(001,002...)
Personal de gestión
ID PK and FK
N puerta INT
Personal de investigacion
ID PK and FK
Titulacion varchar
Personal de vigilancia
ID PK and FK
area FK
vehiculo id FK
Personal de conservación
ID PK and FK
area FK
especialidad varchar
Vehiculo
ID PK
tipo varchar
marca varchar
Especies
ID PK
Nombre cientifico varchar
Nombre vulgar varchar
tipo set (Vegetales, Animales o Minerales)
Inventario especie
ID PK
Numero individuos INT
ID area FK
ID especie FK
Entidad responsable
ID PK
Nombre varchar
Direccion varchar
Telefonos int
Departamento
ID PK
Nombre varchar
Entidad responsable FK
Parque natural
ID PK
Nombre varchar
fecha declaracion varchar
Departamento FK
Area
ID PK
Nombre varchar
Extension varchar
ID parque FK
Proyecto investigacion
ID PK
presupuesto decimal(10,2)
fecha inicio date
fecha fin date
Visitantes
ID PK
Nombre varchar
Direccion varchar
profesion INT

Normalización del modelo lógico

Primera forma normal (1FN)

Para que una tabla esté en la Primera Forma Normal (1FN), debe cumplir con los siguientes requisitos: cada columna debe contener solo valores atómicos y cada fila debe ser única. En el caso de nuestro proyecto ParquesNaturalesColombia, todas las tablas cumplen con la 1FN. Por ejemplo, la tabla visitantes tiene columnas como id, nombre, direccion, y profesion, cada una conteniendo un único valor atómico por registro. No hay grupos repetitivos ni múltiples valores en una sola columna.

Grafico:



Segunda Forma Normal (2FN)

Para alcanzar la Segunda Forma Normal (2FN), una tabla primero debe estar en 1FN y luego, todos los atributos no clave deben depender completamente de la clave primaria. Esto significa que no puede haber dependencias parciales en tablas con claves primarias compuestas.

En nuestras tablas, hemos asegurado que todas las dependencias sean completas. Por ejemplo, en la tabla personal, todos los atributos (documento, nombre, direccion, telefono, sueldo, tipo, parque) dependen completamente de la clave primaria id. De manera similar, en la tabla departamentos, los atributos nombre y entidad_responsable dependen completamente de la clave primaria id.

Así, aseguramos que cada atributo de una tabla está completamente dependiente de la clave primaria, lo que elimina redundancias y mejora la integridad de los datos. Este proceso de normalización hasta la 2FN nos ayuda a estructurar la base de datos de manera eficiente, facilitando las consultas y el mantenimiento del sistema.

Grafico:

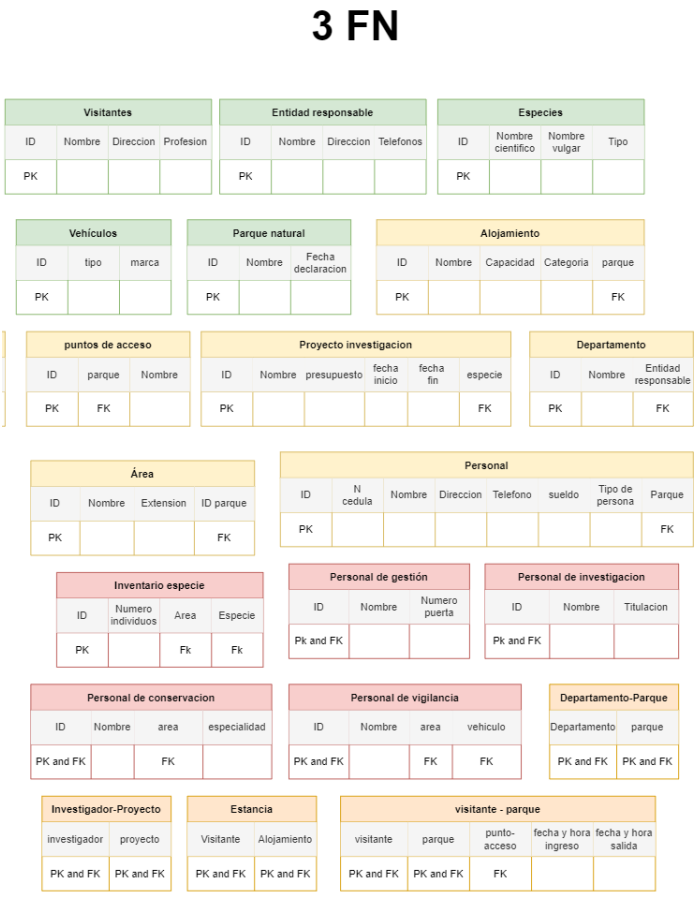
Tercera forma normal (3FN)

Para que una tabla esté en la Tercera Forma Normal (3FN), debe estar en 2FN y todos los atributos no clave deben depender solo de la clave primaria, sin dependencias transitivas. Esto significa que no puede haber dependencias entre atributos no clave.

En nuestro proyecto ParquesNaturalesColombia, hemos aplicado la 3FN para asegurar la integridad y eficiencia de la base de datos. Por ejemplo, en la tabla personal, todos sus atributos como documento, nombre, direccion, telefono, sueldo, tipo, y parque dependen directamente de la clave primaria id, sin dependencias entre ellos.

La normalización hasta la 3FN garantiza que cada atributo no clave esté relacionado solo con la clave primaria, eliminando redundancias, minimizando inconsistencias y facilitando el mantenimiento de la base de datos. Con estas medidas, logramos un diseño de base de datos robusto y eficiente para manejar la información sobre los parques naturales en Colombia.

Grafico:



Construcción del modelo Físico

La construcción del modelo físico de la base de datos ParquesNaturalesColombia implica traducir el modelo lógico a una implementación concreta en SQL. Este proceso incluye la creación de tablas, definición de tipos de datos, restricciones y relaciones entre las tablas.

Para cada tabla, definimos los tipos de datos más apropiados para cada columna. Por ejemplo, en la tabla visitantes, el id es un entero auto-incremental, mientras que nombre y direccion son de tipo varchar. Además, se establecen claves primarias para asegurar la unicidad de cada registro y se definen las claves foráneas para mantener la integridad referencial entre tablas.

También creamos relaciones entre tablas usando claves foráneas. Por ejemplo, la tabla departamentos tiene una clave foránea que referencia a la tabla entidad_responsable, asegurando que cada departamento esté asociado a una entidad responsable.

Adicionalmente, para optimizar el rendimiento de las consultas, definimos índices en las columnas que se utilizan frecuentemente en los filtros y ordenaciones de consultas.

Código:

```
-- ### base de datos ###
-- Crear base de datos
create database ParquesNaturalesColombia;
-- Usar base de datos
use ParquesNaturalesColombia;



-- ### Tablas ###
-- ##### Tablas sin foreign key #####
-- Creacion de la tabla (visitantes)
create table visitantes(
   id int primary key auto_increment,
   Nombre varchar(50) not null,
   direccion varchar(50) not null,
   profesion varchar(50)
```

```
);



-- creacion de la tabla (entidad_responsable)
create table entidad_responsable(
    id int primary key auto_increment,
    nombre varchar(50) not null,
    direccion varchar(50) not null,
    telefono int
);


-- Creacion de la tabla (especies)
create table especies(
    id int primary key auto_increment,
    nombre_cientifico varchar(50) not null,
    nombre_vulgar varchar(50) not null,
    categoria enum('vegetal','animal','mineral') not null
);

-- creacion de la tabla (proyectos_investigacion)
create table proyectos_investigacion (
    id int primary key auto_increment,
    nombre varchar(50) not null,
    presupuesto decimal(15,2) not null,
    fecha_inicio date not null,
    fecha_fin date not null,
    especie int,
    foreign key (especie) references especies(id)
);

-- Creacion de la tabla (vehiculos)
create table vehiculos(
    id int primary key auto_increment,
    tipo varchar(25) not null,
```

```
    marca varchar(25) not null
);



-- creacion de la tabla (parques_naturales)
create table parques_naturales(
    id int primary key auto_increment,
    nombre varchar(25) not null,
    fecha_declaracion date not null
);

-- creacion de la tabla (alojamientos)
create table alojamientos(
    id int primary key auto_increment,
    nombre varchar(25) not null,
    capacidad int not null,
    categoria ENUM('hotel', 'cabaña') NOT NULL,
    parque int,
    foreign key (parque) references parques_naturales(id)
);

create table punto_acceso(
id int primary key auto_increment,
parque int not null,
foreign key(parque) references parques_naturales(id),
nombre varchar (50) not null
);

-- ##### Tablas con foreign key #####
-- creacion de tabla (departamentos)
create table departamentos(
    id int primary key auto_increment,
    nombre varchar(25) not null,
    entidad_responsable int,
    foreign key(entidad_responsable) references entidad_responsable(id)
```

```
);

-- creacion de la tabla (area)
create table area(
    id int primary key auto_increment,
    nombre varchar(25) not null,
    extension decimal(10,2) not null,
    parque int,
    foreign key(parque) references parques_naturales(id)
);

-- creacion de la tabla (personal)
create table personal(
    id int primary key auto_increment,
    documento int not null,
    nombre varchar(50) not null,
    direccion varchar(50) not null,
    telefono varchar(15) not null,
    sueldo decimal(10,2) not null,
    tipo enum('Personal de Gestión','Personal de Vigilancia','Personal de Conservación','Personal
Investigador') not null,
    parque int,
    foreign key(parque) references parques_naturales(id)
);


-- ##### Tablas con mas de 1 foreign key #####
-- creacion de la tabla (inventario especie)
create table inventario_especie(
    id int primary key auto_increment,
    numero_individuos int not null,
    area int,
    foreign key(area) references area(id),
    especie int,
    foreign key(especie) references especies(id)
);
```

```sql
-- creacion de tabla (personal_gestion)
CREATE TABLE personal_gestion (
    id INT,
    nombre VARCHAR(50) NOT NULL,
    punto_acceso INT,
    PRIMARY KEY (id),
    FOREIGN KEY (id) REFERENCES personal(id),
    foreign key (punto_acceso) references punto_acceso(id)
);
-- creacion de la tabla (personal_investigacion)
CREATE TABLE personal_investigacion (
    id INT,
    nombre VARCHAR(50) NOT NULL,
    titulacion varchar(25),
    PRIMARY KEY (id),
    FOREIGN KEY (id) REFERENCES personal(id)
);

-- creacion de la tabla (personal_conservacion)
CREATE TABLE personal_conservacion (
    id INT,
    nombre VARCHAR(50) NOT NULL,
    area int,
    especialidad varchar(25),
    PRIMARY KEY (id),
    FOREIGN KEY (id) REFERENCES personal(id),
    foreign key (area) references area(id)
);


-- creacion de la tabla (personal_vigilancia)
CREATE TABLE personal_vigilancia (
    id INT,
    nombre VARCHAR(50) NOT NULL,
    area int,
```

```
    vehiculo int,
    PRIMARY KEY (id),
    FOREIGN KEY (id) REFERENCES personal(id),
    foreign key (area) references area(id),
    foreign key (vehiculo) references vehiculos(id)
);

-- ### relaciones ###
-- creacion de la tabla de relacion entre departamento y parque
CREATE TABLE departamento_parque (
    departamento INT,
    parque INT,
    PRIMARY KEY (departamento, parque),
    FOREIGN KEY (departamento) REFERENCES departamentos(id),
    FOREIGN KEY (parque) REFERENCES parques_naturales(id)
);
-- creacion de la tabla de relacion entre investigador y proyecto
CREATE TABLE investigador_proyecto (
    investigador INT,
    proyecto INT,
    PRIMARY KEY (investigador, proyecto),
    FOREIGN KEY (investigador) REFERENCES personal_investigacion(id),
    FOREIGN KEY (proyecto) REFERENCES proyectos_investigacion(id)
);

-- creacion de la tabla de relacion entre visitante y alojamiento
CREATE TABLE visitante_alojamiento (
    visitante INT,
    alojamiento INT,
    PRIMARY KEY (visitante, alojamiento),
    FOREIGN KEY (visitante) REFERENCES visitantes(id),
    FOREIGN KEY (alojamiento) REFERENCES alojamientos(id)
);

-- creacion de la tabla de relacion entre visitante y parque
CREATE TABLE visitante_parque (
```

```
    visitante INT,
    parque INT,
    punto_acceso int,
    fecha_ingreso TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (visitante, parque),
    FOREIGN KEY (visitante) REFERENCES visitantes(id),
    FOREIGN KEY (parque) REFERENCES parques_naturales(id),
    FOREIGN KEY (punto_acceso) REFERENCES punto_acceso(id)
);



-- crear tabla para el registro de las acciones del administrador

CREATE TABLE history_administrador (
    accion VARCHAR(10) NOT NULL,
    registro enum('entidad_responsable','departamento','parque'),
    hora_actual TIMESTAMP DEFAULT CURRENT_TIMESTAMP

);
```

Diagrama E-R

El diagrama Entidad-Relación (ER) de nuestra base de datos ParquesNaturalesColombia es una representación visual que muestra cómo se interrelacionan las entidades principales de nuestro sistema. Este diagrama nos ayuda a entender y comunicar la estructura de la base de datos de manera clara y sencilla.

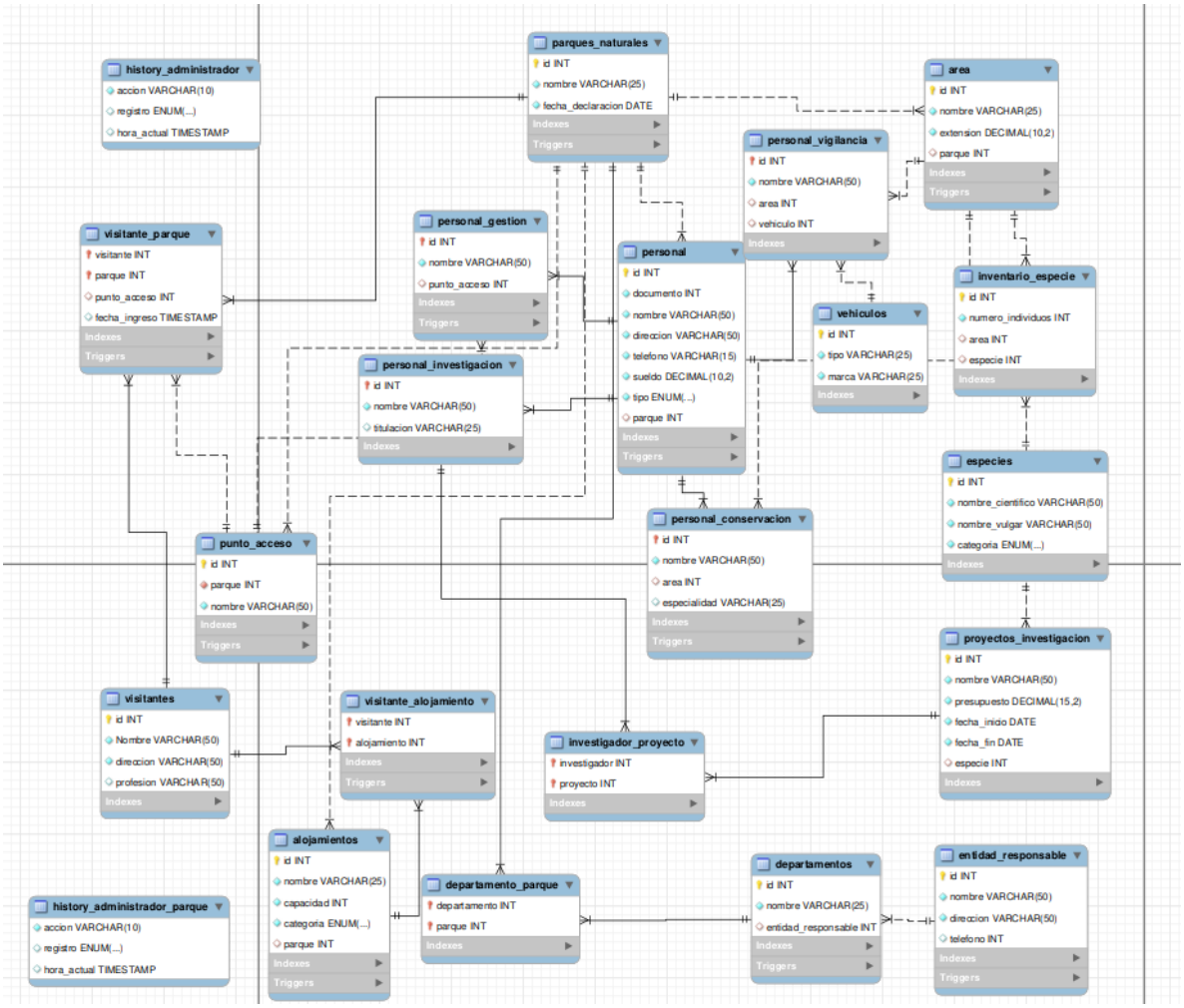
Las principales entidades en el diagrama incluyen visitantes, proyectos_investigacion, entidad_responsable, especies, vehiculos, alojamientos, parques_naturales, departamentos, area, personal, y varias tablas de relación para manejar las interacciones entre estas entidades.

Cada entidad está representada por un rectángulo y contiene sus atributos clave, como en visitantes, donde tenemos id, nombre, direccion, y profesion. Las relaciones entre las

entidades se representan con líneas que conectan los rectángulos, y los diamantes describen el tipo de relación. Por ejemplo, la relación entre departamentos y parques_naturales es gestionada por la tabla intermedia departamento_parque.

Las claves primarias y foráneas se muestran claramente, indicando cómo se mantiene la integridad referencial entre las entidades. Esto incluye relaciones uno a muchos, como entre departamentos y entidad_responsable, así como relaciones muchos a muchos, como entre proyectos_investigacion y personal_investigacion, manejadas a través de la tabla investigador_proyecto.

Grafico:

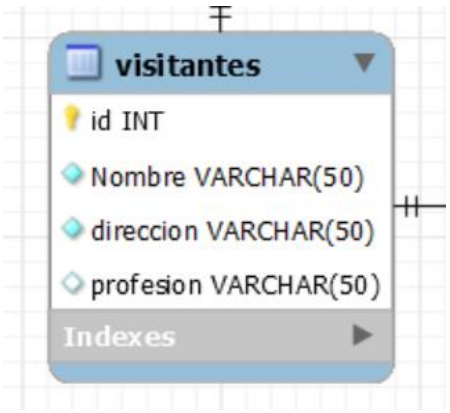


Tablas

- **Visitantes:** La tabla visitantes está diseñada para almacenar información esencial sobre las personas que visitan los parques naturales. Esta tabla es fundamental para gestionar y analizar datos sobre el flujo de visitantes y sus características.

  La estructura de la tabla visitantes incluye los siguientes campos:

  - **id:** Un identificador único para cada visitante, que se incrementa automáticamente. Este campo sirve como clave primaria, garantizando que cada registro en la tabla sea único.
  - **nombre:** Un campo de tipo varchar(50) que almacena el nombre del visitante. Es un dato crucial para identificar y personalizar la información de cada visitante.
  - **direccion:** Otro campo de tipo varchar(50) que registra la dirección del visitante. Este dato es importante para posibles comunicaciones y análisis demográficos.
  - **profesion:** Un campo de tipo varchar(50) que contiene la profesión del visitante. Aunque no es obligatorio, proporciona información valiosa para entender el perfil de los visitantes y realizar estudios de mercado o segmentación**.**



- **entidad_responsable:** La tabla entidad_responsable está diseñada para almacenar información sobre las entidades encargadas de la gestión de los parques naturales en

cada departamento. Esta tabla es crucial para identificar y organizar las diferentes organizaciones responsables de la administración y conservación de estos espacios.

La estructura de la tabla entidad_responsable incluye los siguientes campos:

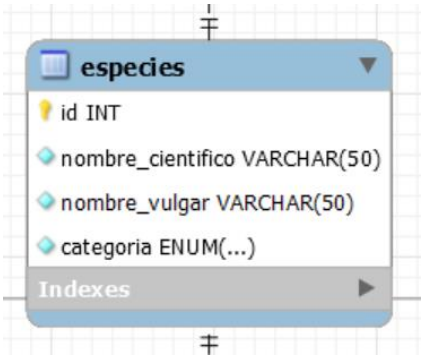
- **id:** Un identificador único para cada entidad responsable, que se incrementa automáticamente. Este campo sirve como clave primaria, asegurando la unicidad de cada registro en la tabla.
- **nombre:** Un campo de tipo varchar(50) que almacena el nombre de la entidad responsable. Este dato es fundamental para identificar a las distintas organizaciones que gestionan los parques.
- **direccion:** Un campo de tipo varchar(50) que registra la dirección de la entidad responsable. Este dato es esencial para la comunicación y localización de la entidad.
- **telefono:** Un campo de tipo int que contiene el número de teléfono de la entidad responsable. Este dato es importante para establecer contacto directo con la entidad cuando sea necesario.

- **especies:** La tabla especies está diseñada para almacenar información detallada sobre las especies que habitan en las áreas de los parques naturales. Esta tabla es esencial para la gestión y conservación de la biodiversidad en estos espacios.

  La estructura de la tabla especies incluye los siguientes campos:

  - **id:** Un identificador único para cada especie, que se incrementa automáticamente. Este campo sirve como clave primaria, garantizando que cada registro en la tabla sea único.

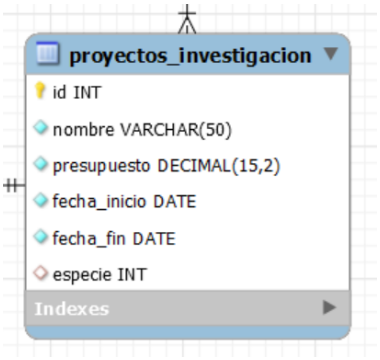
- **nombre_cientifico:** Un campo de tipo varchar(50) que almacena el nombre científico de la especie. Este dato es crucial para la identificación precisa y la clasificación taxonómica.
- **nombre_vulgar:** Un campo de tipo varchar(50) que contiene el nombre común o vulgar de la especie. Este dato facilita la comprensión y el manejo de la información por parte de personas no especializadas.
- **categoria:** Un campo de tipo enum que clasifica la especie en una de las siguientes categorías: 'vegetal', 'animal', o 'mineral'. Esta clasificación es importante para la organización y análisis de la biodiversidad en los parques.



- **proyectos_investigacion:** La tabla proyectos_investigacion está diseñada para almacenar información sobre los proyectos de investigación que se llevan a cabo en los parques naturales. Esta tabla es fundamental para gestionar y seguir el progreso de los estudios científicos y de conservación realizados en estos espacios.

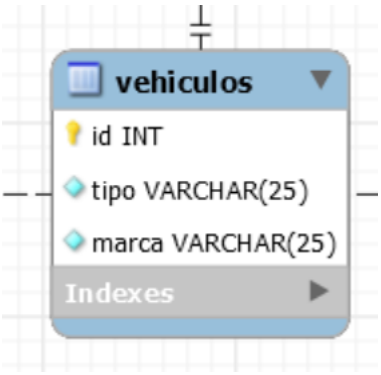
  La estructura de la tabla proyectos_investigacion incluye los siguientes campos:
  - **id:** Un identificador único para cada proyecto de investigación, que se incrementa automáticamente. Este campo sirve como clave primaria, asegurando que cada registro en la tabla sea único.
  - **nombre:** Un campo de tipo varchar(50) que almacena el nombre del proyecto. Este dato es esencial para identificar y diferenciar cada proyecto.

- **presupuesto:** Un campo de tipo decimal(15,2) que registra el presupuesto asignado al proyecto. Este dato es crucial para la gestión financiera y la planificación de recursos.
- **fecha_inicio:** Un campo de tipo date que indica la fecha de inicio del proyecto. Este dato es importante para el seguimiento y la planificación temporal del proyecto.
- **fecha_fin:** Un campo de tipo date que indica la fecha de finalización del proyecto. Este dato ayuda a establecer los plazos y evaluar el cumplimiento de los objetivos.



- **vehículos:** La tabla vehiculos está diseñada para almacenar información sobre los vehículos utilizados en los parques naturales. Esta tabla es esencial para gestionar los recursos de transporte y vigilancia dentro de estos espacios.

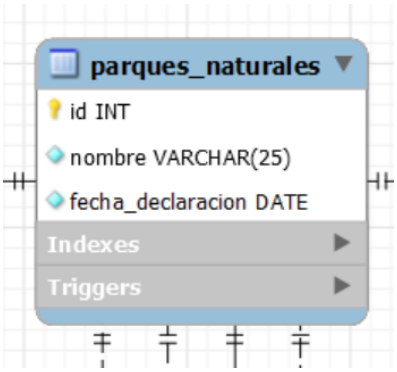
  La estructura de la tabla vehiculos incluye los siguientes campos:
  - **id:** Un identificador único para cada vehículo, que se incrementa automáticamente. Este campo sirve como clave primaria, garantizando que cada registro en la tabla sea único.
  - **tipo:** Un campo de tipo varchar(25) que almacena el tipo de vehículo. Este dato es crucial para categorizar y gestionar los diferentes tipos de transporte disponibles.
  - **marca:** Un campo de tipo varchar(25) que registra la marca del vehículo. Este dato es importante para la identificación y el mantenimiento de los vehículos.



- **parques_naturales:** La tabla **parques_naturales** está diseñada para almacenar información sobre los parques naturales gestionados. Esta tabla es fundamental para registrar y administrar los datos básicos de cada parque, facilitando su manejo y conservación.

  La estructura de la tabla **parques_naturales** incluye los siguientes campos:

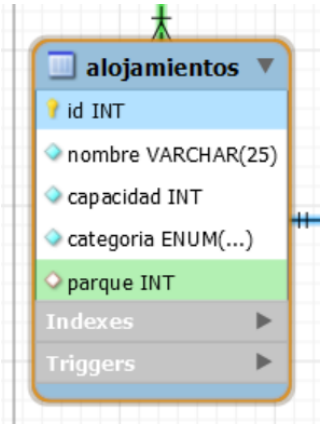
  - **id**: Un identificador único para cada parque natural, que se incrementa automáticamente. Este campo sirve como clave primaria, garantizando que cada registro en la tabla sea único.
  - **nombre**: Un campo de tipo **varchar(25)** que almacena el nombre del parque. Este dato es esencial para identificar y diferenciar cada parque natural.
  - **fecha_declaracion**: Un campo de tipo **date** que indica la fecha en la que el parque fue declarado como área protegida. Este dato es importante para llevar un registro histórico y legal del establecimiento del parque.



- **alojamientos:** La tabla **alojamientos** está diseñada para almacenar información sobre los alojamientos disponibles en los parques naturales. Esta tabla es esencial para gestionar los recursos de hospedaje y asegurar una adecuada administración de los visitantes.

  La estructura de la tabla **alojamientos** incluye los siguientes campos:

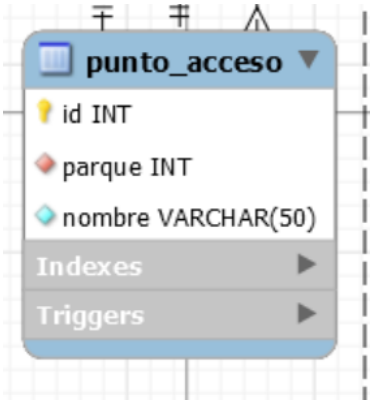
  - **id**: Un identificador único para cada alojamiento, que se incrementa automáticamente. Este campo sirve como clave primaria, garantizando que cada registro en la tabla sea único.
  - **nombre**: Un campo de tipo **varchar(25)** que almacena el nombre del alojamiento. Este dato es crucial para identificar y diferenciar cada opción de hospedaje.
  - **capacidad**: Un campo de tipo **int** que registra la capacidad de huéspedes del alojamiento. Este dato es importante para la gestión y planificación del uso de los recursos de hospedaje.
  - **categoria**: Un campo de tipo **enum** que clasifica el alojamiento en categorías como 'hotel' o 'cabaña'. Esta clasificación facilita la organización y selección de alojamientos según las necesidades y preferencias de los visitantes.



- **punto_acceso:** La tabla **punto_acceso** está diseñada para almacenar información sobre los diferentes puntos de acceso a los parques naturales. Esta tabla es esencial para gestionar y controlar las entradas y salidas de visitantes y personal en los parques.

La estructura de la tabla **punto_acceso** incluye los siguientes campos:

- **id**: Un identificador único para cada punto de acceso, que se incrementa automáticamente. Este campo sirve como clave primaria, garantizando que cada registro en la tabla sea único.
- **numero**: Un campo de tipo **int** que almacena el número de identificación del punto de acceso. Este dato es crucial para la gestión y control de las diferentes entradas y salidas en los parques.
- **nombre**: Un campo de tipo **varchar(50)** que registra el nombre del punto de acceso. Este dato es importante para la identificación y ubicación precisa de cada entrada.



- **departamentos:** La tabla **departamentos** está diseñada para almacenar información sobre los diferentes departamentos que administran los parques naturales. Esta tabla es fundamental para gestionar la relación entre los parques y las entidades responsables de su gestión.

  La estructura de la tabla **departamentos** incluye los siguientes campos:

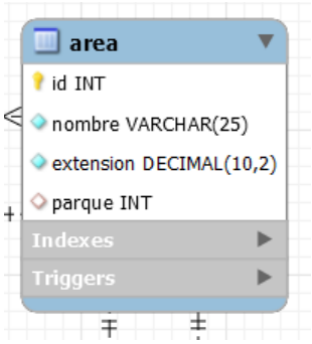
  - **id**: Un identificador único para cada departamento, que se incrementa automáticamente. Este campo sirve como clave primaria, garantizando que cada registro en la tabla sea único.
  - **nombre**: Un campo de tipo **varchar(25)** que almacena el nombre del departamento. Este dato es crucial para identificar y diferenciar cada departamento.

- **entidad_responsable**: Un campo de tipo **int** que almacena el identificador de la entidad responsable de los parques en ese departamento. Este dato es importante para establecer la relación entre el departamento y la entidad que gestiona sus parques, y se define como una clave foránea que referencia a la tabla **entidad_responsable**.

- **area:** La tabla **area** está diseñada para almacenar información sobre las distintas áreas dentro de los parques naturales. Esta tabla es esencial para gestionar y mantener un registro detallado de las diferentes secciones de los parques.

  La estructura de la tabla **area** incluye los siguientes campos:

  - **id**: Un identificador único para cada área, que se incrementa automáticamente. Este campo sirve como clave primaria, garantizando que cada registro en la tabla sea único.
  - **nombre**: Un campo de tipo **varchar(25)** que almacena el nombre del área. Este dato es crucial para identificar y diferenciar cada área dentro del parque.
  - **extension**: Un campo de tipo **decimal(10,2)** que registra la extensión del área en hectáreas. Este dato es importante para gestionar los recursos y planificar las actividades de conservación.
  - **parque**: Un campo de tipo **int** que almacena el identificador del parque al que pertenece el área. Este dato establece una relación con la tabla **parques_naturales**, indicando a qué parque pertenece cada área, y se define como una clave foránea que referencia a la tabla **parques_naturales**.



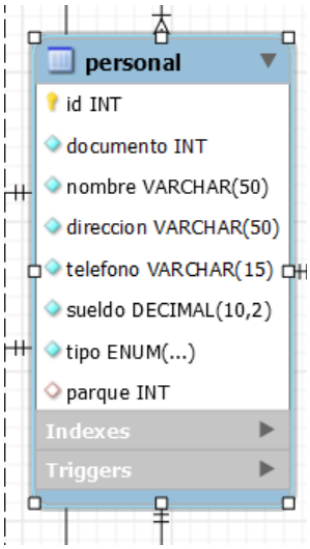
- personal: La tabla **personal** está diseñada para almacenar información sobre el personal que trabaja en los parques naturales. Esta tabla es crucial para gestionar los datos de los empleados y sus roles dentro de los parques.

  La estructura de la tabla **personal** incluye los siguientes campos:

  - **id**: Un identificador único para cada empleado, que se incrementa automáticamente. Este campo sirve como clave primaria, garantizando que cada registro en la tabla sea único.
  - **documento**: Un campo de tipo **int** que almacena el número de identificación del empleado. Este dato es esencial para la identificación personal y administrativa.
  - **nombre**: Un campo de tipo **varchar(50)** que registra el nombre completo del empleado. Este dato es importante para la identificación y gestión del personal.
  - **direccion**: Un campo de tipo **varchar(50)** que almacena la dirección del empleado. Este dato es útil para fines administrativos y de contacto.
  - **telefono**: Un campo de tipo **varchar(15)** que registra el número de teléfono del empleado, incluyendo el móvil. Este dato es importante para la comunicación interna.
  - **sueldo**: Un campo de tipo **decimal(10,2)** que almacena el sueldo del empleado. Este dato es esencial para la gestión financiera y de recursos humanos.
  - **tipo**: Un campo de tipo **enum** que clasifica al personal en diferentes roles: 'Personal de Gestión', 'Personal de Vigilancia', 'Personal de Conservación',

'Personal Investigador'. Este dato es fundamental para la organización y asignación de tareas.

- **parque**: Un campo de tipo **int** que almacena el identificador del parque donde trabaja el empleado. Este dato establece una relación con la tabla **parques_naturales**, indicando en qué parque trabaja cada miembro del personal, y se define como una clave foránea que referencia a la tabla **parques_naturales**.



- **inventario_especie:** La tabla **inventario_especie** está diseñada para almacenar información sobre las especies que habitan en las diferentes áreas de los parques naturales. Esta tabla es esencial para llevar un control detallado del número de individuos de cada especie y su distribución en las áreas del parque.

  La estructura de la tabla **inventario_especie** incluye los siguientes campos:

  - **id**: Un identificador único para cada registro de inventario, que se incrementa automáticamente. Este campo sirve como clave primaria, garantizando que cada registro en la tabla sea único.
  - **numero_individuos**: Un campo de tipo **int** que almacena el número de individuos de la especie en una determinada área. Este dato es crucial para el monitoreo y gestión de las poblaciones de especies.

- **area**: Un campo de tipo **int** que almacena el identificador del área donde se encuentra la especie. Este dato establece una relación con la tabla **area**, indicando en qué área del parque se encuentra la especie, y se define como una clave foránea que referencia a la tabla **area**.
- **especie**: Un campo de tipo **int** que almacena el identificador de la especie. Este dato establece una relación con la tabla **especies**, indicando qué especie está siendo inventariada, y se define como una clave foránea que referencia a la tabla **especies**.

- **personal_gestion:** La tabla **personal_gestion** está diseñada para almacenar información específica sobre el personal de gestión que trabaja en los parques naturales. Este tipo de personal se encarga de registrar los datos de los visitantes y se encuentra en las entradas del parque.

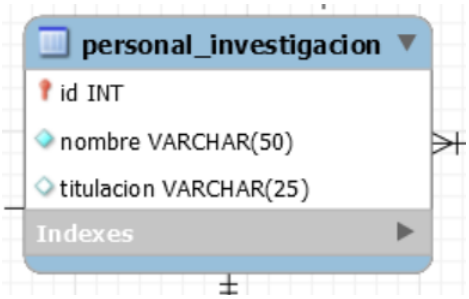
  La estructura de la tabla **personal_gestion** incluye los siguientes campos:
  - **id**: Un identificador único para cada miembro del personal de gestión, que coincide con el identificador en la tabla **personal**. Este campo sirve como clave primaria, garantizando que cada registro en la tabla sea único y se relaciona con la tabla **personal** mediante una clave foránea.
  - **nombre**: Un campo de tipo **varchar(50)** que almacena el nombre del miembro del personal de gestión. Este dato es importante para la identificación y gestión del personal.

- **puerta**: Un campo de tipo **int** que registra el número de la entrada del parque donde está ubicado el personal de gestión. Este dato es esencial para coordinar y organizar las actividades en las diferentes entradas del parque.

- **personal_investigacion:** La tabla **personal_investigacion** está diseñada para almacenar información específica sobre el personal de investigación que trabaja en los parques naturales. Este tipo de personal se dedica a realizar investigaciones sobre las especies y participa en diversos proyectos de investigación.

  La estructura de la tabla **personal_investigacion** incluye los siguientes campos:

  - **id**: Un identificador único para cada miembro del personal de investigación, que coincide con el identificador en la tabla **personal**. Este campo sirve como clave primaria, garantizando que cada registro en la tabla sea único y se relaciona con la tabla **personal** mediante una clave foránea.
  - **nombre**: Un campo de tipo **varchar(50)** que almacena el nombre del miembro del personal de investigación. Este dato es importante para la identificación y gestión del personal.
  - **titulacion**: Un campo de tipo **varchar(25)** que registra la titulación académica del investigador. Este dato es esencial para conocer la especialización y las competencias del personal de investigación.



- **personal_conservacion:** La tabla **personal_conservacion** está diseñada para almacenar información específica sobre el personal de conservación que trabaja en los parques naturales. Este tipo de personal se encarga de mantener y conservar diferentes áreas del parque.

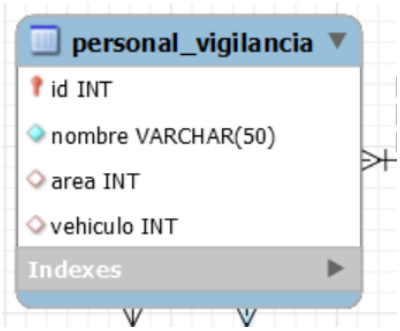
  La estructura de la tabla **personal_conservacion** incluye los siguientes campos:

  - **id**: Un identificador único para cada miembro del personal de conservación, que coincide con el identificador en la tabla **personal**. Este campo sirve como clave primaria, garantizando que cada registro en la tabla sea único y se relaciona con la tabla **personal** mediante una clave foránea.
  - **nombre**: Un campo de tipo **varchar(50)** que almacena el nombre del miembro del personal de conservación. Este dato es importante para la identificación y gestión del personal.
  - **area**: Un campo de tipo **int** que registra el identificador del área que el personal de conservación mantiene. Este dato establece una relación con la tabla **area**, indicando en qué área del parque trabaja el personal de conservación, y se define como una clave foránea que referencia a la tabla **area**.
  - **especialidad**: Un campo de tipo **varchar(25)** que almacena la especialidad del personal de conservación, como limpieza, caminos, etc. Este dato es esencial para conocer las competencias y responsabilidades específicas de cada miembro del personal de conservación.

- **personal_vigilancia:** La tabla **personal_vigilancia** está diseñada para almacenar información específica sobre el personal de vigilancia que trabaja en los parques naturales. Este tipo de personal se encarga de vigilar las áreas del parque, asegurando su seguridad y conservación.

  La estructura de la tabla **personal_vigilancia** incluye los siguientes campos:

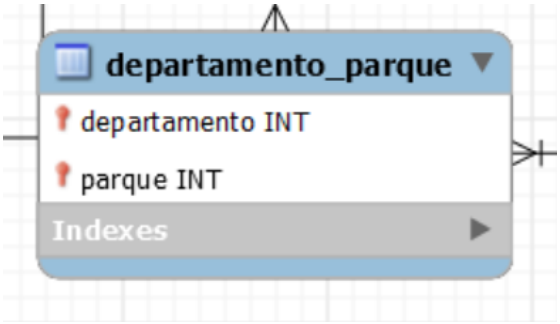
  - **id**: Un identificador único para cada miembro del personal de vigilancia, que coincide con el identificador en la tabla **personal**. Este campo sirve como clave primaria, garantizando que cada registro en la tabla sea único y se relaciona con la tabla **personal** mediante una clave foránea.
  - **nombre**: Un campo de tipo **varchar(50)** que almacena el nombre del miembro del personal de vigilancia. Este dato es importante para la identificación y gestión del personal.
  - **area**: Un campo de tipo **int** que registra el identificador del área que el personal de vigilancia recorre y protege. Este dato establece una relación con la tabla **area**, indicando en qué área del parque trabaja el personal de vigilancia, y se define como una clave foránea que referencia a la tabla **area**.
  - **vehiculo**: Un campo de tipo **int** que almacena el identificador del vehículo que utiliza el personal de vigilancia. Este dato establece una relación con la tabla **vehiculos**, indicando qué vehículo está asignado al personal de vigilancia, y se define como una clave foránea que referencia a la tabla **vehiculos**.



- **departamento_parque:** La tabla **departamento_parque** está diseñada para gestionar la relación entre los departamentos y los parques naturales en Colombia. Esta relación es crucial ya que un parque natural puede estar ubicado en más de un departamento y un departamento puede tener varios parques naturales.

  La estructura de la tabla **departamento_parque** incluye los siguientes campos:

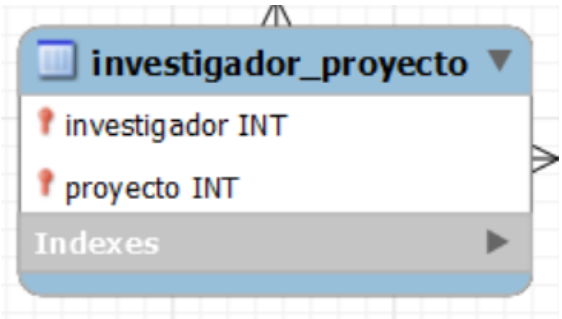
  - **departamento**: Un campo de tipo **int** que almacena el identificador único del departamento. Este dato establece una relación con la tabla **departamentos**, indicando qué departamento está relacionado con el parque natural, y se define como una clave foránea que referencia a la tabla **departamentos**.
  - **parque**: Un campo de tipo **int** que almacena el identificador único del parque natural. Este dato establece una relación con la tabla **parques_naturales**, indicando qué parque natural está relacionado con el departamento, y se define como una clave foránea que referencia a la tabla **parques_naturales**.
  - La clave primaria de la tabla **departamento_parque** está compuesta por los campos **departamento** y **parque**, asegurando que cada combinación de departamento y parque natural sea única.



- **investigador_proyecto**: La tabla **investigador_proyecto** está diseñada para gestionar la relación muchos a muchos entre los investigadores y los proyectos de investigación en los parques naturales. Esta relación es fundamental para registrar qué investigadores participan en qué proyectos y viceversa.

  La estructura de la tabla **investigador_proyecto** incluye los siguientes campos:

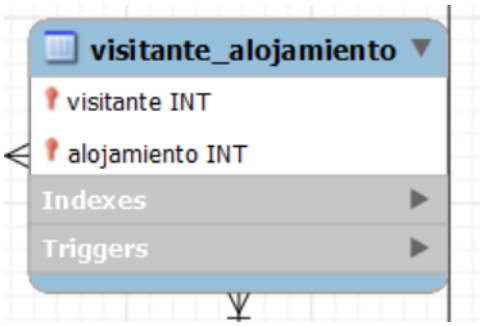
  - **investigador**: Un campo de tipo **int** que almacena el identificador único del investigador. Este dato establece una relación con la tabla **personal_investigacion**, indicando qué investigador está involucrado en el proyecto, y se define como una clave foránea que referencia a la tabla **personal_investigacion**.
  - **proyecto**: Un campo de tipo **int** que almacena el identificador único del proyecto de investigación. Este dato establece una relación con la tabla **proyectos_investigacion**, indicando en qué proyecto está involucrado el investigador, y se define como una clave foránea que referencia a la tabla **proyectos_investigacion**.
  - La clave primaria de la tabla **investigador_proyecto** está compuesta por los campos **investigador** y **proyecto**, asegurando que cada relación entre un investigador y un proyecto sea única.



- **visitante_alojamiento:** La tabla **visitante_alojamiento** está diseñada para gestionar la relación muchos a muchos entre los visitantes y los alojamientos disponibles en los parques naturales. Esta relación es crucial para registrar qué visitantes se alojan en qué tipo de alojamiento durante su estadía en los parques.

  La estructura de la tabla **visitante_alojamiento** incluye los siguientes campos:

- **visitante**: Un campo de tipo **int** que almacena el identificador único del visitante. Este dato establece una relación con la tabla **visitantes**, indicando qué visitante está alojado en qué alojamiento, y se define como una clave foránea que referencia a la tabla **visitantes**.
- **alojamiento**: Un campo de tipo **int** que almacena el identificador único del alojamiento. Este dato establece una relación con la tabla **alojamientos**, indicando en qué alojamiento se hospeda el visitante, y se define como una clave foránea que referencia a la tabla **alojamientos**.
- La clave primaria de la tabla **visitante_alojamiento** está compuesta por los campos **visitante** y **alojamiento**, asegurando que cada combinación de visitante y alojamiento sea única.



- **visitante_parque:** La tabla **visitante_parque** está diseñada para registrar la relación entre los visitantes y los parques naturales que visitan en Colombia. Esta tabla es fundamental para mantener un registro detallado de las visitas de los usuarios a diferentes áreas naturales protegidas.

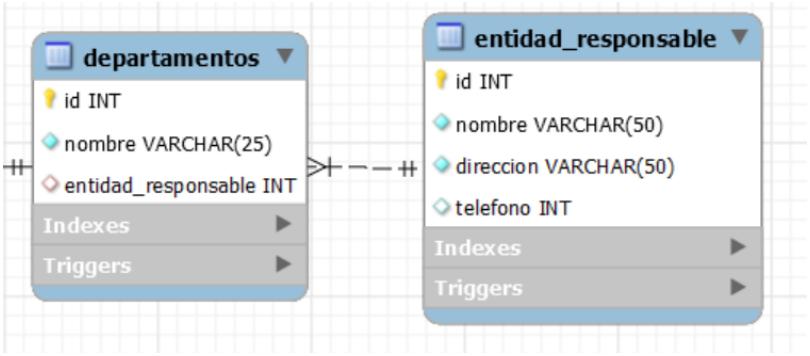
  La estructura de la tabla **visitante_parque** incluye los siguientes campos:
  - **visitante_id**: Un campo de tipo **int** que representa el identificador único del visitante. Este campo establece una relación con la tabla **visitantes**, indicando quién está visitando el parque, y se define como una clave foránea que referencia a la tabla **visitantes**.
  - **parque_id**: Un campo de tipo **int** que representa el identificador único del parque natural visitado. Este campo establece una relación con la tabla

parques_naturales, indicando a qué parque está asociado el registro de visita, y se define como una clave foránea que referencia a la tabla **parques_naturales**.

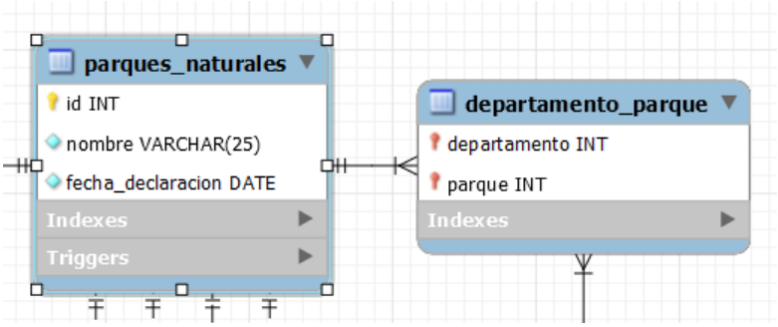
- **fecha_visita**: Un campo de tipo **date** que registra la fecha en que el visitante realizó la visita al parque natural.
- La tabla **visitante_parque** tiene como clave primaria la combinación de los campos **visitante_id** y **parque_id**, asegurando que cada visita de un visitante a un parque natural sea única y esté correctamente registrada.

Relaciones entre tablas

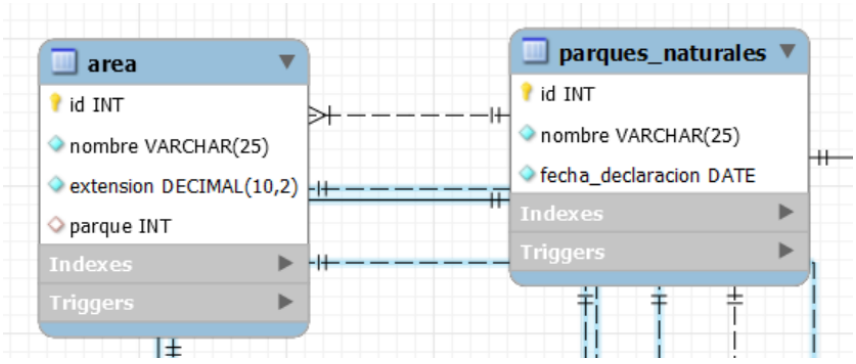
1. **departamentos** y **entidad_responsable**:
    - La tabla **departamentos** tiene una relación con la tabla **entidad_responsable** a través del campo **entidad_responsable**. Esto permite asociar cada departamento con una entidad responsable específica que gestiona los parques naturales en dicho departamento.



2. **parques_naturales** y **departamento_parque**:
    - La tabla **parques_naturales** está relacionada con la tabla **departamento_parque** mediante la relación muchos a muchos definida por los campos **id** en **parques_naturales** y las claves foráneas **departamento** y **parque** en **departamento_parque**. Esto permite asignar múltiples departamentos a un parque natural y viceversa.

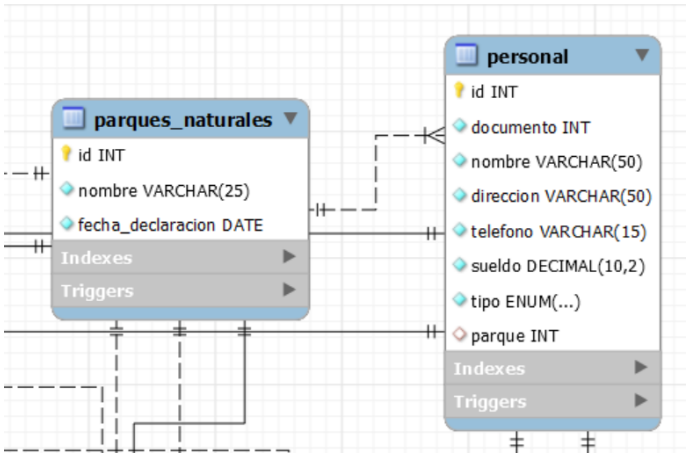


3. **area** y **parques_naturales**:
    - La tabla **area** está relacionada con la tabla **parques_naturales** mediante el campo **parque** en **area**, que actúa como clave foránea hacia **parques_naturales**. Esta relación permite definir áreas específicas dentro de cada parque natural, asociando extensiones de terreno con un parque particular.



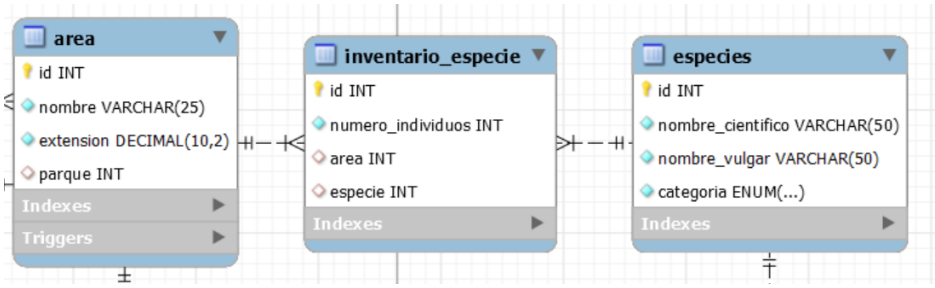
4. **personal** y **parques_naturales**:
    - La tabla **personal** tiene una relación con la tabla **parques_naturales** a través del campo **parque** en **personal**, que actúa como clave foránea hacia **parques_naturales**. Esta relación permite asignar personal a áreas

específicas dentro de cada parque natural según su función (gestión, vigilancia, conservación o investigación).



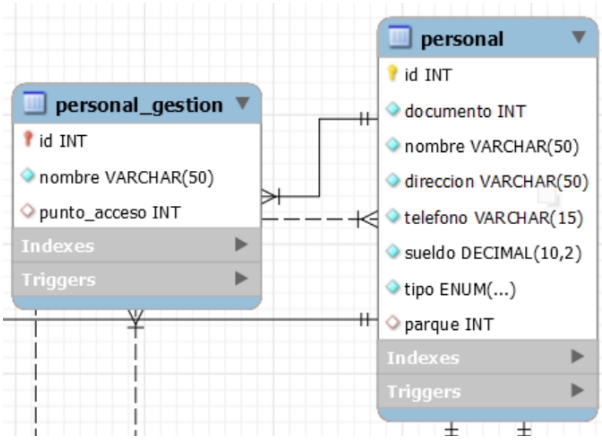
5. **inventario_especie** y **area**, **especies**:

   - La tabla **inventario_especie** está relacionada con las tablas **area** y **especies** mediante las claves foráneas **area** y **especie** respectivamente. Esto permite registrar la cantidad de individuos de cada especie en áreas específicas dentro de los parques naturales, facilitando la gestión y conservación de la biodiversidad.



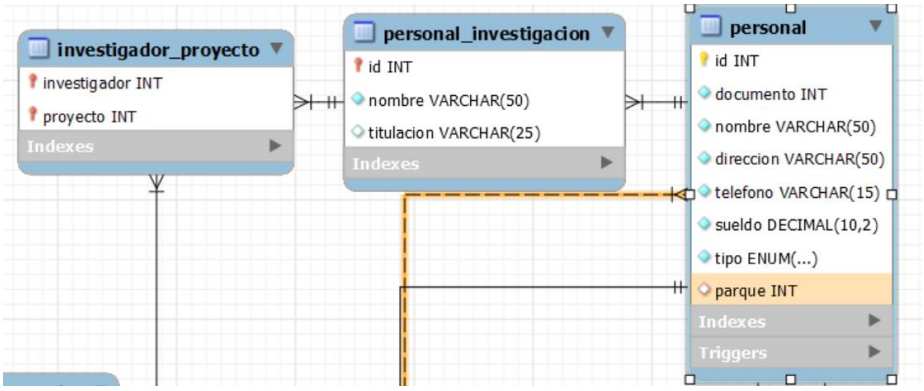
6. **personal_gestion** y **personal**:

   - La tabla **personal_gestion** está relacionada con la tabla **personal** mediante la clave foránea **id**, que referencia al identificador único de personal en **personal**. Esta relación permite distinguir y gestionar específicamente al personal de gestión dentro del conjunto general de personal asignado a los parques naturales.



7. **personal_investigacion** y **personal**, **proyectos_investigacion**:

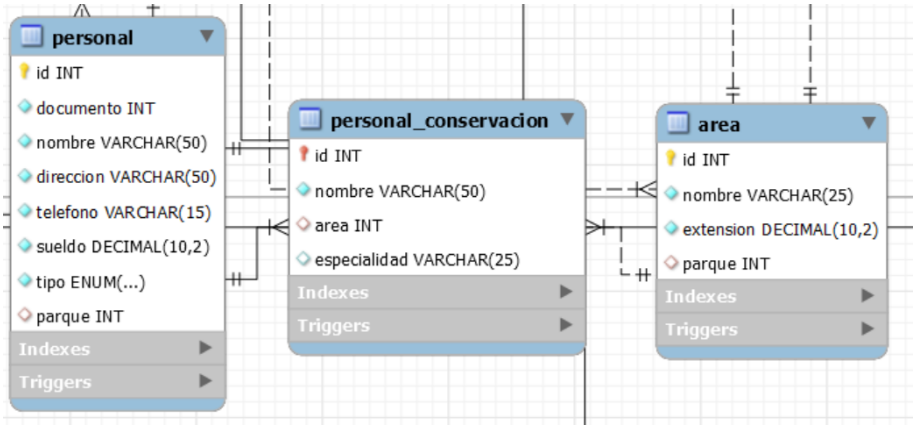
    - La tabla **personal_investigacion** está relacionada con la tabla **personal** mediante la clave foránea **id**, que referencia al identificador único de personal en **personal**. Además, la tabla **personal_investigacion** está relacionada con la tabla **proyectos_investigacion** mediante la clave foránea **proyecto**, que referencia al identificador único de proyectos en **proyectos_investigacion**. Estas relaciones permiten gestionar y asignar investigadores a proyectos específicos de investigación en los parques naturales.



8. **personal_conservacion** y **personal**, **area**:

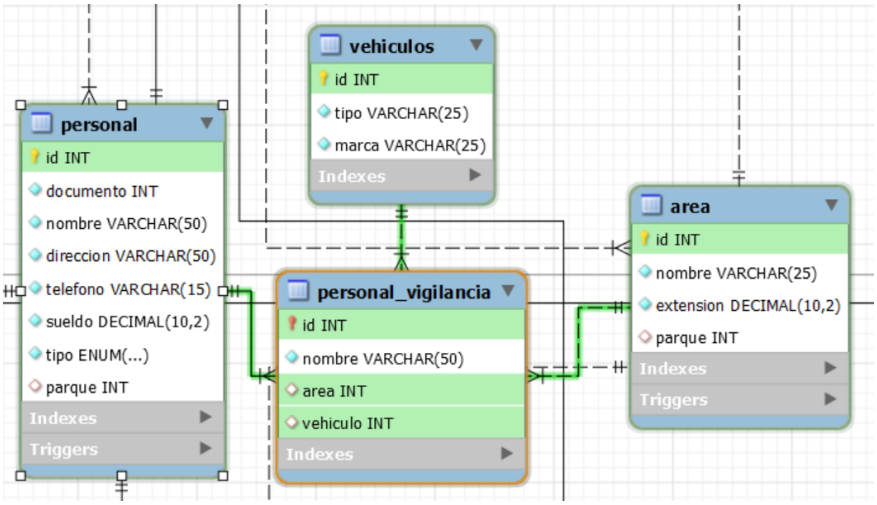
    - La tabla **personal_conservacion** está relacionada con la tabla **personal** mediante la clave foránea **id**, que referencia al identificador único de personal en **personal**. Además, la tabla **personal_conservacion** está relacionada con la tabla **area** mediante la clave foránea **area**, que referencia al identificador único de área en **area**. Estas relaciones permiten asignar personal de

conservación a áreas específicas dentro de los parques naturales según su especialidad.



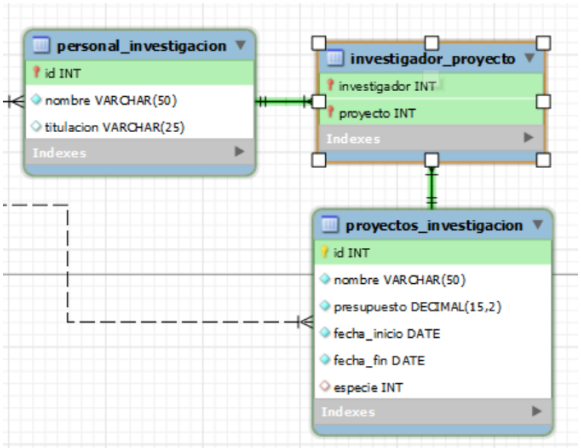
9. **personal_vigilancia** y **personal**, **area**, **vehiculos**:

   - La tabla **personal_vigilancia** está relacionada con la tabla **personal** mediante la clave foránea **id**, que referencia al identificador único de personal en **personal**. Además, la tabla **personal_vigilancia** está relacionada con las tablas **area** y **vehiculos** mediante las claves foráneas **area** y **vehiculo** respectivamente, que referencia al identificador único de área en **area** y al identificador único de vehículo en **vehiculos**. Estas relaciones permiten asignar personal de vigilancia a áreas específicas dentro de los parques naturales, gestionando su movilidad y responsabilidades dentro de un área determinada.



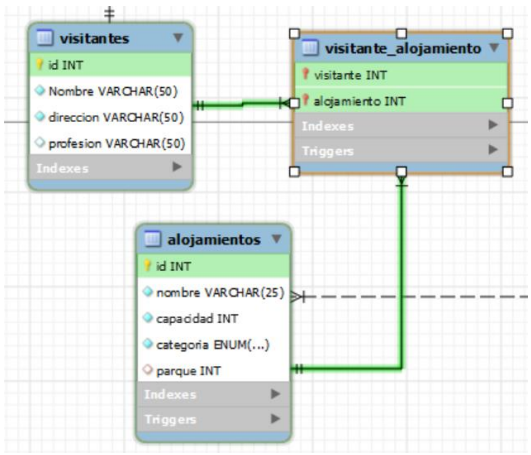
10. **investigador_proyecto** y **personal_investigacion**, **proyectos_investigacion**:

- La tabla **investigador_proyecto** establece una relación muchos a muchos entre las tablas **personal_investigacion** y **proyectos_investigacion** mediante las claves foráneas **investigador** y **proyecto** respectivamente. Esto permite asignar investigadores a múltiples proyectos de investigación y viceversa, facilitando la gestión y seguimiento de las actividades de investigación en los parques naturales.



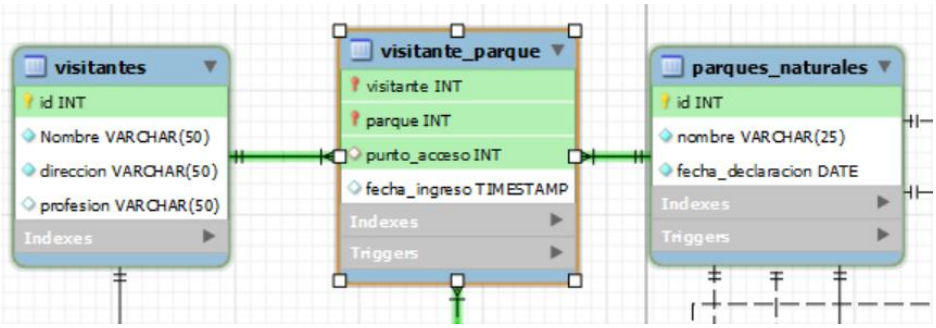
11. **visitante_alojamiento** y **visitantes**, **alojamientos**:

- La tabla **visitante_alojamiento** establece una relación muchos a muchos entre las tablas **visitantes** y **alojamientos** mediante las claves foráneas **visitante** y **alojamiento** respectivamente. Esto permite registrar qué visitantes se alojan en qué tipo de alojamiento durante su estadía en los parques naturales, gestionando así la ocupación y disponibilidad de los alojamientos.



12. **visitante_parque** y **visitantes**, **parques_naturales**:

    - La tabla **visitante_parque** establece una relación muchos a muchos entre las tablas **visitantes** y **parques_naturales** mediante las claves foráneas **visitante_id** y **parque_id** respectivamente. Esto permite registrar las visitas de los visitantes a diferentes parques naturales, gestionando la información relacionada con las actividades y movimientos de los visitantes en los parques.



Inserción de datos

Los inserts realizados en la base de datos ParqueNaturalesColombia comprenden una serie detallada de operaciones SQL que abarcan múltiples tablas relacionales. Estos inserts incluyen datos críticos como entidades responsables, departamentos, parques naturales, puntos de acceso, áreas específicas dentro del parque, alojamientos, especies animales, vegetales y minerales, vehículos, proyectos de investigación, visitantes, inventario de especies por área, y personal categorizado en gestión, vigilancia, conservación e investigación. Cada inserción está diseñada para simular datos reales que permiten validar el diseño y la funcionalidad de la base de datos en un contexto de gestión ambiental y conservación de recursos naturales.

```
-- ##############################
-- ########## Inserts ###########
-- ########## Tayrona ###########
-- ##############################
-- Entidad responsable
INSERT INTO entidad_responsable(nombre, direccion, telefono)
VALUES ('Alberto Mario Garzón Wilches', 'Calle 17 No. 1-C-78 Santa
Marta',313395031);
```

```sql
-- departamento
INSERT INTO departamentos(nombre, entidad_responsable)
VALUES ('Magdalena', 1);


-- Parque
INSERT INTO parques_naturales(nombre, fecha_declaracion)
VALUES
    ('Tayrona', '1988-11-17');

-- puntos de acceso del parque
INSERT INTO punto_acceso (parque, nombre) VALUES
(1, 'Zaino'),
(1, 'Calabazo'),
(1, 'Palangana'),
(1, 'Neguanje'),
(1, 'Cabo San Juan'),
(1, 'Playa Cristal');

-- departamento del parque
INSERT INTO departamento_parque(departamento, parque)
VALUES (1, 1);
-- areas del parque tayrona
INSERT INTO area(nombre, extension, parque)
VALUES
('SAN JUAN DEL GUÍA', 100.00, 1),
('LA PISCINA', 75.00, 1),
('ARENILLA', 50.00, 1),
('ARRECIFE', 120.00, 1),
('CAÑAVERAL', 90.00, 1),
('concha', 110.00, 1),
('CHENGUE', 80.00, 1),
('GAYRACA', 95.00, 1),
('NEGUANJE', 150.00, 1),
```

```
('7 OLAS', 65.00, 1),
('CINTO', 45.00, 1),
('GUACHAKYTA', 55.00, 1),
('PALMARITO', 30.00, 1),
('BRAVA', 40.00, 1),
('BOCA DEL SACO', 25.00, 1),
('CASTILLETE', 20.00, 1);

-- Alojamientos parque tayrona
INSERT INTO alojamientos(nombre, capacidad, categoria,parque)
VALUES
('Senda Koguiwa', 20, 'hotel',1),
('ECOHABS Tayrona Park', 30, 'hotel',1),
('CASA Tayrona Los Naranjos', 15, 'hotel',1),
('CABAÑA Barlovento', 10, 'cabaña',1),
('MALOKA Barlovento', 8, 'cabaña',1),
('SENDA Watapuy', 12, 'hotel',1);


-- especies del parque tayrona
    -- Inserción de datos para especies animales
    INSERT INTO especies(nombre_cientifico, nombre_vulgar, categoria) VALUES
        ('Choloepus hoffmanni', 'Perezoso de tres dedos', 'animal'),
        ('Choloepus didactylus', 'Perezoso de dos dedos', 'animal'),
        ('Leopardus tigrinus', 'Tigrillo', 'animal'),
        ('Leopardus pardalis', 'Ocelote', 'animal'),
        ('Puma concolor', 'Puma', 'animal'),
        ('Alouatta seniculus', 'Mono aullador', 'animal'),
        ('Saguinus oedipus', 'Mono tití', 'animal'),
        ('Eretmochelys imbricata', 'Tortuga carey', 'animal'),
        ('Chelonia mydas', 'Tortuga verde', 'animal'),
        ('Dermochelys coriacea', 'Tortuga laúd', 'animal'),
        ('Delfines', 'Delfines', 'animal'),
        ('Ballenas', 'Ballenas', 'animal'),
        ('Peces coralinos', 'Peces diversos', 'animal');
```

```sql
    -- Inserción de datos para especies vegetales
    INSERT INTO especies(nombre_cientifico, nombre_vulgar, categoria) VALUES
        ('Ceroxylon quindiuense', 'Palma de cera', 'vegetal'),
        ('Ceroxylon ceriferum', 'Palma de cera', 'vegetal'),
        ('Bromeliaceae', 'Bromelias', 'vegetal'),
        ('Orchidaceae', 'Orquídeas', 'vegetal'),
        ('Manglares', 'Manglares', 'vegetal'),
        ('Cactaceae', 'Cactus', 'vegetal'),
        ('Algas marinas', 'Algas marinas', 'vegetal'),
        ('Especies arbóreas diversas', 'Especies arbóreas diversas',
'vegetal');

    -- Inserción de datos para especies minerales
    INSERT INTO especies(nombre_cientifico, nombre_vulgar, categoria) VALUES
        ('Arenisca', 'Arenisca', 'mineral'),
        ('Granito', 'Granito', 'mineral'),
        ('Arcilla', 'Arcilla', 'mineral'),
        ('Cuarcita', 'Cuarcita', 'mineral'),
        ('Carbón', 'Carbón', 'mineral'),
        ('Rocas metamórficas', 'Rocas metamórficas', 'mineral'),
        ('Minerales de hierro', 'Minerales de hierro', 'mineral');



-- vehiculos
INSERT INTO vehiculos(tipo, marca)
VALUES
('Moto todo terreno', 'Honda'),
('Camioneta 4x4', 'Toyota'),
('Bicicleta de montaña', 'Specialized'),
('Embarcación', 'Yamaha');

-- proyectos de investigacion del parque tayrona
-- Proyectos de investigación del parque Tayrona con presupuestos en COP
```

```sql
INSERT INTO proyectos_investigacion (nombre, presupuesto, fecha_inicio, fecha_fin, especie) VALUES
    ('Conservación del Perezoso de tres dedos', 200000000, '2024-01-15', '2024-12-15', 1),
    ('Estudio de hábitos del Tigrillo', 300000000, '2024-02-01', '2024-11-01', 3),
    ('Reforestación con Palma de cera', 240000000, '2024-03-01', '2024-12-01', 14),
    ('Protección de la Tortuga carey', 340000000, '2024-04-01', '2025-03-01', 8),
    ('Investigación sobre Manglares', 360000000, '2024-05-01', '2025-04-01', 18),
    ('Monitoreo de Delfines en la costa', 280000000, '2024-06-01', '2025-05-01', 11),
    ('Preservación de Orquídeas', 220000000, '2024-07-01', '2025-06-01', 17),
    ('Estudio de impacto en Minerales de hierro', 380000000, '2024-08-01', '2025-07-01', 28),
    ('Conservación de Especies arbóreas diversas', 320000000, '2024-09-01', '2025-08-01', 21),
    ('Investigación sobre Rocas metamórficas', 272000000, '2024-10-01', '2025-09-01', 27);




-- visitantes
INSERT INTO visitantes (Nombre, direccion, profesion) VALUES
    ('Camila López', 'Carrera 7 # 72-54, Bogotá', 'Ingeniera'),
    ('Juan Ramírez', 'Calle 10 # 33-25, Medellín', 'Médico'),
    ('María González', 'Avenida 5 # 15-10, Cali', 'Profesora'),
    ('Carlos Rodríguez', 'Carrera 25 # 50-15, Barranquilla', 'Abogado'),
    ('Laura Martínez', 'Carrera 30 # 40-20, Bucaramanga', 'Arquitecta'),
    ('Santiago Gómez', 'Calle 50 # 12-30, Cartagena', 'Economista'),
    ('Valentina Sánchez', 'Avenida 3 # 8-12, Manizales', 'Estudiante'),
```

```
('Jorge Pérez', 'Carrera 15 # 22-35, Pereira', 'Artista'),
('Ana Ramírez', 'Calle 7 # 5-10, Armenia', 'Actor'),
('Diego Vargas', 'Carrera 2 # 4-5, Santa Marta', 'Empresario'),
('Isabella Castro', 'Avenida 10 # 25-30, Villavicencio', 'Diseñadora'),
('Sebastián Herrera', 'Carrera 12 # 15-18, Cúcuta', 'Programador'),
('Natalia Díaz', 'Calle 80 # 90-45, Ibague', 'Consultora'),
('Andrés Rojas', 'Carrera 45 # 35-25, Pasto', 'Investigador'),
('Carolina Gutiérrez', 'Avenida 20 # 10-5, Neiva', 'Escritora'),
('David Torres', 'Calle 22 # 18-20, Tunja', 'Científico'),
('Valeria Muñoz', 'Carrera 18 # 12-8, Popayán', 'Periodista'),
('Felipe Ortiz', 'Calle 33 # 40-22, Quibdó', 'Chef'),
('Diana Gómez', 'Avenida 7 # 6-5, Montería', 'Psicóloga'),
('Alejandro Varela', 'Carrera 12 # 15-10, Florencia', 'Piloto'),
('Paola Jiménez', 'Calle 10 # 8-5, Sincelejo', 'Bióloga'),
('Andrea Castañeda', 'Avenida 5 # 4-3, Leticia', 'Geólogo'),
('Manuel Ramírez', 'Carrera 8 # 10-12, Yopal', 'Traductora'),
('Laura Soto', 'Calle 15 # 20-25, Valledupar', 'Historiador'),
('Gabriel Montoya', 'Avenida 25 # 30-35, Mocoa', 'Marketing'),
('Daniela García', 'Carrera 40 # 50-60, Riohacha', 'Emprendedor'),
('Julian Zapata', 'Calle 7 # 6-5, Arauca', 'Influencer'),
('Sofía Rojas', 'Avenida 10 # 12-15, Quindío', 'Fotógrafo'),
('Luisa Pérez', 'Carrera 30 # 35-40, Tolima', 'Pintor'),
('Martín Silva', 'Calle 5 # 8-10, Caquetá', 'Ingeniero'),
('Valentina López', 'Avenida 6 # 10-12, Guaviare', 'Nutricionista'),
('Juan Carlos Ramírez', 'Carrera 20 # 25-30, Vaupés', 'Músico'),
('Valeria Guzmán', 'Calle 15 # 18-20, Amazonas', 'Político'),
('Esteban Ríos', 'Avenida 7 # 8-10, San Andrés', 'Empresaria'),
('Natalia Quintero', 'Carrera 12 # 15-20, Providencia', 'Financiero'),
('Miguel Gómez', 'Calle 25 # 30-35, Chocó', 'Consultora'),
('Ana María Ramírez', 'Avenida 20 # 25-30, Guainía', 'Director'),
('Carlos Rojas', 'Carrera 18 # 15-10, Casanare', 'Estilista'),
('Carolina García', 'Calle 30 # 35-40, Vichada', 'Gastrónomo');


-- inventario especie
```

```sql
-- Inserción de datos para especies animales
INSERT INTO inventario_especie(numero_individuos, area, especie)
VALUES
    (10, 1, 1),    -- Perezoso de tres dedos en SAN JUAN DEL GUÍA
    (15, 2, 2),    -- Perezoso de dos dedos en LA PISCINA
    (20, 3, 3),    -- Tigrillo en ARENILLA
    (25, 4, 4),    -- Ocelote en ARRECIFE
    (30, 5, 5),    -- Puma en CAÑAVERAL
    (5, 6, 6),     -- Mono aullador en concha
    (8, 7, 7),     -- Mono tití en CHENGUE
    (3, 8, 8),     -- Tortuga carey en GAYRACA
    (12, 9, 9),    -- Tortuga verde en NEGUANJE
    (7, 10, 10),   -- Tortuga laúd en 7 OLAS
    (50, 11, 11),  -- Delfines en CINTO
    (40, 12, 12),  -- Ballenas en GUACHAKYTA
    (100, 13, 13); -- Peces diversos en PALMARITO

-- Inserción de datos para especies vegetales
INSERT INTO inventario_especie(numero_individuos, area, especie)
VALUES
    (200, 1, 14),   -- Palma de cera en SAN JUAN DEL GUÍA
    (150, 2, 14),   -- Palma de cera en LA PISCINA
    (100, 3, 15),   -- Bromelias en ARENILLA
    (80, 4, 16),    -- Orquídeas en ARRECIFE
    (120, 5, 17),   -- Manglares en CAÑAVERAL
    (300, 6, 18),   -- Cactus en concha
    (250, 7, 19),   -- Algas marinas en CHENGUE
    (180, 8, 20);   -- Especies arbóreas diversas en GAYRACA

-- Inserción de datos para especies minerales
INSERT INTO inventario_especie(numero_individuos, area, especie)
VALUES
    (500, 9, 21),   -- Arenisca en NEGUANJE
    (300, 10, 22),  -- Granito en 7 OLAS
    (400, 11, 23),  -- Arcilla en CINTO
```

```
        (150, 12, 24),  -- Cuarcita en GUACHAKYTA
        (200, 13, 25),  -- Carbón en PALMARITO
        (100, 14, 26),  -- Rocas metamórficas en BRAVA
        (80, 15, 27);   -- Minerales de hierro en BOCA DEL SACO

-- personal
-- Insertar 4 registros de Personal de Gestión (tipo '001')
INSERT INTO personal (documento, nombre, direccion, telefono, sueldo, tipo,
parque)
VALUES
    (123456789, 'Carlos Sánchez', 'Carrera 5 #10-15, Santa Marta',
'3012345678', 4000000.00, 'Personal de Gestión', 1),
    (234567890, 'Ana Gómez', 'Calle 23 #8-12, Santa Marta', '3056789012',
4100000.00, 'Personal de Gestión', 1),
    (345678901, 'Pedro Rodríguez', 'Avenida Libertador #15-30, Santa Marta',
'3001234567', 4200000.00, 'Personal de Gestión', 1),
    (456789012, 'María Pérez', 'Carrera 7 #12-25, Santa Marta',
'3045678901', 4300000.00, 'Personal de Gestión', 1);

-- Insertar 20 registros de Personal de Vigilancia (tipo '002')
INSERT INTO personal (documento, nombre, direccion, telefono, sueldo, tipo,
parque)
VALUES
    (567890123, 'Luis Martínez', 'Calle 45 #20-10, Santa Marta',
'3012345678', 3500000.00, 'Personal de Vigilancia', 1),
    (678901234, 'Juan López', 'Avenida Circunvalar #30-40, Santa Marta',
'3056789012', 3600000.00, 'Personal de Vigilancia', 1),
    (789012345, 'Elena Ramírez', 'Carrera 15 #5-25, Santa Marta',
'3001234567', 3700000.00, 'Personal de Vigilancia', 1),
    (890123456, 'Gabriel Herrera', 'Calle 60 #18-12, Santa Marta',
'3045678901', 3800000.00, 'Personal de Vigilancia', 1),
    (901234567, 'Sofía González', 'Carrera 10 #22-30, Santa Marta',
'3012345678', 3900000.00, 'Personal de Vigilancia', 1),
    (123456780, 'Diego Castro', 'Calle 70 #35-15, Santa Marta',
'3056789012', 4000000.00, 'Personal de Vigilancia', 1),
```

```
    (234567891, 'Laura Díaz', 'Avenida del Río #40-50, Santa Marta',
'3001234567', 4100000.00, 'Personal de Vigilancia', 1),
    (345678902, 'Andrés Montoya', 'Carrera 25 #12-18, Santa Marta',
'3045678901', 4200000.00, 'Personal de Vigilancia', 1),
    (456789013, 'Marta Jiménez', 'Calle 80 #28-30, Santa Marta',
'3012345678', 4300000.00, 'Personal de Vigilancia', 1),
    (567890124, 'Javier Vargas', 'Carrera 30 #15-20, Santa Marta',
'3056789012', 4400000.00, 'Personal de Vigilancia', 1),
    (678901235, 'Camila Restrepo', 'Avenida del Mar #50-60, Santa Marta',
'3001234567', 4500000.00, 'Personal de Vigilancia', 1),
    (789012346, 'Felipe Soto', 'Calle 90 #38-42, Santa Marta', '3045678901',
4600000.00, 'Personal de Vigilancia', 1),
    (890123457, 'Valentina Sierra', 'Carrera 35 #22-28, Santa Marta',
'3012345678', 4700000.00, 'Personal de Vigilancia', 1),
    (901234568, 'Daniel Cruz', 'Calle 100 #45-50, Santa Marta',
'3056789012', 4800000.00, 'Personal de Vigilancia', 1),
    (123456781, 'Juliana Medina', 'Avenida Libertadores #60-70, Santa
Marta', '3001234567', 4900000.00, 'Personal de Vigilancia', 1),
    (234567892, 'Roberto Torres', 'Carrera 40 #18-22, Santa Marta',
'3045678901', 5000000.00, 'Personal de Vigilancia', 1),
    (345678903, 'Carolina Duarte', 'Calle 110 #48-52, Santa Marta',
'3012345678', 5100000.00, 'Personal de Vigilancia', 1),
    (456789014, 'Fernando Varela', 'Carrera 45 #25-30, Santa Marta',
'3056789012', 5200000.00, 'Personal de Vigilancia', 1),
    (567890125, 'Natalia Rojas', 'Calle 120 #58-62, Santa Marta',
'3001234567', 5300000.00, 'Personal de Vigilancia', 1),
    (678901236, 'Ricardo Gómez', 'Avenida del Bosque #70-80, Santa Marta',
'3045678901', 5400000.00, 'Personal de Vigilancia', 1);

-- Insertar 10 registros de Personal de Conservación (tipo '003')
INSERT INTO personal (documento, nombre, direccion, telefono, sueldo, tipo,
parque)
VALUES
    (789012348, 'Antonio Herrera', 'Calle 50 #30-10, Santa Marta',
'3012345678', 4200000.00, 'Personal de Conservación', 1),
```

```
    (890123459, 'Isabel Cruz', 'Carrera 20 #10-5, Santa Marta',
'3056789012', 4300000.00, 'Personal de Conservación', 1),
    (901234560, 'Jorge Sánchez', 'Avenida Sierra Nevada #15-20, Santa
Marta', '3001234567', 4400000.00, 'Personal de Conservación', 1),
    (123456783, 'Ana María Giraldo', 'Calle 55 #25-30, Santa Marta',
'3045678901', 4500000.00, 'Personal de Conservación', 1),
    (234567894, 'Gustavo Ríos', 'Carrera 22 #12-18, Santa Marta',
'3012345678', 4600000.00, 'Personal de Conservación', 1),
    (345678905, 'Mariana Ramírez', 'Avenida del Prado #30-35, Santa Marta',
'3056789012', 4700000.00, 'Personal de Conservación', 1),
    (456789016, 'Sebastián López', 'Calle 60 #28-32, Santa Marta',
'3001234567', 4800000.00, 'Personal de Conservación', 1),
    (567890127, 'Valeria Gutiérrez', 'Carrera 25 #15-20, Santa Marta',
'3045678901', 4900000.00, 'Personal de Conservación', 1),
    (678901238, 'Rafaela Castro', 'Avenida del Mar #40-45, Santa Marta',
'3012345678', 5000000.00, 'Personal de Conservación', 1),
    (789012349, 'Martín Duque', 'Calle 70 #35-40, Santa Marta',
'3056789012', 5100000.00, 'Personal de Conservación', 1);

-- Insertar 10 registros de Personal Investigador (tipo '004')
INSERT INTO personal (documento, nombre, direccion, telefono, sueldo, tipo,
parque)
VALUES
    (890123461, 'Luisa Martínez', 'Carrera 15 #5-10, Santa Marta',
'3001234567', 6000000.00, 'Personal Investigador', 1),
    (123456784, 'Andrés Herrera', 'Calle 25 #10-15, Santa Marta',
'3045678901', 6100000.00, 'Personal Investigador', 1),
    (234567895, 'Valentín Gómez', 'Avenida Libertadores #20-25, Santa
Marta', '3012345678', 6200000.00, 'Personal Investigador', 1),
    (345678906, 'Camilo Ramírez', 'Carrera 30 #15-20, Santa Marta',
'3056789012', 6300000.00, 'Personal Investigador', 1),
    (456789017, 'Carolina López', 'Calle 45 #18-22, Santa Marta',
'3001234567', 6400000.00, 'Personal Investigador', 1),
    (567890128, 'Santiago Castro', 'Avenida del Río #25-30, Santa Marta',
'3045678901', 6500000.00, 'Personal Investigador', 1),
```

```
    (678901239, 'Daniela Duarte', 'Carrera 40 #22-25, Santa Marta',
'3012345678', 6600000.00, 'Personal Investigador', 1),
    (789012350, 'Roberto Gutiérrez', 'Calle 55 #30-35, Santa Marta',
'3056789012', 6700000.00, 'Personal Investigador', 1),
    (890123462, 'Laura Ramírez', 'Avenida del Bosque #35-40, Santa Marta',
'3001234567', 6800000.00, 'Personal Investigador', 1),
    (123456785, 'Juan Pablo Martínez', 'Calle 65 #28-32, Santa Marta',
'3045678901', 6900000.00, 'Personal Investigador', 1);

-- investigador - proyecto
INSERT INTO investigador_proyecto (investigador, proyecto) VALUES
    (39, 1), -- Luisa Martínez en Conservación del Perezoso de tres dedos
    (39, 2), -- Luisa Martínez en Estudio de hábitos del Tigrillo
    (40, 3), -- Andrés Herrera en Reforestación con Palma de cera
    (40, 4), -- Andrés Herrera en Protección de la Tortuga carey
    (41, 5), -- Valentín Gómez en Investigación sobre Manglares
    (41, 6), -- Valentín Gómez en Monitoreo de Delfines en la costa
    (42, 7), -- Camilo Ramírez en Preservación de Orquídeas
    (42, 8), -- Camilo Ramírez en Estudio de impacto en Minerales de hierro
    (43, 9), -- Carolina López en Conservación de Especies arbóreas diversas
    (43, 10), -- Carolina López en Investigación sobre Rocas metamórficas
    (44, 1), -- Santiago Castro en Conservación del Perezoso de tres dedos
    (44, 3), -- Santiago Castro en Reforestación con Palma de cera
    (39, 5), -- Luisa Martínez en Investigación sobre Manglares
    (40, 7), -- Andrés Herrera en Preservación de Orquídeas
    (41, 9), -- Valentín Gómez en Conservación de Especies arbóreas diversas
    (42, 2), -- Camilo Ramírez en Estudio de hábitos del Tigrillo
    (43, 4), -- Carolina López en Protección de la Tortuga carey
    (44, 6), -- Santiago Castro en Monitoreo de Delfines en la costa
    (39, 8), -- Luisa Martínez en Estudio de impacto en Minerales de hierro
    (40, 10); -- Andrés Herrera en Investigación sobre Rocas metamórficas


-- visitantes parque
```

```sql
-- Insertar registros en la tabla visitante_parque en desorden y sin usar todos los visitantes
INSERT INTO visitante_parque (visitante, parque, punto_acceso) VALUES
(1, 1, 3),  -- Camila López visitó Palangana
(5, 1, 2),  -- Laura Martínez visitó Calabazo
(7, 1, 4),  -- Valentina Sánchez visitó Neguanje
(10, 1, 1), -- Diego Vargas visitó Zaino
(12, 1, 6), -- Sebastián Herrera visitó Playa Cristal
(15, 1, 5), -- Carolina Gutiérrez visitó Cabo San Juan
(18, 1, 2), -- Diana Gómez visitó Calabazo
(21, 1, 3), -- Andrea Castañeda visitó Palangana
(25, 1, 1), -- Gabriel Montoya visitó Zaino
(28, 1, 5); -- Julian Zapata visitó Cabo San Juan

-- visitante - alojamiento
-- Insertando relaciones entre visitantes y alojamientos
INSERT INTO visitante_alojamiento (visitante, alojamiento) VALUES
(1, 1),   -- Camila López se aloja en Senda Koguiwa
(5, 2),   -- Laura Martínez se aloja en ECOHABS Tayrona Park
(7, 3),   -- Valentina Sánchez se aloja en CASA Tayrona Los Naranjos
(10, 4),  -- Diego Vargas se aloja en CABAÑA Barlovento
(12, 5),  -- Sebastián Herrera se aloja en MALOKA Barlovento
(15, 6),  -- Carolina Gutiérrez se aloja en SENDA Watapuy
(18, 1),  -- Diana Gómez se aloja en Senda Koguiwa
(21, 2),  -- Andrea Castañeda se aloja en ECOHABS Tayrona Park
(25, 3),  -- Gabriel Montoya se aloja en CASA Tayrona Los Naranjos
(28, 4);  -- Julian Zapata se aloja en CABAÑA Barlovento

-- Actualizar los datos existentes en la tabla personal_conservacion con áreas y especialidades asignadas aleatoriamente
UPDATE personal_conservacion SET area = 5, especialidad = 'Botánica' WHERE id = 25;
UPDATE personal_conservacion SET area = 3, especialidad = 'Zoología' WHERE id = 26;
```

```sql
UPDATE personal_conservacion SET area = 8, especialidad = 'Ecología' WHERE id = 27;
UPDATE personal_conservacion SET area = 12, especialidad = 'Geología' WHERE id = 28;
UPDATE personal_conservacion SET area = 2, especialidad = 'Hidrología' WHERE id = 29;
UPDATE personal_conservacion SET area = 7, especialidad = 'Microbiología' WHERE id = 30;
UPDATE personal_conservacion SET area = 6, especialidad = 'Ornitología' WHERE id = 31;
UPDATE personal_conservacion SET area = 1, especialidad = 'Entomología' WHERE id = 32;
UPDATE personal_conservacion SET area = 4, especialidad = 'Climatología' WHERE id = 33;
UPDATE personal_conservacion SET area = 9, especialidad = 'Biología Marina' WHERE id = 34;



-- Actualizar los datos existentes en la tabla personal_gestion con punto de acceso
UPDATE personal_gestion SET punto_acceso = 1 WHERE id = 1;
UPDATE personal_gestion SET punto_acceso = 2 WHERE id = 2;
UPDATE personal_gestion SET punto_acceso = 3 WHERE id = 3;
UPDATE personal_gestion SET punto_acceso = 4 WHERE id = 4;

-- Agregar titulacion a los registros especificados
UPDATE personal_investigacion SET titulacion = 'Biología' WHERE id = 35;
UPDATE personal_investigacion SET titulacion = 'Ecología' WHERE id = 36;
UPDATE personal_investigacion SET titulacion = 'Gestión Ambiental' WHERE id = 37;
UPDATE personal_investigacion SET titulacion = 'Zoología' WHERE id = 38;
UPDATE personal_investigacion SET titulacion = 'Botánica' WHERE id = 39;
UPDATE personal_investigacion SET titulacion = 'Biología Marina' WHERE id = 40;
```

```sql
UPDATE personal_investigacion SET titulacion = 'Geología' WHERE id = 41;
UPDATE personal_investigacion SET titulacion = 'Conservación' WHERE id = 42;
UPDATE personal_investigacion SET titulacion = 'Ciencias Ambientales' WHERE
id = 43;
UPDATE personal_investigacion SET titulacion = 'Ingeniería Forestal' WHERE
id = 44;

-- Asignación aleatoria de área y vehículo al personal de vigilancia

UPDATE personal_vigilancia SET area = 8, vehiculo = 3 WHERE id = 5;
UPDATE personal_vigilancia SET area = 4, vehiculo = 4 WHERE id = 6;
UPDATE personal_vigilancia SET area = 14, vehiculo = 2 WHERE id = 7;
UPDATE personal_vigilancia SET area = 6, vehiculo = 1 WHERE id = 8;
UPDATE personal_vigilancia SET area = 11, vehiculo = 3 WHERE id = 9;
UPDATE personal_vigilancia SET area = 9, vehiculo = 2 WHERE id = 10;
UPDATE personal_vigilancia SET area = 12, vehiculo = 4 WHERE id = 11;
UPDATE personal_vigilancia SET area = 3, vehiculo = 1 WHERE id = 12;
UPDATE personal_vigilancia SET area = 5, vehiculo = 3 WHERE id = 13;
UPDATE personal_vigilancia SET area = 7, vehiculo = 4 WHERE id = 14;
UPDATE personal_vigilancia SET area = 16, vehiculo = 2 WHERE id = 15;
UPDATE personal_vigilancia SET area = 1, vehiculo = 1 WHERE id = 16;
UPDATE personal_vigilancia SET area = 15, vehiculo = 3 WHERE id = 17;
UPDATE personal_vigilancia SET area = 2, vehiculo = 4 WHERE id = 18;
UPDATE personal_vigilancia SET area = 10, vehiculo = 1 WHERE id = 19;
UPDATE personal_vigilancia SET area = 13, vehiculo = 2 WHERE id = 20;
UPDATE personal_vigilancia SET area = 8, vehiculo = 3 WHERE id = 21;
UPDATE personal_vigilancia SET area = 4, vehiculo = 4 WHERE id = 22;
UPDATE personal_vigilancia SET area = 14, vehiculo = 1 WHERE id = 23;
UPDATE personal_vigilancia SET area = 6, vehiculo = 2 WHERE id = 24;
```

Funciones y procedimientos

Procedimientos

Agregar o eliminar un departamento sin especificar ID

A. Descripción

Este procedimiento permite agregar o eliminar un departamento en la base de datos ParquesNaturalesColombia sin necesidad de especificar el ID del departamento. Dependiendo de la opción seleccionada ('agregar' o 'eliminar'), el procedimiento realizará una inserción o eliminación en la tabla **departamentos** usando el nombre del departamento y el ID de la entidad responsable.

B. Sintaxis

```
CREATE PROCEDURE agregar_eliminar_departamento(
    IN opcion VARCHAR(10),  -- 'agregar' o 'eliminar'
    IN nombre_departamento VARCHAR(50),
    IN id_entidad_responsable INT
)
BEGIN
    IF opcion = 'agregar' THEN
        INSERT INTO departamentos (nombre, entidad_responsable)
        VALUES (nombre_departamento, id_entidad_responsable);
    ELSEIF opcion = 'eliminar' THEN
        DELETE FROM departamentos WHERE nombre = nombre_departamento AND
entidad_responsable = id_entidad_responsable;
    ELSE
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Opción no válida';
    END IF;
END
```

C. Parámetros

- **opcion**: Un valor de tipo VARCHAR que indica la acción a realizar. Puede ser 'agregar' o 'eliminar'.
- **nombre_departamento**: El nombre del departamento a agregar o eliminar.
- **id_entidad_responsable**: El ID de la entidad responsable asociada al departamento.

D. Salida/Retorno

El procedimiento no retorna ningún valor. Realiza una inserción o eliminación en la tabla **departamentos** o lanza un error si la opción proporcionada no es válida.

E. Ejemplo de Implementación

Supongamos que queremos agregar un nuevo departamento llamado "Gestión Ambiental" con un ID de entidad responsable de 3. Luego, queremos eliminar el departamento "Conservación" que también tiene el mismo ID de entidad responsable.

```
-- Agregar un nuevo departamento
CALL agregar_eliminar_departamento('agregar', 'Gestión Ambiental', 3);

-- Eliminar un departamento existente
CALL agregar_eliminar_departamento('eliminar', 'Conservación', 3);
```

En estos ejemplos, el primer llamado al procedimiento agrega el departamento "Gestión Ambiental" con el ID de entidad responsable 3, mientras que el segundo llamado elimina el departamento "Conservación" con el mismo ID de entidad responsable.

Agregar, eliminar o modificar una entidad responsable sin especificar ID

A. Descripción

Este procedimiento permite agregar, eliminar o modificar una entidad responsable en la base de datos ParquesNaturalesColombia sin necesidad de especificar el ID de la entidad. Dependiendo de la opción seleccionada ('agregar', 'eliminar' o 'modificar'), el procedimiento realizará una inserción, eliminación o modificación en la tabla **entidad_responsable** usando el nombre de la entidad y sus detalles.

B. Sintaxis

```
CREATE PROCEDURE agregar_eliminar_modificar_entidad_responsable(
    IN opcion VARCHAR(15),  -- 'agregar', 'eliminar' o 'modificar'
    IN nombre_entidad VARCHAR(50),
```

```
    IN direccion_entidad VARCHAR(50),
    IN telefono_entidad INT
)
BEGIN
    IF opcion = 'agregar' THEN
        INSERT INTO entidad_responsable (nombre, direccion, telefono)
        VALUES (nombre_entidad, direccion_entidad, telefono_entidad);
    ELSEIF opcion = 'eliminar' THEN
        DELETE FROM entidad_responsable WHERE nombre = nombre_entidad LIMIT
1;
    ELSEIF opcion = 'modificar' THEN
        UPDATE entidad_responsable
        SET direccion = direccion_entidad, telefono = telefono_entidad
        WHERE nombre = nombre_entidad LIMIT 1;
    ELSE
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Opción no válida';
    END IF;
END
```

c. Parámetros

- **opcion**: Un valor de tipo VARCHAR que indica la acción a realizar. Puede ser 'agregar', 'eliminar' o 'modificar'.
- **nombre_entidad**: El nombre de la entidad responsable a agregar, eliminar o modificar.
- **direccion_entidad**: La dirección de la entidad responsable.
- **telefono_entidad**: El número de teléfono de la entidad responsable.

d. Salida/Retorno

El procedimiento no retorna ningún valor. Realiza una inserción, eliminación o modificación en la tabla **entidad_responsable** o lanza un error si la opción proporcionada no es válida.

e. Ejemplo de Implementación

Supongamos que queremos agregar una nueva entidad responsable llamada "Fundación Ecológica" con una dirección "Calle Verde 123" y un teléfono "987654321". Luego, queremos modificar la misma entidad para cambiar su dirección a "Avenida Verde 456" y finalmente eliminarla.

```
-- Agregar una nueva entidad responsable
CALL agregar_eliminar_modificar_entidad_responsable('agregar', 'Fundación
Ecológica', 'Calle Verde 123', 987654321);

-- Modificar la entidad responsable existente
CALL agregar_eliminar_modificar_entidad_responsable('modificar', 'Fundación
Ecológica', 'Avenida Verde 456', 987654321);

-- Eliminar la entidad responsable existente
CALL agregar_eliminar_modificar_entidad_responsable('eliminar', 'Fundación
Ecológica', '', 0);
```

En estos ejemplos, el primer llamado al procedimiento agrega la entidad "Fundación Ecológica" con su dirección y teléfono, el segundo llamado modifica la dirección de la entidad a "Avenida Verde 456", y el tercer llamado elimina la entidad "Fundación Ecológica" de la base de datos.

agregar, eliminar o modificar un parque sin especificar ID

a. Descripción

Este procedimiento permite agregar, eliminar o modificar un parque en la base de datos ParquesNaturalesColombia sin necesidad de especificar el ID del parque. Dependiendo de la opción seleccionada ('agregar', 'eliminar' o 'modificar'), el procedimiento realizará una inserción, eliminación o modificación en la tabla **parques_naturales** usando el nombre del parque y su fecha de declaración.

b. Sintaxis

```
agregar, eliminar o modificar parque sin especificar ID
DELIMITER //

CREATE PROCEDURE agregar_eliminar_modificar_parque(
    IN opcion VARCHAR(15),  -- 'agregar', 'eliminar' o 'modificar'
    IN nombre_parque VARCHAR(50),
    IN fecha_declaracion DATE
)
BEGIN
    IF opcion = 'agregar' THEN
```

```
        INSERT INTO parques_naturales (nombre, fecha_declaracion)
        VALUES (nombre_parque, fecha_declaracion);
    ELSEIF opcion = 'eliminar' THEN
        DELETE FROM parques_naturales WHERE nombre = nombre_parque LIMIT 1;
    ELSEIF opcion = 'modificar' THEN
        UPDATE parques_naturales
        SET fecha_declaracion = fecha_declaracion
        WHERE nombre = nombre_parque LIMIT 1;
    ELSE
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Opción no válida';
    END IF;
END //
```

c. Parámetros

- **opcion**: Un valor de tipo VARCHAR que indica la acción a realizar. Puede ser 'agregar', 'eliminar' o 'modificar'.
- **nombre_parque**: El nombre del parque natural a agregar, eliminar o modificar.
- **fecha_declaracion**: La fecha de declaración del parque natural.

d. Salida/Retorno

El procedimiento no retorna ningún valor. Realiza una inserción, eliminación o modificación en la tabla **parques_naturales** o lanza un error si la opción proporcionada no es válida.

e. Ejemplo de Implementación

Supongamos que queremos agregar un nuevo parque llamado "Parque Nacional de la Sierra" con una fecha de declaración "2022-05-15". Luego, queremos modificar el mismo parque para cambiar su fecha de declaración a "2023-06-20" y finalmente eliminarlo.

```
-- Agregar un nuevo parque
CALL agregar_eliminar_modificar_parque('agregar', 'Parque Nacional de la
Sierra', '2022-05-15');

-- Modificar el parque existente
CALL agregar_eliminar_modificar_parque('modificar', 'Parque Nacional de la
Sierra', '2023-06-20');

-- Eliminar el parque existente
```

```
CALL agregar_eliminar_modificar_parque('eliminar', 'Parque Nacional de la
Sierra', '2023-06-20');
```

En estos ejemplos, el primer llamado al procedimiento agrega el parque "Parque Nacional de la Sierra" con su fecha de declaración, el segundo llamado modifica la fecha de declaración del parque a "2023-06-20", y el tercer llamado elimina el parque "Parque Nacional de la Sierra" de la base de datos.

Procedimientos para Agregar, Eliminar y Modificar Alojamiento

a. Descripción

Estos procedimientos permiten agregar, eliminar y modificar registros en la tabla **alojamientos**.

- El procedimiento agregar_alojamiento añade un nuevo alojamiento a la base de datos.
- El procedimiento eliminar_alojamiento elimina un alojamiento existente por su ID.
- El procedimiento modificar_alojamiento actualiza los detalles de un alojamiento existente.

b. Sintaxis

```
DELIMITER //

-- Procedimiento para agregar un alojamiento
CREATE PROCEDURE agregar_alojamiento(
    IN p_nombre VARCHAR(25),
    IN p_capacidad INT,
    IN p_categoria ENUM('hotel', 'cabaña'),
    IN p_parque INT
)
BEGIN
    INSERT INTO alojamientos (nombre, capacidad, categoria, parque)
    VALUES (p_nombre, p_capacidad, p_categoria, p_parque);
END //

-- Procedimiento para eliminar un alojamiento
CREATE PROCEDURE eliminar_alojamiento(
    IN p_id INT
```

```
)
BEGIN
    DELETE FROM alojamientos WHERE id = p_id;
END //

-- Procedimiento para modificar un alojamiento
CREATE PROCEDURE modificar_alojamiento(
    IN p_id INT,
    IN p_nombre VARCHAR(25),
    IN p_capacidad INT,
    IN p_categoria ENUM('hotel', 'cabaña'),
    IN p_parque INT
)
BEGIN
    UPDATE alojamientos
    SET nombre = p_nombre, capacidad = p_capacidad, categoria = p_categoria,
parque = p_parque
    WHERE id = p_id;
END //
```

c. Parámetros

Agregar Alojamiento

- **p_nombre**: El nombre del alojamiento.
- **p_capacidad**: La capacidad del alojamiento.
- **p_categoria**: La categoría del alojamiento ('hotel' o 'cabaña').
- **p_parque**: El ID del parque al que pertenece el alojamiento.

Eliminar Alojamiento

- **p_id**: El ID del alojamiento a eliminar.

Modificar Alojamiento

- **p_id**: El ID del alojamiento a modificar.
- **p_nombre**: El nuevo nombre del alojamiento.
- **p_capacidad**: La nueva capacidad del alojamiento.

- **p_categoria**: La nueva categoría del alojamiento ('hotel' o 'cabaña').
- **p_parque**: El nuevo ID del parque al que pertenece el alojamiento.

d. Salida/Retorno

Ninguno de estos procedimientos retorna un valor. Cada uno realiza una acción específica en la tabla **alojamientos**:

- agregar_alojamiento inserta un nuevo registro.
- eliminar_alojamiento elimina un registro existente.
- modificar_alojamiento actualiza un registro existente.

e. Ejemplo de Implementación

Agregar Alojamiento

En este ejemplo, se agrega un nuevo alojamiento llamado "Cabaña del Sol" con capacidad para 4 personas y categoría "cabaña" en el parque con ID 1.

Eliminar Alojamiento

En este ejemplo, se elimina el alojamiento con ID 1.

Modificar Alojamiento

En este ejemplo, se modifica el alojamiento con ID 1 para tener el nombre "Hotel de la Montaña", capacidad para 10 personas, categoría "hotel" y pertenecer al parque con ID 2.

Procedimientos para Agregar, Eliminar y Modificar Punto de Acceso

a. Descripción

Estos procedimientos permiten agregar, eliminar y modificar registros en la tabla **punto_acceso**.

- El procedimiento agregar_punto_acceso añade un nuevo punto de acceso a la base de datos.

- El procedimiento eliminar_punto_acceso elimina un punto de acceso existente por su ID.
- El procedimiento modificar_punto_acceso actualiza los detalles de un punto de acceso existente.

b. Sintaxis

```
-- Procedimiento para agregar un punto de acceso
CREATE PROCEDURE agregar_punto_acceso(
    IN p_parque INT,
    IN p_nombre VARCHAR(50)
)
BEGIN
    INSERT INTO punto_acceso (parque, nombre)
    VALUES (p_parque, p_nombre);
END //

-- Procedimiento para eliminar un punto de acceso
CREATE PROCEDURE eliminar_punto_acceso(
    IN p_id INT
)
BEGIN
    DELETE FROM punto_acceso WHERE id = p_id;
END //

-- Procedimiento para modificar un punto de acceso
CREATE PROCEDURE modificar_punto_acceso(
    IN p_id INT,
    IN p_parque INT,
    IN p_nombre VARCHAR(50)
)
BEGIN
    UPDATE punto_acceso
    SET parque = p_parque, nombre = p_nombre
    WHERE id = p_id;
END //
```

c. Parámetros

Agregar Punto de Acceso

- **p_parque**: El ID del parque al que pertenece el punto de acceso.

- **p_nombre**: El nombre del punto de acceso.

Eliminar Punto de Acceso

- **p_id**: El ID del punto de acceso a eliminar.

Modificar Punto de Acceso

- **p_id**: El ID del punto de acceso a modificar.
- **p_parque**: El nuevo ID del parque al que pertenece el punto de acceso.
- **p_nombre**: El nuevo nombre del punto de acceso.

d. Salida/Retorno

Ninguno de estos procedimientos retorna un valor. Cada uno realiza una acción específica en la tabla **punto_acceso**:

- agregar_punto_acceso inserta un nuevo registro.
- eliminar_punto_acceso elimina un registro existente.
- modificar_punto_acceso actualiza un registro existente.

e. Ejemplo de Implementación

Agregar Punto de Acceso

En este ejemplo, se agrega un nuevo punto de acceso llamado "Entrada Norte" en el parque con ID 1.

Eliminar Punto de Acceso

CALL eliminar_punto_acceso(1);

En este ejemplo, se elimina el punto de acceso con ID 1.

Modificar Punto de Acceso

CALL modificar_punto_acceso(1, 2, 'Entrada Sur');

En este ejemplo, se modifica el punto de acceso con ID 1 para tener el nombre "Entrada Sur" y pertenecer al parque con ID 2.

Procedimientos para Agregar, Eliminar y Modificar Área

a. Descripción

Estos procedimientos permiten agregar, eliminar y modificar registros en la tabla **area**.

- El procedimiento agregar_area añade una nueva área a la base de datos.
- El procedimiento eliminar_area elimina una área existente por su ID.
- El procedimiento modificar_area actualiza los detalles de una área existente.

b. Sintaxis

```
-- Procedimiento para agregar un área
CREATE PROCEDURE agregar_area(
    IN p_nombre VARCHAR(25),
    IN p_extension DECIMAL(10,2),
    IN p_parque INT
)
BEGIN
    INSERT INTO area (nombre, extension, parque)
    VALUES (p_nombre, p_extension, p_parque);
END //

-- Procedimiento para eliminar un área
CREATE PROCEDURE eliminar_area(
    IN p_id INT
)
BEGIN
    DELETE FROM area WHERE id = p_id;
END //

-- Procedimiento para modificar un área
CREATE PROCEDURE modificar_area(
    IN p_id INT,
    IN p_nombre VARCHAR(25),
    IN p_extension DECIMAL(10,2),
    IN p_parque INT
)
BEGIN
    UPDATE area
    SET nombre = p_nombre, extension = p_extension, parque = p_parque
    WHERE id = p_id;
END //
```

```
DELIMITER ;
```

c. Parámetros

Agregar Área

- **p_nombre**: El nombre del área.
- **p_extension**: La extensión del área.
- **p_parque**: El ID del parque al que pertenece el área.

Eliminar Área

- **p_id**: El ID del área a eliminar.

Modificar Área

- **p_id**: El ID del área a modificar.
- **p_nombre**: El nuevo nombre del área.
- **p_extension**: La nueva extensión del área.
- **p_parque**: El nuevo ID del parque al que pertenece el área.

d. Salida/Retorno

Ninguno de estos procedimientos retorna un valor. Cada uno realiza una acción específica en la tabla **area**:

- agregar_area inserta un nuevo registro.
- eliminar_area elimina un registro existente.
- modificar_area actualiza un registro existente.

e. Ejemplo de Implementación

Agregar Área

CALL agregar_area('Área Norte', 500.00, 1);

En este ejemplo, se agrega una nueva área llamada "Área Norte" con una extensión de 500.00 y perteneciente al parque con ID 1.

Eliminar Área

CALL eliminar_area(1);

En este ejemplo, se elimina el área con ID 1.

Modificar Área

CALL modificar_area(1, 'Área Sur', 750.00, 2);

En este ejemplo, se modifica el área con ID 1 para tener el nombre "Área Sur", una extensión de 750.00 y pertenecer al parque con ID 2.

Procedimientos para Agregar, Eliminar y Modificar Personal

a. Descripción

Estos procedimientos permiten agregar, eliminar y modificar registros en la tabla **personal** y sus respectivas tablas secundarias.

- El procedimiento agregar_personal añade un nuevo personal a la base de datos.
- El procedimiento eliminar_personal elimina un personal existente por su ID.
- El procedimiento modificar_personal actualiza los detalles de un personal existente.

b. Sintaxis

```
-- agregar personal
DELIMITER //

CREATE PROCEDURE agregar_personal(
    IN p_documento INT,
    IN p_nombre VARCHAR(50),
    IN p_direccion VARCHAR(50),
    IN p_telefono VARCHAR(15),
    IN p_sueldo DECIMAL(10,2),
    IN p_tipo ENUM('Personal de Gestión', 'Personal de Vigilancia',
'Personal de Conservación', 'Personal Investigador'),
```

```
    IN p_parque INT
)
BEGIN
    INSERT INTO personal (documento, nombre, direccion, telefono, sueldo,
tipo, parque)
    VALUES (p_documento, p_nombre, p_direccion, p_telefono, p_sueldo,
p_tipo, p_parque);

    INSERT INTO history_administrador_personal (accion, registro)
    VALUES ('agregar', p_nombre);
END //

DELIMITER ;


-- modificar personal
DELIMITER //

CREATE PROCEDURE modificar_personal(
    IN p_id INT,
    IN p_documento INT,
    IN p_nombre VARCHAR(50),
    IN p_direccion VARCHAR(50),
    IN p_telefono VARCHAR(15),
    IN p_sueldo DECIMAL(10,2),
    IN p_tipo ENUM('Personal de Gestión', 'Personal de Vigilancia',
'Personal de Conservación', 'Personal Investigador'),
    IN p_parque INT
)
BEGIN
    UPDATE personal
    SET documento = p_documento,
        nombre = p_nombre,
        direccion = p_direccion,
        telefono = p_telefono,
        sueldo = p_sueldo,
        tipo = p_tipo,
        parque = p_parque
    WHERE id = p_id;

    INSERT INTO history_administrador_personal (accion, registro)
    VALUES ('modificar', p_nombre);
END //

DELIMITER ;
```

```sql
-- eliminar personal
DELIMITER //

CREATE PROCEDURE eliminar_personal(
    IN p_id INT
)
BEGIN
    -- Recuperar el nombre del personal antes de eliminarlo
    DECLARE nombre_personal VARCHAR(50);
    SELECT nombre INTO nombre_personal FROM personal WHERE id = p_id;

    -- Eliminar las filas relacionadas en investigador_proyecto primero
    DELETE FROM investigador_proyecto WHERE investigador = p_id;

    -- Eliminar las filas relacionadas en las tablas secundarias
    DELETE FROM personal_gestion WHERE id = p_id;
    DELETE FROM personal_investigacion WHERE id = p_id;
    DELETE FROM personal_conservacion WHERE id = p_id;
    DELETE FROM personal_vigilancia WHERE id = p_id;

    -- Luego eliminar la fila en la tabla personal
    DELETE FROM personal WHERE id = p_id;

    -- Registrar la acción en history_administrador_personal
    INSERT INTO history_administrador_personal (accion, registro)
    VALUES ('eliminar', nombre_personal);
END //

DELIMITER ;
```

c. Parámetros

Agregar Personal

- **p_documento**: El documento del personal.
- **p_nombre**: El nombre del personal.
- **p_direccion**: La dirección del personal.
- **p_telefono**: El teléfono del personal.

- **p_sueldo**: El sueldo del personal.
- **p_tipo**: El tipo de personal (Gestión, Vigilancia, Conservación, Investigador).
- **p_parque**: El ID del parque al que pertenece el personal.

Eliminar Personal

- **p_id**: El ID del personal a eliminar.

Modificar Personal

- **p_id**: El ID del personal a modificar.
- **p_documento**: El nuevo documento del personal.
- **p_nombre**: El nuevo nombre del personal.
- **p_direccion**: La nueva dirección del personal.
- **p_telefono**: El nuevo teléfono del personal.
- **p_sueldo**: El nuevo sueldo del personal.
- **p_tipo**: El nuevo tipo de personal.
- **p_parque**: El nuevo ID del parque al que pertenece el personal.

d. Salida/Retorno

Ninguno de estos procedimientos retorna un valor. Cada uno realiza una acción específica en la tabla **personal** y sus tablas relacionadas:

- agregar_personal inserta un nuevo registro.
- eliminar_personal elimina un registro existente.
- modificar_personal actualiza un registro existente.

e. Ejemplo de Implementación

Agregar Personal

CALL agregar_personal(12345678, 'Juan Pérez', 'Calle Falsa 123', '1234567890', 1500.00, 'Personal de Gestión', 1);

En este ejemplo, se agrega un nuevo personal con el documento 12345678, nombre "Juan Pérez", dirección "Calle Falsa 123", teléfono "1234567890", sueldo 1500.00, tipo "Personal de Gestión" y perteneciente al parque con ID 1.

Eliminar Personal

CALL eliminar_personal(1);

En este ejemplo, se elimina el personal con ID 1.

Modificar Personal

CALL modificar_personal(1, 87654321, 'Pedro González', 'Avenida Siempre Viva 742', '0987654321', 2000.00, 'Personal de Conservación', 2);

En este ejemplo, se modifica el personal con ID 1 para tener el documento 87654321, nombre "Pedro González", dirección "Avenida Siempre Viva 742", teléfono "0987654321", sueldo 2000.00, tipo "Personal de Conservación" y pertenecer al parque con ID 2.

Procedimientos para Actualizar Datos Específicos del Personal

a. Descripción

Estos procedimientos permiten actualizar datos específicos de diferentes tipos de personal (investigación, conservación, vigilancia, gestión).

- El procedimiento actualizar_titulacion actualiza la titulación de personal de investigación.
- El procedimiento actualizar_area_especialidad actualiza el área y especialidad de personal de conservación.
- El procedimiento actualizar_area_vehiculo actualiza el área y vehículo de personal de vigilancia.
- El procedimiento actualizar_punto_acceso actualiza el punto de acceso de personal de gestión.

b. Sintaxis

```
-- titulacion al personal de investigacion
DELIMITER //

CREATE PROCEDURE actualizar_titulacion(
    IN p_id INT,
    IN p_titulacion VARCHAR(25)
)
BEGIN
    UPDATE personal_investigacion
    SET titulacion = p_titulacion
    WHERE id = p_id;

    INSERT INTO history_administrador_personal (accion, registro)
    VALUES ('modificar', (SELECT nombre FROM personal_investigacion WHERE id
= p_id));
END //

DELIMITER ;


-- area y especialidad al personal de conservacion
DELIMITER //

CREATE PROCEDURE actualizar_area_especialidad(
    IN p_id INT,
    IN p_area INT,
    IN p_especialidad VARCHAR(25)
)
BEGIN
    UPDATE personal_conservacion
    SET area = p_area,
        especialidad = p_especialidad
    WHERE id = p_id;

    INSERT INTO history_administrador_personal (accion, registro)
    VALUES ('modificar', (SELECT nombre FROM personal_conservacion WHERE id
= p_id));
END //

DELIMITER ;


-- area y vehiculo al personal de vigilancia
```

```
DELIMITER //

CREATE PROCEDURE actualizar_area_vehiculo(
    IN p_id INT,
    IN p_area INT,
    IN p_vehiculo INT
)
BEGIN
    UPDATE personal_vigilancia
    SET area = p_area,
        vehiculo = p_vehiculo
    WHERE id = p_id;

    INSERT INTO history_administrador_personal (accion, registro)
    VALUES ('modificar', (SELECT nombre FROM personal_vigilancia WHERE id =
p_id));
END //

DELIMITER ;


-- punto acceso personal de gestion
DELIMITER //

CREATE PROCEDURE actualizar_punto_acceso(
    IN p_id INT,
    IN p_punto_acceso INT
)
BEGIN
    UPDATE personal_gestion
    SET punto_acceso = p_punto_acceso
    WHERE id = p_id;

    INSERT INTO history_administrador_personal (accion, registro)
    VALUES ('modificar', (SELECT nombre FROM personal_gestion WHERE id =
p_id));
END //

DELIMITER ;
```

c. Parámetros

Actualizar Titulación de Personal de Investigación

- **p_id**: El ID del personal a actualizar.
- **p_titulacion**: La nueva titulación del personal.

Actualizar Área y Especialidad de Personal de Conservación

- **p_id**: El ID del personal a actualizar.
- **p_area**: El nuevo área del personal.
- **p_especialidad**: La nueva especialidad del personal.

Actualizar Área y Vehículo de Personal de Vigilancia

- **p_id**: El ID del personal a actualizar.
- **p_area**: El nuevo área del personal.
- **p_vehiculo**: El nuevo vehículo del personal.

Actualizar Punto de Acceso de Personal de Gestión

- **p_id**: El ID del personal a actualizar.
- **p_punto_acceso**: El nuevo punto de acceso del personal.

d. Salida/Retorno

Ninguno de estos procedimientos retorna un valor. Cada uno realiza una acción específica en las tablas relacionadas del personal:

- actualizar_titulacion actualiza la titulación del personal de investigación.
- actualizar_area_especialidad actualiza el área y especialidad del personal de conservación.
- actualizar_area_vehiculo actualiza el área y vehículo del personal de vigilancia.
- actualizar_punto_acceso actualiza el punto de acceso del personal de gestión.

e. Ejemplo de Implementación

Actualizar Titulación de Personal de Investigación

CALL actualizar_titulacion(1, 'Doctorado en Biología');

En este ejemplo, se actualiza la titulación del personal de investigación con ID 1 a "Doctorado en Biología".

Actualizar Área y Especialidad de Personal de Conservación

CALL actualizar_area_especialidad(2, 3, 'Ecología');

En este ejemplo, se actualiza el área a 3 y la especialidad a "Ecología" del personal de conservación con ID 2.

Actualizar Área y Vehículo de Personal de Vigilancia

CALL actualizar_area_vehiculo(3, 2, 1);

En este ejemplo, se actualiza el área a 2 y el vehículo a 1 del personal de vigilancia con ID 3.

Actualizar Punto de Acceso de Personal de Gestión

CALL actualizar_punto_acceso(4, 5);

En este ejemplo, se actualiza el punto de acceso a 5 del personal de gestión con ID 4.

Procedimientos para Gestionar Visitantes y sus Relaciones

a. Descripción

Estos procedimientos permiten agregar, eliminar y modificar registros en la tabla **visitantes** y gestionar sus relaciones con **parques** y **alojamientos**.

- El procedimiento AgregarVisitante añade un nuevo visitante a la base de datos.
- El procedimiento ModificarVisitante actualiza los detalles de un visitante existente.
- El procedimiento EliminarVisitante elimina un visitante existente por su ID.
- El procedimiento AgregarRelacionVisitanteParque añade una nueva relación entre un visitante y un parque.

- El procedimiento EliminarRelacionVisitanteParque elimina una relación existente entre un visitante y un parque.
- El procedimiento agregar_visitante_a_alojamiento añade un visitante a un alojamiento, verificando la capacidad.
- El procedimiento eliminar_visitante_de_alojamiento elimina un visitante de un alojamiento específico.

b. Sintaxis

```
-- agregar usuario a alojamiento
DELIMITER //

CREATE PROCEDURE agregar_visitante_a_alojamiento(
    IN p_id_visitante INT,
    IN p_id_alojamiento INT
)
BEGIN
    DECLARE capacidad_actual INT;
    DECLARE capacidad_total INT;

    -- Obtener capacidad actual del alojamiento
    SELECT COUNT(*) INTO capacidad_actual
    FROM visitante_alojamiento
    WHERE alojamiento = p_id_alojamiento;

    -- Obtener capacidad total del alojamiento
    SELECT capacidad INTO capacidad_total
    FROM alojamientos
    WHERE id = p_id_alojamiento;

    -- Verificar si hay espacio disponible
    IF capacidad_actual < capacidad_total THEN
        -- Si hay espacio disponible, agregar el visitante al alojamiento
        INSERT INTO visitante_alojamiento (visitante, alojamiento)
        VALUES (p_id_visitante, p_id_alojamiento);
    ELSE
        -- Si no hay espacio, generar un error
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'El alojamiento está
completo';
    END IF;
END //
```

```
DELIMITER ;

DELIMITER //
-- eliminar visitante de alojamiento
CREATE PROCEDURE eliminar_visitante_de_alojamiento(
    IN p_id_visitante INT,
    IN p_id_alojamiento INT
)
BEGIN
    DECLARE num_rows INT;

    -- Verificar si el visitante está alojado en el alojamiento
    SELECT COUNT(*) INTO num_rows
    FROM visitante_alojamiento
    WHERE visitante = p_id_visitante AND alojamiento = p_id_alojamiento;

    -- Si el visitante está alojado, proceder con la eliminación
    IF num_rows > 0 THEN
        DELETE FROM visitante_alojamiento
        WHERE visitante = p_id_visitante AND alojamiento = p_id_alojamiento;
    ELSE
        -- Si el visitante no está alojado en el alojamiento, generar un
error
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'El visitante no está
registrado en este alojamiento';
    END IF;
END //

DELIMITER ;


-- Procedimiento para agregar un nuevo visitante
DELIMITER //

CREATE PROCEDURE AgregarVisitante(
    IN p_nombre VARCHAR(50),
    IN p_direccion VARCHAR(50),
    IN p_profesion VARCHAR(50)
)
BEGIN
    -- Insertar el nuevo visitante
    INSERT INTO visitantes (Nombre, direccion, profesion)
    VALUES (p_nombre, p_direccion, p_profesion);

    -- Registrar la acción en el historial
```

```
    INSERT INTO history_personal_gestion (accion, hora_actual)
    VALUES ('agregar', CURRENT_TIMESTAMP);

    -- Seleccionar el ID del último visitante insertado
    SELECT LAST_INSERT_ID() AS nuevo_id;
END //

DELIMITER ;


-- Procedimiento para modificar los datos de un visitante
DELIMITER //

CREATE PROCEDURE ModificarVisitante(
    IN p_id INT,
    IN p_nombre VARCHAR(50),
    IN p_direccion VARCHAR(50),
    IN p_profesion VARCHAR(50)
)
BEGIN
    -- Actualizar los datos del visitante
    UPDATE visitantes
    SET Nombre = p_nombre, direccion = p_direccion, profesion = p_profesion
    WHERE id = p_id;

    -- Registrar la acción en el historial
    INSERT INTO history_personal_gestion (accion, hora_actual)
    VALUES ('modificar', CURRENT_TIMESTAMP);
END //

DELIMITER ;



-- Procedimiento para eliminar un visitante
DELIMITER //

CREATE PROCEDURE EliminarVisitante(
    IN p_id INT
)
BEGIN
    DECLARE v_nombre VARCHAR(50);

    -- Obtener el nombre del visitante que se va a eliminar
    SELECT Nombre INTO v_nombre FROM visitantes WHERE id = p_id;
```

```
    -- Eliminar registros relacionados en otras tablas
    DELETE FROM visitante_alojamiento WHERE visitante = p_id;
    DELETE FROM visitante_parque WHERE visitante = p_id;

    -- Eliminar al visitante de la tabla visitantes
    DELETE FROM visitantes WHERE id = p_id;

    -- Registrar la acción en el historial
    INSERT INTO history_personal_gestion (accion, hora_actual)
    VALUES ('eliminar', CURRENT_TIMESTAMP);
END //

DELIMITER ;



-- Procedimiento para agregar una nueva relación entre visitante y parque
DELIMITER //

CREATE PROCEDURE AgregarRelacionVisitanteParque(
    IN p_visitante INT,
    IN p_parque INT,
    IN p_punto_acceso INT
)
BEGIN
    -- Insertar la relación entre visitante y parque
    INSERT INTO visitante_parque (visitante, parque, punto_acceso)
    VALUES (p_visitante, p_parque, p_punto_acceso);

    -- Registrar la acción en el historial
    INSERT INTO history_personal_gestion (accion, hora_actual)
    VALUES ('agregar', CURRENT_TIMESTAMP);
END //

DELIMITER ;

-- Procedimiento para eliminar una relación entre visitante y parque
DELIMITER //

CREATE PROCEDURE EliminarRelacionVisitanteParque(
    IN p_visitante INT,
    IN p_parque INT
)
BEGIN
```

```
    -- Eliminar la relación entre visitante y parque
    DELETE FROM visitante_parque
    WHERE visitante = p_visitante AND parque = p_parque;

    -- Registrar la acción en el historial
    INSERT INTO history_personal_gestion (accion, hora_actual)
    VALUES ('eliminar', CURRENT_TIMESTAMP);
END //

DELIMITER ;
```

c. Parámetros

Agregar Visitante

- **p_nombre**: El nombre del visitante.
- **p_direccion**: La dirección del visitante.
- **p_profesion**: La profesión del visitante.

Modificar Visitante

- **p_id**: El ID del visitante a modificar.
- **p_nombre**: El nuevo nombre del visitante.
- **p_direccion**: La nueva dirección del visitante.
- **p_profesion**: La nueva profesión del visitante.

Eliminar Visitante

- **p_id**: El ID del visitante a eliminar.

Agregar Relación Visitante-Parque

- **p_visitante**: El ID del visitante.
- **p_parque**: El ID del parque.
- **p_punto_acceso**: El ID del punto de acceso.

Eliminar Relación Visitante-Parque

- **p_visitante**: El ID del visitante.
- **p_parque**: El ID del parque.

Agregar Visitante a Alojamiento

- **p_id_visitante**: El ID del visitante.
- **p_id_alojamiento**: El ID del alojamiento.

Eliminar Visitante de Alojamiento

- **p_id_visitante**: El ID del visitante.
- **p_id_alojamiento**: El ID del alojamiento.

d. Salida/Retorno

- AgregarVisitante retorna el ID del nuevo visitante agregado.
- Ninguno de los otros procedimientos retorna un valor. Cada uno realiza una acción específica en las tablas relacionadas de visitantes:
    - ModificarVisitante actualiza los detalles de un visitante.
    - EliminarVisitante elimina un visitante y sus relaciones.
    - AgregarRelacionVisitanteParque añade una nueva relación entre visitante y parque.
    - EliminarRelacionVisitanteParque elimina una relación entre visitante y parque.
    - agregar_visitante_a_alojamiento añade un visitante a un alojamiento si hay capacidad.
    - eliminar_visitante_de_alojamiento elimina un visitante de un alojamiento específico.

e. Ejemplo de Implementación

Agregar Visitante

CALL AgregarVisitante('Ana López', 'Calle 45 #67-89', 'Ingeniera');

En este ejemplo, se agrega un nuevo visitante con el nombre "Ana López", dirección "Calle 45 #67-89" y profesión "Ingeniera".

Modificar Visitante

CALL ModificarVisitante(1, 'Ana López', 'Avenida 123 #45-67', 'Doctora');

En este ejemplo, se modifica el visitante con ID 1 para tener el nombre "Ana López", dirección "Avenida 123 #45-67" y profesión "Doctora".

Eliminar Visitante

CALL EliminarVisitante(1);

En este ejemplo, se elimina el visitante con ID 1.

Agregar Relación Visitante-Parque

CALL AgregarRelacionVisitanteParque(1, 2, 3);

En este ejemplo, se agrega una relación entre el visitante con ID 1 y el parque con ID 2, usando el punto de acceso con ID 3.

Eliminar Relación Visitante-Parque

CALL EliminarRelacionVisitanteParque(1, 2);

En este ejemplo, se elimina la relación entre el visitante con ID 1 y el parque con ID 2.

Agregar Visitante a Alojamiento

CALL agregar_visitante_a_alojamiento(1, 2);

En este ejemplo, se agrega el visitante con ID 1 al alojamiento con ID 2, si hay capacidad.

Eliminar Visitante de Alojamiento

CALL eliminar_visitante_de_alojamiento(1, 2);

En este ejemplo, se elimina al visitante con ID 1 del alojamiento con ID 2.

Procedimientos para Gestionar Alojamiento

a. Descripción

Estos procedimientos permiten agregar, eliminar y modificar registros en la tabla **alojamientos** y gestionar sus relaciones con **visitantes**.

- El procedimiento agregar_alojamiento añade un nuevo alojamiento a la base de datos.
- El procedimiento modificar_alojamiento actualiza los detalles de un alojamiento existente.
- El procedimiento eliminar_alojamiento elimina un alojamiento existente por su ID.

c. Parámetros

Agregar Alojamiento

- **p_nombre**: El nombre del alojamiento.
- **p_descripcion**: La descripción del alojamiento.
- **p_tipo**: El tipo de alojamiento.
- **p_parque**: El ID del parque al que pertenece el alojamiento.
- **p_capacidad**: La capacidad del alojamiento.

Modificar Alojamiento

- **p_id**: El ID del alojamiento a modificar.
- **p_nombre**: El nuevo nombre del alojamiento.
- **p_descripcion**: La nueva descripción del alojamiento.
- **p_tipo**: El nuevo tipo del alojamiento.
- **p_parque**: El nuevo ID del parque al que pertenece el alojamiento.
- **p_capacidad**: La nueva capacidad del alojamiento.

Eliminar Alojamiento

- **p_id**: El ID del alojamiento a eliminar.

d. Salida/Retorno

- agregar_alojamiento retorna el ID del nuevo alojamiento agregado.
- Ninguno de los otros procedimientos retorna un valor. Cada uno realiza una acción específica en la tabla **alojamientos** y sus relaciones:
    - modificar_alojamiento actualiza los detalles de un alojamiento.
    - eliminar_alojamiento elimina un alojamiento y sus relaciones.

e. Ejemplo de Implementación

Agregar Alojamiento

CALL agregar_alojamiento('Cabaña Las Orquídeas', 'Alojamiento familiar con vista a la montaña', 'Cabaña', 1, 5);

En este ejemplo, se agrega un nuevo alojamiento con el nombre "Cabaña Las Orquídeas", descripción "Alojamiento familiar con vista a la montaña", tipo "Cabaña", parque con ID 1 y capacidad para 5 personas.

Modificar Alojamiento

CALL modificar_alojamiento(1, 'Cabaña Las Rosas', 'Alojamiento familiar con vista al lago', 'Cabaña', 2, 6);

En este ejemplo, se modifica el alojamiento con ID 1 para tener el nombre "Cabaña Las Rosas", descripción "Alojamiento familiar con vista al lago", tipo "Cabaña", parque con ID 2 y capacidad para 6 personas.

Eliminar Alojamiento

CALL eliminar_alojamiento(1);

En este ejemplo, se elimina el alojamiento con ID 1.

Procedimientos para Gestionar Parques

a. Descripción

Estos procedimientos permiten agregar, eliminar y modificar registros en la tabla **parques** y gestionar sus relaciones con **visitantes** y **alojamientos**.

- El procedimiento agregar_parque añade un nuevo parque a la base de datos.
- El procedimiento modificar_parque actualiza los detalles de un parque existente.
- El procedimiento eliminar_parque elimina un parque existente por su ID.

b. Sintaxis

c. Parámetros

Agregar Parque

- **p_nombre**: El nombre del parque.
- **p_descripcion**: La descripción del parque.
- **p_ubicacion**: La ubicación del parque.

Modificar Parque

- **p_id**: El ID del parque a modificar.
- **p_nombre**: El nuevo nombre del parque.
- **p_descripcion**: La nueva descripción del parque.
- **p_ubicacion**: La nueva ubicación del parque.

Eliminar Parque

- **p_id**: El ID del parque a eliminar.

d. Salida/Retorno

- agregar_parque retorna el ID del nuevo parque agregado.
- Ninguno de los otros procedimientos retorna un valor. Cada uno realiza una acción específica en la tabla **parques** y sus relaciones:

- modificar_parque actualiza los detalles de un parque.
- eliminar_parque elimina un parque y sus relaciones.

e. Ejemplo de Implementación

Agregar Parque

CALL agregar_parque('Parque Nacional Los Nevados', 'Reserva natural con picos nevados', 'Caldas, Colombia');

En este ejemplo, se agrega un nuevo parque con el nombre "Parque Nacional Los Nevados", descripción "Reserva natural con picos nevados" y ubicación "Caldas, Colombia".

Modificar Parque

CALL modificar_parque(1, 'Parque Natural Chingaza', 'Bosque húmedo de montaña', 'Cundinamarca, Colombia');

En este ejemplo, se modifica el parque con ID 1 para tener el nombre "Parque Natural Chingaza", descripción "Bosque húmedo de montaña" y ubicación "Cundinamarca, Colombia".

Eliminar Parque

CALL eliminar_parque(1);

En este ejemplo, se elimina el parque con ID 1.

Procedimientos para Historial de Gestión

a. Descripción

Este procedimiento registra las acciones realizadas por el personal de gestión en la base de datos.

- El procedimiento agregar_registro_historial añade una nueva entrada al historial de gestión.

b. Sintaxis

c. Parámetros

Agregar Registro al Historial

- **p_accion**: La acción realizada por el personal de gestión.
- **p_hora_actual**: La fecha y hora en que se realizó la acción.

d. Salida/Retorno

- Ninguno de los procedimientos retorna un valor. agregar_registro_historial registra una nueva entrada en el historial de gestión.

e. Ejemplo de Implementación

Agregar Registro al Historial

CALL agregar_registro_historial('Modificación de alojamiento', CURRENT_TIMESTAMP);

En este ejemplo, se registra en el historial de gestión que se realizó una modificación en un alojamiento en el momento actual.

Triggers y eventos

Triggers

Trigger para Auto Llenar Tablas Según el Personal

a. **Descripción:** Este trigger se activa después de insertar un nuevo registro en la tabla personal. Según el tipo de personal insertado (Personal de Gestión, Personal de Vigilancia, Personal de Conservación, Personal Investigador), inserta automáticamente en la tabla correspondiente (personal_gestion, personal_vigilancia, personal_conservacion, personal_investigacion).

b. **Sintaxis:**

```
DELIMITER //
-- Trigger para auto llenar las tablas segun el personal
CREATE TRIGGER after_personal_insert
AFTER INSERT ON personal
FOR EACH ROW
BEGIN
    DECLARE tipo_persona VARCHAR(50);

    -- Determinar el tipo de persona según el campo 'tipo' en la tabla
'personal'
    SELECT CASE NEW.tipo
        WHEN 'Personal de Gestión' THEN 'gestion'
        WHEN 'Personal de Vigilancia' THEN 'vigilancia'
        WHEN 'Personal de Conservación' THEN 'conservacion'
        WHEN 'Personal Investigador' THEN 'investigacion'
        ELSE NULL
    END INTO tipo_persona;

    -- Insertar en la tabla correspondiente según el tipo
    IF tipo_persona = 'gestion' THEN
        INSERT INTO personal_gestion (id, nombre, punto_acceso)
        VALUES (NEW.id, NEW.nombre, NULL);

    ELSEIF tipo_persona = 'vigilancia' THEN
        INSERT INTO personal_vigilancia (id, nombre, area, vehiculo)
        VALUES (NEW.id, NEW.nombre, NULL, NULL);

    ELSEIF tipo_persona = 'conservacion' THEN
        INSERT INTO personal_conservacion (id, nombre, area, especialidad)
```

```
        VALUES (NEW.id, NEW.nombre, NULL, NULL);

    ELSEIF tipo_persona = 'investigacion' THEN
        INSERT INTO personal_investigacion (id, nombre, titulacion)
        VALUES (NEW.id, NEW.nombre, NULL);

    END IF;

END //

DELIMITER ;
```

c. **Parámetros:** No aplica.

d. **Salida/Retorno:** No aplica.

Trigger para Comprobar Capacidad de Alojamiento

a. **Descripción:** Este trigger se activa antes de insertar un nuevo registro en la tabla visitante_alojamiento. Verifica si el alojamiento alcanzó su capacidad máxima de visitantes antes de permitir la inserción.

b. **Sintaxis:**

```
DELIMITER //

CREATE TRIGGER check_capacity_before_insert
BEFORE INSERT ON visitante_alojamiento
FOR EACH ROW
BEGIN
    DECLARE current_count INT;
    DECLARE max_capacity INT;

    -- Contar el número actual de visitantes en el alojamiento específico
    SELECT COUNT(*) INTO current_count
    FROM visitante_alojamiento
    WHERE alojamiento = NEW.alojamiento;

    -- Verificar la capacidad máxima del alojamiento
    SELECT capacidad INTO max_capacity
    FROM alojamientos
```

```
    WHERE id = NEW.alojamiento;

    -- Si el número actual de visitantes es mayor o igual a la capacidad
máxima, lanzar un error
    IF current_count >= max_capacity THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Capacidad máxima del
alojamiento alcanzada';
    END IF;
END; //

DELIMITER ;
```

c. **Parámetros:** No aplica.

d. **Salida/Retorno:** Lanza un error si la capacidad máxima del alojamiento se ha alcanzado.

Trigger para Validar Punto de Acceso

a. **Descripción:** Este trigger se activa antes de insertar un nuevo registro en la tabla visitante_parque. Llama a un procedimiento almacenado validar_punto_acceso para verificar si el punto de acceso pertenece al parque especificado.

b. **Sintaxis:**

```
-- trigger para comprobar que el punto de acceso si pertenece a ese parque
DELIMITER //
CREATE TRIGGER trigger_validar_punto_acceso
BEFORE INSERT ON visitante_parque
FOR EACH ROW
BEGIN
    CALL validar_punto_acceso(NEW.parque, NEW.punto_acceso);
END //
DELIMITER ;
```

c. **Parámetros:** No aplica.

d. **Salida/Retorno:** No aplica.

Trigger para Validar Visitante en Parque

a. **Descripción:** Este trigger se activa antes de insertar un nuevo registro en la tabla visitante_alojamiento. Verifica si el visitante está registrado en el parque correspondiente al alojamiento.

b. **Sintaxis:**

```
-- comprobar alojamiento y parque del visitante
DELIMITER //

CREATE TRIGGER validar_visitante_en_parque
BEFORE INSERT ON visitante_alojamiento
FOR EACH ROW
BEGIN
    DECLARE parque_alojamiento INT;

    -- Obtener el parque correspondiente al alojamiento
    SELECT parque INTO parque_alojamiento
    FROM alojamientos
    WHERE id = NEW.alojamiento;

    -- Verificar que el visitante esté en el parque correspondiente
    IF NOT EXISTS (
        SELECT 1
        FROM visitante_parque
        WHERE visitante = NEW.visitante
        AND parque = parque_alojamiento
    ) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'El visitante no se encuentra en el parque
correspondiente al alojamiento.';
    END IF;
END //

DELIMITER ;
```

c. **Parámetros:** No aplica.

d. **Salida/Retorno:** Lanza un error si el visitante no se encuentra en el parque correspondiente al alojamiento.

Trigger para Verificar Área de Personal de Conservación

a. **Descripción:** Este trigger se activa antes de actualizar un registro en la tabla personal_conservacion. Verifica si el área seleccionada pertenece al parque donde trabaja el personal de conservación.

b. **Sintaxis:**

```
DELIMITER //
-- trigger para revisar que el area seleccionada para el personal de
conservacion si pertenezca al parque
CREATE TRIGGER check_area_before_update
BEFORE UPDATE ON personal_conservacion
FOR EACH ROW
BEGIN
    DECLARE parque_area INT;
    DECLARE parque_personal INT;

    -- Obtener el parque asociado al área seleccionada
    SELECT parque INTO parque_area
    FROM area
    WHERE id = NEW.area;

    -- Obtener el parque asociado al personal
    SELECT parque INTO parque_personal
    FROM personal
    WHERE id = NEW.id;

    -- Verificar si el parque del área coincide con el parque del personal
    IF parque_area != parque_personal THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'El área seleccionada no se encuentra en el
parque donde trabaja el personal.';
    END IF;
END//

DELIMITER ;
```

c. **Parámetros:** No aplica.

d. **Salida/Retorno:** Lanza un error si el área seleccionada no pertenece al parque del personal de conservación.

Trigger para Historial de Acciones del Personal de Gestión

a. **Descripción:** Este trigger se activa después de insertar, actualizar o eliminar un registro en la tabla visitantes. Registra la acción realizada en la tabla history_personal_gestion.

b. **Sintaxis:**

```
-- Trigger para registrar acciones en history_personal_gestion
DELIMITER //

CREATE TRIGGER trg_history_personal_gestion
AFTER INSERT ON visitantes
FOR EACH ROW
BEGIN
    INSERT INTO history_personal_gestion (accion)
    VALUES ('agregar');
END //

DELIMITER ;

DELIMITER //

CREATE TRIGGER trg_history_personal_gestion_modificar
AFTER UPDATE ON visitantes
FOR EACH ROW
BEGIN
    INSERT INTO history_personal_gestion (accion)
    VALUES ('modificar');
END //

DELIMITER ;

DELIMITER //

CREATE TRIGGER trg_history_personal_gestion_eliminar
AFTER DELETE ON visitantes
FOR EACH ROW
BEGIN
    INSERT INTO history_personal_gestion (accion)
    VALUES ('eliminar');
END //
```

```
DELIMITER ;

DELIMITER //
```

c. **Parámetros:** No aplica.

d. **Salida/Retorno:** No aplica.

Trigger para Verificar Punto de Acceso en Personal de Gestión

a. **Descripción:** Este trigger se activa antes de actualizar un registro en la tabla personal_gestion. Verifica si el punto de acceso seleccionado pertenece al parque donde trabaja el personal de gestión.

b. **Sintaxis:**

```
DELIMITER //
-- trigger para revisar que el punto de acceso si pertenece al parque
CREATE TRIGGER check_punto_acceso_before_update
BEFORE UPDATE ON personal_gestion
FOR EACH ROW
BEGIN
    DECLARE parque_punto_acceso INT;
    DECLARE parque_personal INT;

    -- Obtener el parque asociado al punto de acceso seleccionado
    SELECT parque INTO parque_punto_acceso
    FROM punto_acceso
    WHERE id = NEW.punto_acceso;

    -- Obtener el parque asociado al personal
    SELECT parque INTO parque_personal
    FROM personal
    WHERE id = NEW.id;

    -- Verificar si el parque del punto de acceso coincide con el parque del
personal
    IF parque_punto_acceso != parque_personal THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'El punto de acceso seleccionado no se encuentra
en el parque donde trabaja el personal.';
```

```
    END IF;
END//

DELIMITER ;
```

c. **Parámetros:** No aplica.

d. **Salida/Retorno:** Lanza un error si el punto de acceso seleccionado no pertenece al parque del personal de gestión.

e. **Ejemplo de Implementación:**

Triggers para Acciones del Administrador

a. **Descripción:** Estos triggers registran acciones realizadas por el administrador en varias tablas (departamentos, entidad_responsable, parques_naturales). Cada acción de inserción, actualización o eliminación se registra en la tabla history_administrador.

b. **Sintaxis:**

```
-- ############################ triggers para las acciones del administrador
############################
DELIMITER //

CREATE TRIGGER trg_history_administrador
AFTER INSERT ON departamentos
FOR EACH ROW
BEGIN
    INSERT INTO history_administrador (accion, registro)
    VALUES ('agregar', 'departamento');
END //

CREATE TRIGGER trg_history_administrador_update_departamentos
AFTER DELETE ON departamentos
FOR EACH ROW
BEGIN
    INSERT INTO history_administrador (accion, registro)
    VALUES ('eliminar', 'departamento');
END //

CREATE TRIGGER trg_history_administrador_insert_entidad_responsable
AFTER INSERT ON entidad_responsable
FOR EACH ROW
```

```
BEGIN
    INSERT INTO history_administrador (accion, registro)
    VALUES ('agregar', 'entidad_responsable');
END //

CREATE TRIGGER trg_history_administrador_delete_entidad_responsable
AFTER DELETE ON entidad_responsable
FOR EACH ROW
BEGIN
    INSERT INTO history_administrador (accion, registro)
    VALUES ('eliminar', 'entidad_responsable');
END //

CREATE TRIGGER trg_history_administrador_update_entidad_responsable
AFTER UPDATE ON entidad_responsable
FOR EACH ROW
BEGIN
    INSERT INTO history_administrador (accion, registro)
    VALUES ('modificar', 'entidad_responsable');
END //

CREATE TRIGGER trg_history_administrador_insert_parque
AFTER INSERT ON parques_naturales
FOR EACH ROW
BEGIN
    INSERT INTO history_administrador (accion, registro)
    VALUES ('agregar', 'parque');
END //

CREATE TRIGGER trg_history_administrador_delete_parque
AFTER DELETE ON parques_naturales
FOR EACH ROW
BEGIN
    INSERT INTO history_administrador (accion, registro)
    VALUES ('eliminar', 'parque');
END //

CREATE TRIGGER trg_history_administrador_update_parque
AFTER UPDATE ON parques_naturales
FOR EACH ROW
BEGIN
    INSERT INTO history_administrador (accion, registro)
    VALUES ('modificar', 'parque');
END //
```

```
DELIMITER ;
```

c. **Parámetros:** No aplica.

d. **Salida/Retorno:** No aplica.

Trigger para Evitar Duplicados en Departamentos

a. **Descripción:** Este trigger se activa antes de insertar un nuevo registro en la tabla departamentos. Evita la inserción si ya existe un departamento con el mismo nombre.

b. **Sintaxis:**

```
DELIMITER //

-- Trigger para evitar duplicados al agregar departamento
CREATE TRIGGER trg_check_duplicate_departamento
BEFORE INSERT ON departamentos
FOR EACH ROW
BEGIN
    DECLARE count_departamentos INT;
    SELECT COUNT(*) INTO count_departamentos FROM departamentos WHERE nombre
= NEW.nombre;

    IF count_departamentos > 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Ya existe un
departamento con este nombre.';
    END IF;
END //

-- Trigger para evitar duplicados al agregar entidad responsable
CREATE TRIGGER trg_check_duplicate_entidad_responsable
BEFORE INSERT ON entidad_responsable
FOR EACH ROW
BEGIN
    DECLARE count_entidades INT;
    SELECT COUNT(*) INTO count_entidades FROM entidad_responsable WHERE
nombre = NEW.nombre;

    IF count_entidades > 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Ya existe una entidad
responsable con este nombre.';
    END IF;
END //
```

```
-- Trigger para evitar duplicados al agregar parque natural
CREATE TRIGGER trg_check_duplicate_parque
BEFORE INSERT ON parques_naturales
FOR EACH ROW
BEGIN
    DECLARE count_parques INT;
    SELECT COUNT(*) INTO count_parques FROM parques_naturales WHERE nombre =
NEW.nombre;

    IF count_parques > 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Ya existe un parque
natural con este nombre.';
    END IF;
END //

DELIMITER ;
```

c. **Parámetros:** No aplica.

d. **Salida/Retorno:** Lanza un error si ya existe un departamento con el mismo nombre.

Triggers para Historial de Acciones en Alojamientos, Puntos de Acceso y Áreas

a. **Descripción:** Estos triggers registran acciones realizadas por el administrador en las tablas alojamientos, punto_acceso y area. Cada acción de inserción, actualización o eliminación se registra en la tabla history_administrador_parque.

b. **Sintaxis:**

```
-- Triggers para la tabla alojamientos
DELIMITER //

CREATE TRIGGER after_insert_alojamiento
AFTER INSERT ON alojamientos
FOR EACH ROW
BEGIN
    INSERT INTO history_administrador_parque (accion, registro)
    VALUES ('agregar', 'alojamiento');
END//

CREATE TRIGGER after_update_alojamiento
```

```
AFTER UPDATE ON alojamientos
FOR EACH ROW
BEGIN
    INSERT INTO history_administrador_parque (accion, registro)
    VALUES ('modificar', 'alojamiento');
END//

CREATE TRIGGER after_delete_alojamiento
AFTER DELETE ON alojamientos
FOR EACH ROW
BEGIN
    INSERT INTO history_administrador_parque (accion, registro)
    VALUES ('eliminar', 'alojamiento');
END//

-- Triggers para la tabla punto_acceso
CREATE TRIGGER after_insert_punto_acceso
AFTER INSERT ON punto_acceso
FOR EACH ROW
BEGIN
    INSERT INTO history_administrador_parque (accion, registro)
    VALUES ('agregar', 'punto_acceso');
END//

CREATE TRIGGER after_update_punto_acceso
AFTER UPDATE ON punto_acceso
FOR EACH ROW
BEGIN
    INSERT INTO history_administrador_parque (accion, registro)
    VALUES ('modificar', 'punto_acceso');
END//

CREATE TRIGGER after_delete_punto_acceso
AFTER DELETE ON punto_acceso
FOR EACH ROW
BEGIN
    INSERT INTO history_administrador_parque (accion, registro)
    VALUES ('eliminar', 'punto_acceso');
END//

-- Triggers para la tabla area
CREATE TRIGGER after_insert_area
AFTER INSERT ON area
FOR EACH ROW
BEGIN
```

```
    INSERT INTO history_administrador_parque (accion, registro)
    VALUES ('agregar', 'area');
END//

CREATE TRIGGER after_update_area
AFTER UPDATE ON area
FOR EACH ROW
BEGIN
    INSERT INTO history_administrador_parque (accion, registro)
    VALUES ('modificar', 'area');
END//

CREATE TRIGGER after_delete_area
AFTER DELETE ON area
FOR EACH ROW
BEGIN
    INSERT INTO history_administrador_parque (accion, registro)
    VALUES ('eliminar', 'area');
END//

DELIMITER ;
```

c. **Parámetros:** No aplica.

d. **Salida/Retorno:** No aplica.

Trigger para Historial de Visitantes en Alojamiento

a. **Descripción:** Este trigger se activa después de insertar un nuevo registro en la tabla visitante_alojamiento. Registra la acción realizada (agregar o eliminar visitante) en la tabla history_administrador_alojamiento.

b. **Sintaxis:**

```
-- registro de alojamiento
DELIMITER //

CREATE TRIGGER trg_history_administrador_alojamiento
AFTER INSERT ON visitante_alojamiento
FOR EACH ROW
BEGIN
    DECLARE accion VARCHAR(10);
```

```
    IF NEW.visitante IS NOT NULL THEN
        SET accion = 'agregar';
    ELSE
        SET accion = 'eliminar';
    END IF;

    INSERT INTO history_administrador_alojamiento (accion, registro)
    VALUES (accion, CONCAT('Visitante ID: ', NEW.visitante, ', Alojamiento
ID: ', NEW.alojamiento));
END //

DELIMITER ;
```

c. **Parámetros:** No aplica.

d. **Salida/Retorno:** No aplica.

Usuarios y permisos

1. Usuario Administrador:

- **Nombre de usuario:** administrador
- **Contraseña:** administrador123
- **Permisos:**
    - Tiene todos los privilegios en todas las tablas y procedimientos de la base de datos ParquesNaturalesColombia.

2. Usuario Administrador de Parques:

- **Nombre de usuario:** administrador_parque
- **Contraseña:** administradorparque123
- **Permisos:**
    - Puede usar la base de datos ParquesNaturalesColombia.
    - Puede ejecutar los siguientes procedimientos:

        - agregar_alojamiento
        - eliminar_alojamiento
        - modificar_alojamiento
        - agregar_punto_acceso
        - eliminar_punto_acceso
        - modificar_punto_acceso
        - agregar_area
        - eliminar_area
        - modificar_area

3. Usuario Administrador de Personal:

- **Nombre de usuario:** administrador_personal
- **Contraseña:** administradorpersonal123
- **Permisos:**
    - Puede usar la base de datos ParquesNaturalesColombia.
    - Puede ejecutar los siguientes procedimientos:
        - agregar_personal
        - modificar_personal
        - eliminar_personal
        - actualizar_titulacion
        - actualizar_area_especialidad
        - actualizar_area_vehiculo
        - actualizar_punto_acceso
        - asignar_proyecto_a_investigador
    - Puede seleccionar datos de las siguientes tablas:
        - personal
        - personal_conservacion
        - personal_gestion
        - personal_investigacion
        - personal_vigilancia
        - history_administrador_personal

4. Usuario Administrador de Alojamiento:

- **Nombre de usuario:** administrador_alojamiento
- **Contraseña:** administradoralojamiento123
- **Permisos:**
    - Puede usar la base de datos ParquesNaturalesColombia.
    - Puede ejecutar los siguientes procedimientos:
        - agregar_visitante_a_alojamiento
        - eliminar_visitante_de_alojamiento
    - Puede seleccionar datos de las siguientes tablas:
        - alojamientos
        - visitante_alojamiento

5. Usuario Personal de Gestión:

- **Nombre de usuario:** personal_gestion
- **Contraseña:** personalgestion123
- **Permisos:**
    - Puede usar la base de datos ParquesNaturalesColombia.
    - Puede ejecutar los siguientes procedimientos:
        - AgregarVisitante
        - ModificarVisitante
        - EliminarVisitante
        - AgregarRelacionVisitanteParque
        - EliminarRelacionVisitanteParque
    - Puede seleccionar datos de las siguientes tablas:
        - visitantes
        - visitante_parque
        - history_personal_gestion
        - history_administrador_personal
        - history_administrador_parque

Referencias

MySQL Workbench: Instrucciones para la instalación y configuración de MySQL Workbench, una herramienta esencial para la gestión de bases de datos.

- Oracle. (s.f.). MySQL Workbench: Guía del usuario. Recuperado de https://dev.mysql.com/doc/workbench/en/

Modelo de Datos: Descripción detallada del proceso de construcción del modelo conceptual, lógico y físico, incluyendo la normalización y la creación de tablas y relaciones.

- Garcia-Molina, H., Ullman, J. D., & Widom, J. (2009). Database Systems: The Complete Book (2nd ed.). Pearson Education.

Procedimientos y Triggers: Explicación de cómo implementar funciones, procedimientos almacenados, triggers y eventos en SQL para manejar la base de datos.

- Oracle. (s.f.). MySQL :: MySQL 8.0 Reference Manual :: 13 SQL Statement Syntax. Recuperado de https://docs.oracle.com/cd/E17952_01/mysql-8.0-en/