Sistema de Gestión de Biodiversidad y Conservación en Áreas Protegidas

Entrega Final

Juan Sebastian Cenon Quintero

Andres Felipe Beltran Sicua

Juan Diego Carreño Vasquez

Pontificia Universidad Javeriana de Colombia

Bases de Datos

Bogotá, Colombia

5 de Noviembre de 2024

Resumen

El proyecto "Sistema de Gestión de Biodiversidad y Conservación en Áreas Protegidas" tiene como objetivo desarrollar una base de datos para gestionar la información sobre la biodiversidad y las actividades de conservación en áreas protegidas. Este sistema permite registrar y organizar datos relacionados con indicadores de biodiversidad, especies,actividades de conservación, las amenazas y observaciones, asegurando la integridad y consistencia de la información mediante restricciones y relaciones de integridad referencial. A través de consultas SQL, se facilita el análisis y la generación de reportes que apoyan la toma de decisiones en la gestión ambiental.

Palabras clave: Biodiversidad, conservación, áreas protegidas, gestión ambiental, base de datos.

Abstract

The project "Biodiversity Management and Conservation System in Protected Areas" aims to develop a database to manage information on biodiversity and conservation activities in protected areas. This system allows for the recording and organization of data related to biodiversity indicators, species, conservation activities, threats, and observations, ensuring data integrity and consistency through constraints and referential integrity relationships. SQL queries facilitate data analysis and the generation of reports that support decision-making in environmental management.

Keywords:Biodiversity, conservation, protected areas, environmental management, database.

1. Introducción

La conservación de la biodiversidad es fundamental para el mantenimiento de ecosistemas saludables y funcionales, que a su vez son esenciales para el bienestar humano y la estabilidad ambiental global. Las áreas protegidas juegan un papel crucial en la preservación de la biodiversidad, ya que actúan como refugios para especies en peligro y proporcionan un espacio donde los ecosistemas pueden prosperar sin la intervención directa de actividades humanas perjudiciales. Sin embargo, gestionar y conservar efectivamente estas áreas requiere un manejo cuidadoso y preciso de la información relacionada con los diversos indicadores de biodiversidad, las especies presentes, las actividades de conservación llevadas a cabo, y las amenazas potenciales que pueden impactar estas áreas.

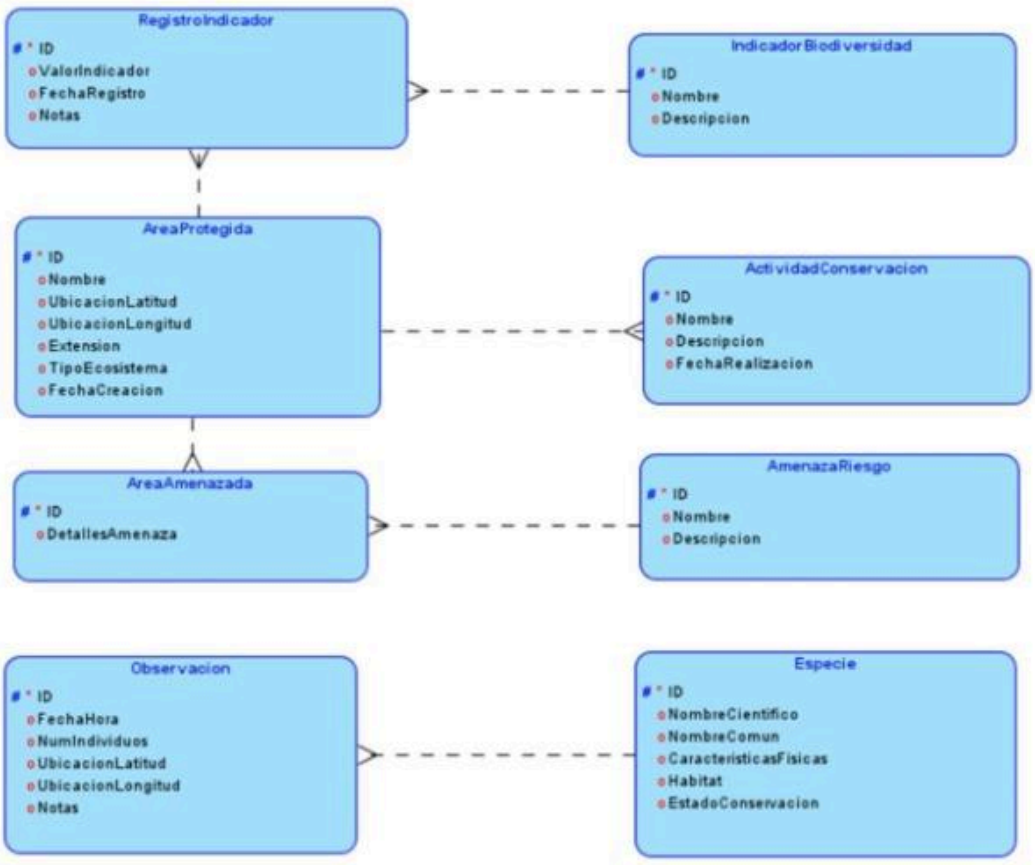
El proyecto "Sistema de Gestión de Biodiversidad y Conservación en Áreas Protegidas" se desarrolla con el propósito de crear una base de datos eficiente que permita almacenar, gestionar y analizar información clave sobre biodiversidad y conservación en áreas protegidas. Este sistema está diseñado para registrar datos detallados de manera estructurada, abarcando desde indicadores de biodiversidad y especies hasta actividades de conservación y amenazas. Además, el uso de restricciones y relaciones de integridad referencial asegura la consistencia y precisión de la información almacenada, proporcionando una herramienta robusta para investigadores, conservacionistas y administradores de áreas protegidas.

Mediante consultas SQL, el sistema facilita la extracción y el análisis de datos, lo cual es crucial para generar reportes que apoyen la toma de decisiones en la gestión ambiental. Estos reportes ayudan a evaluar el estado de la biodiversidad, monitorear la efectividad de las actividades de conservación y responder adecuadamente a las amenazas emergentes. En última instancia, este sistema busca contribuir significativamente a los esfuerzos de

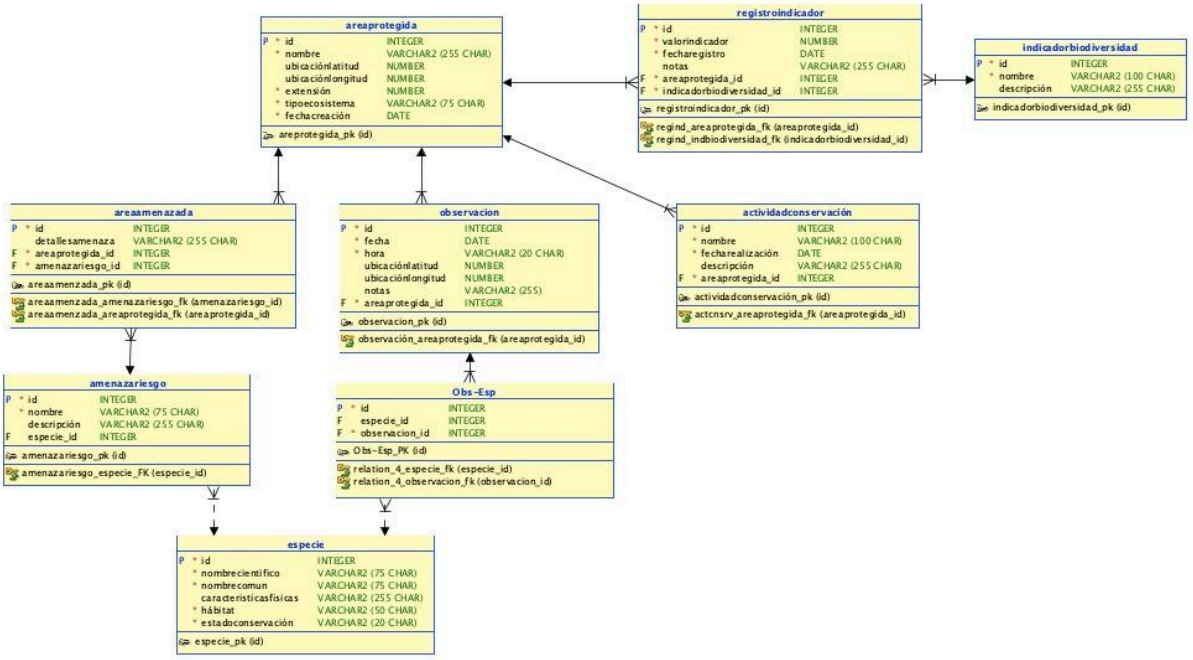
conservación mediante una gestión de información más eficaz y basada en datos en las áreas protegidas.

2. Desarrollo de la base de datos

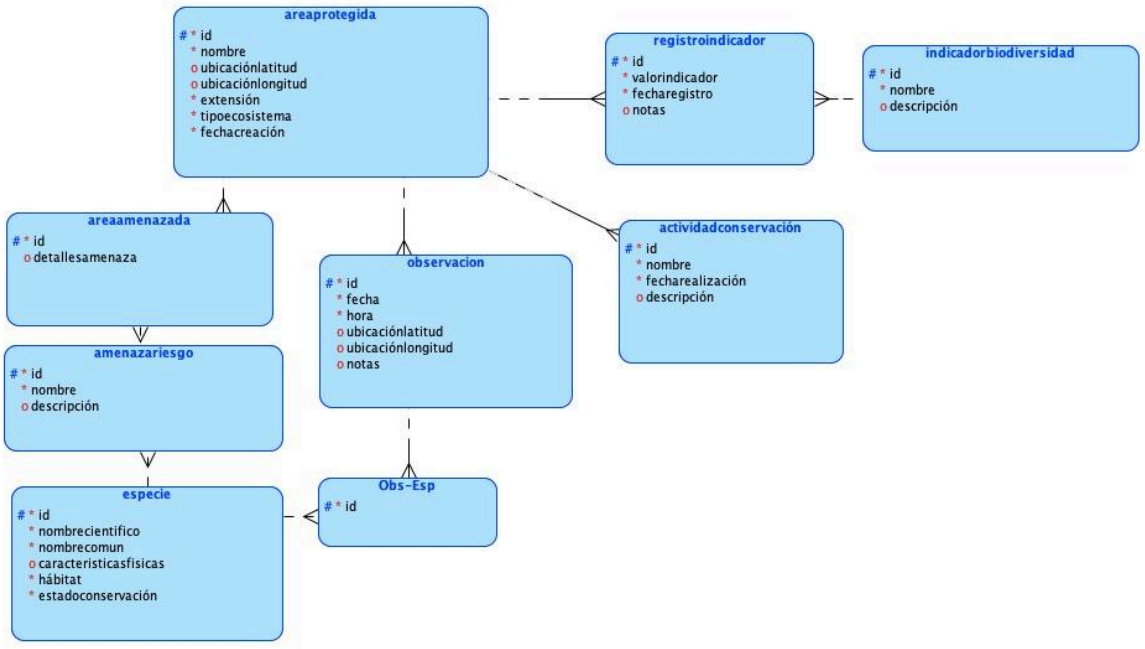
2.1 Modelo Lógico planteado por el docente y nuestro modelo de la entrega #1



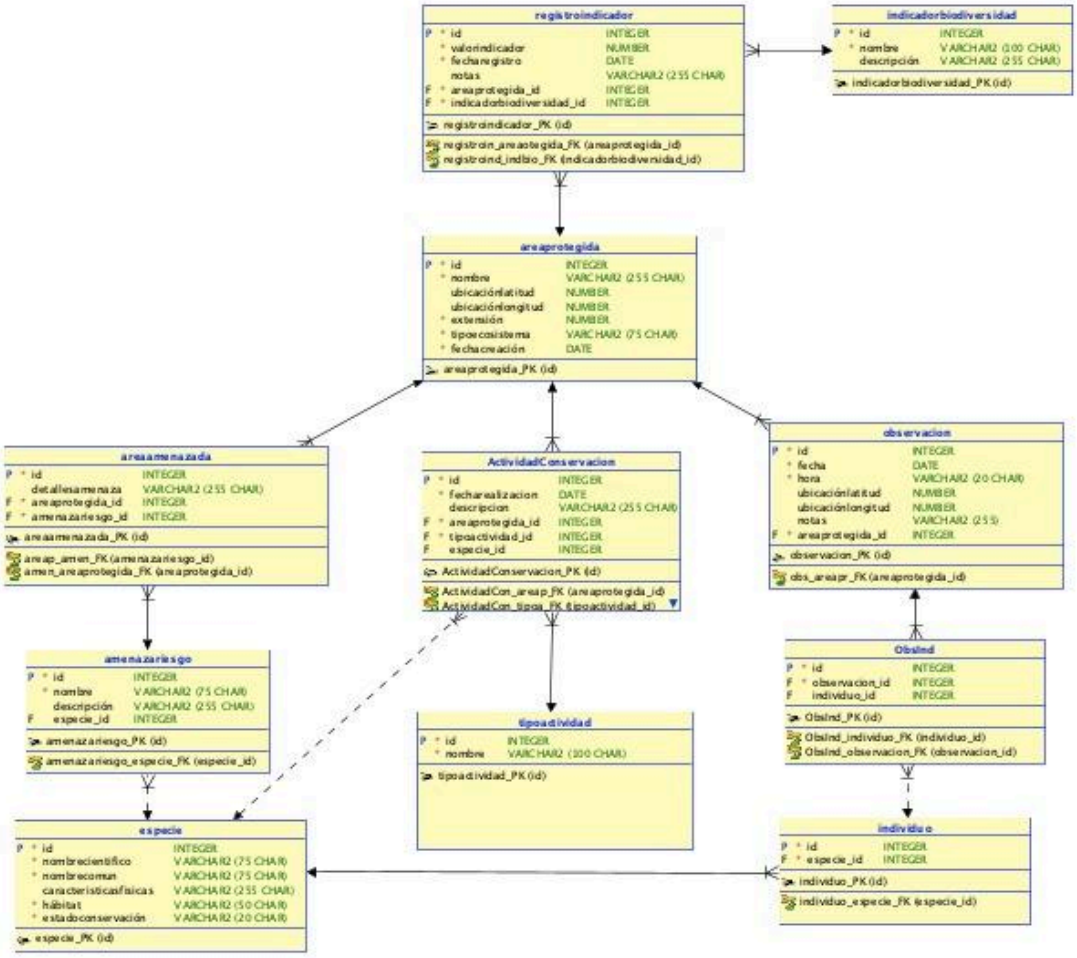
2.1 Modelo Físico de la entrega #2



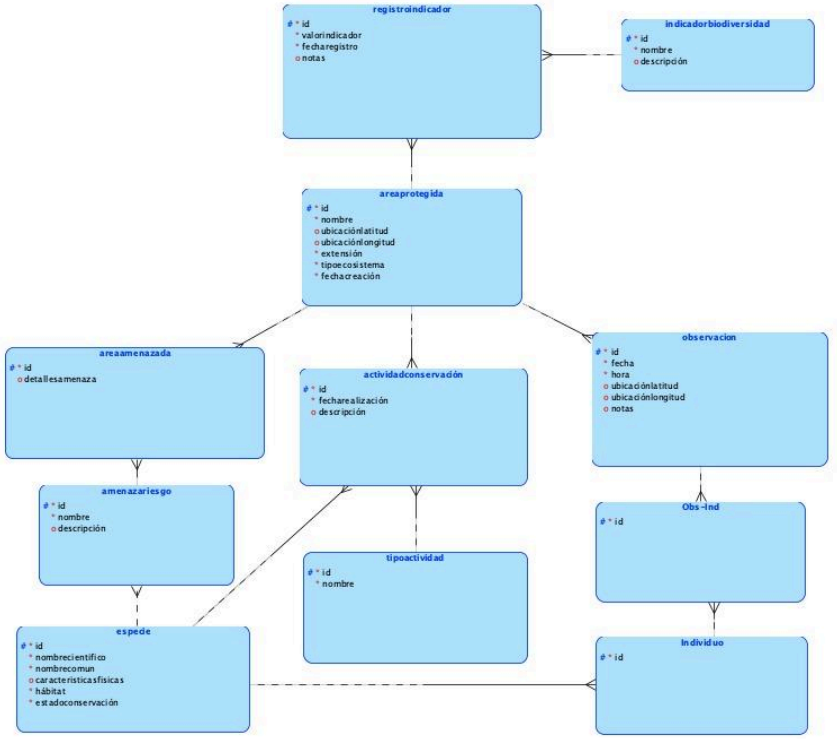
2.1 Modelo Lógico de la entrega #2



2.3 Modelo Físico de la aplicación


registroindicador
P * id INTEGER
* valorindicador NUMBER
* fecharegistro DATE
notas VARCHAR2 (255 CHAR)
F * areaprotegida_id INTEGER
F * indicadorbiodiversidad_id INTEGER
registroindicador_PK (id)
registroin_areaotegida_FK (areaprotegida_id)
registroind_indbio_FK (indicadorbiodiversidad_id)
indicadorbiodiversidad
P * id INTEGER
* nombre VARCHAR2 (100 CHAR)
descripción VARCHAR2 (255 CHAR)
indicadorbiodiversidad_PK (id)
areaprotegida
P * id INTEGER
* nombre VARCHAR2 (255 CHAR)
ubicaciónlatitud NUMBER
ubicaciónlongitud NUMBER
* extensión NUMBER
* tipoecosistema VARCHAR2 (75 CHAR)
* fechacreación DATE
areaprotegida_PK (id)
areaamenazada
P * id INTEGER
detallesamenaza VARCHAR2 (255 CHAR)
F * areaprotegida_id INTEGER
F * amenazariesgo_id INTEGER
areaamenazada_PK (id)
areap_amen_FK (amenazariesgo_id)
amen_areaprotegida_FK (areaprotegida_id)
ActividadConservacion
P * id INTEGER
* fecharealizacion DATE
descripcion VARCHAR2 (255 CHAR)
F * areaprotegida_id INTEGER
F * tipoactividad_id INTEGER
F especie_id INTEGER
ActividadConservacion_PK (id)
ActividadCon_areap_FK (areaprotegida_id)
ActividadCon_tipoa_FK (tipoactividad_id)
observacion
P * id INTEGER
* fecha DATE
* hora VARCHAR2 (20 CHAR)
ubicaciónlatitud NUMBER
ubicaciónlongitud NUMBER
notas VARCHAR2 (255)
F * areaprotegida_id INTEGER
observacion_PK (id)
obs_areapr_FK (areaprotegida_id)
amenazariesgo
P * id INTEGER
* nombre VARCHAR2 (75 CHAR)
descripción VARCHAR2 (255 CHAR)
F especie_id INTEGER
amenazariesgo_PK (id)
amenazariesgo_especie_FK (especie_id)
tipoactividad
P * id INTEGER
* nombre VARCHAR2 (100 CHAR)
tipoactividad_PK (id)
ObsInd
P * id INTEGER
F * observacion_id INTEGER
F individuo_id INTEGER
ObsInd_PK (id)
ObsInd_individuo_FK (individuo_id)
ObsInd_observacion_FK (observacion_id)
especie
P * id INTEGER
* nombrecientifico VARCHAR2 (75 CHAR)
* nombrecomun VARCHAR2 (75 CHAR)
caracteristicasfisicas VARCHAR2 (255 CHAR)
* hábitat VARCHAR2 (50 CHAR)
* estadoconservación VARCHAR2 (20 CHAR)
especie_PK (id)
individuo
P * id INTEGER
F * especie_id INTEGER
individuo_PK (id)
individuo_especie_FK (especie_id)

2.4 Modelo Lógico de aplicación



3. Lenguaje procedural SQL (PL/SQL)

3.1 Funciones

1. Calcular la media de la extensión de todas las Áreas Protegidas

```
CREATE OR REPLACE FUNCTION calcular_media_extension_areas_protegidas
RETURN NUMBER IS
    media_extension NUMBER;
BEGIN
    SELECT AVG(extensión) INTO media_extension
    FROM areaprotegida;

    RETURN media_extension;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL; -- Si no hay áreas protegidas, devolver NULL
    WHEN OTHERS THEN
        RAISE; -- Propagar cualquier otro error
END calcular_media_extension_areas_protegidas;
```

2. Contar el total de Observaciones realizadas por una Especie específica

```
CREATE OR REPLACE FUNCTION contar_observaciones_por_especie(p_especie_id IN INTEGER)
RETURN INTEGER IS
    total_observaciones INTEGER;
BEGIN
    SELECT COUNT(*) INTO total_observaciones
    FROM observacion o
    JOIN obsind oi ON o.id = oi.observacion_id
    JOIN individuo i ON oi.individuo_id = i.id
    WHERE i.especie_id = p_especie_id;

    RETURN total_observaciones;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 0; -- Si no hay observaciones, devolver 0
    WHEN OTHERS THEN
        RAISE; -- Propagar cualquier otro error
END contar_observaciones_por_especie;
```

3. Obtener el nombre de la Especie con el mayor número de Observaciones

```
CREATE OR REPLACE FUNCTION obtener_especie_mas_observaciones
RETURN VARCHAR2 IS
    especie_con_mas_observaciones VARCHAR2(75);
    max_observaciones INTEGER;
BEGIN
    SELECT e.nombrecientifico
    INTO especie_con_mas_observaciones
    FROM especie e
    JOIN (
        SELECT i.especie_id, COUNT(o.id) AS total_observaciones
        FROM observacion o
        JOIN obsind oi ON o.id = oi.observacion_id
        JOIN individuo i ON oi.individuo_id = i.id
        GROUP BY i.especie_id
    ) observaciones_por_especie ON e.id = observaciones_por_especie.especie_id
    WHERE observaciones_por_especie.total_observaciones = (
        SELECT MAX(total_observaciones)
        FROM (
            SELECT COUNT(o.id) AS total_observaciones
            FROM observacion o
            JOIN obsind oi ON o.id = oi.observacion_id
            JOIN individuo i ON oi.individuo_id = i.id
            GROUP BY i.especie_id
        )
    );

    RETURN especie_con_mas_observaciones;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL; -- Si no hay observaciones, devolver NULL
    WHEN OTHERS THEN
        RAISE; -- Propagar cualquier otro error
END obtener_especie_mas_observaciones;
```

3.2 Procedimientos

Registrar una nueva observación:

```
CREATE OR REPLACE PROCEDURE registrar_observacion (
    p_fecha             DATE,
    p_hora              VARCHAR2,
    p_ubicacion_latitud NUMBER,
    p_ubicacion_longitud NUMBER,
    p_notas             VARCHAR2,
    p_areaprotegida_id  INTEGER
) AS
BEGIN
    INSERT INTO observacion (id, fecha, hora, ubicaciónlatitud, ubicaciónlongitud, notas, areaprotegida_id)
    VALUES (observacion_seq.NEXTVAL, p_fecha, p_hora, p_ubicacion_latitud, p_ubicacion_longitud, p_notas, p_areaprotegida_id);

    COMMIT;
END;
```

Actualizar el estado de conservación de una especie:

```
CREATE OR REPLACE PROCEDURE actualizar_estado_conservacion (
    p_especie_id          INTEGER,
    p_nuevo_estado        VARCHAR2
) AS
BEGIN
    UPDATE especie
    SET estadoconservación = p_nuevo_estado
    WHERE id = p_especie_id;

    COMMIT;
END;
```

Eliminar todas las observaciones asociadas a una especie específica

```
CREATE OR REPLACE PROCEDURE eliminar_observaciones_especie (
    p_especie_id INTEGER
) AS
BEGIN
    DELETE FROM observacion
    WHERE id IN (
        SELECT oi.observacion_id
        FROM obsind oi
        JOIN individuo i ON oi.individuo_id = i.id
        WHERE i.especie_id = p_especie_id
    );

    COMMIT;
END;
```

3.3 Disparadores

Disparador para mantener un registro de actividad de Observaciones

Para implementar el disparador que registra automáticamente cada cambio en la tabla “observacion”, primero es necesario crear la tabla “registro_actividad”. Esta tabla se encargará de almacenar el historial de modificaciones, permitiendo llevar un control detallado de las operaciones realizadas sobre las observaciones de especies o áreas protegidas. La estructura de “registro_actividad” incluye un identificador único como clave primaria, un campo para el identificador de la observación modificada, el tipo de operación (como inserción, actualización o eliminación), y la fecha y hora en que ocurrió el cambio. La creación de esta tabla es esencial para que el disparador funcione correctamente, ya que será el lugar donde se registren todos los eventos de modificación en la tabla observacion.

```
CREATE TABLE registro_actividad (
    id                INTEGER PRIMARY KEY,
    observacion_id    INTEGER,
    accion            VARCHAR2(10 CHAR) NOT NULL, -- 'INSERT', 'UPDATE', 'DELETE'
    fecha_actividad   DATE DEFAULT SYSDATE NOT NULL
);
```

La secuencia “registro_actividad_seq” se crea para generar valores únicos e incrementales para la columna id en la tabla “registro_actividad”. Esto asegura que cada registro de actividad tenga un identificador único, evitando conflictos y permitiendo un seguimiento ordenado de las acciones realizadas.

```
CREATE SEQUENCE registro_actividad_seq START WITH 1 INCREMENT BY 1;
```

El disparador “trg_registro_actividad_observacion” se ha diseñado para registrar automáticamente cada inserción, actualización o eliminación en la tabla observacion. Este disparador se activa después de cada una de estas operaciones y registra el tipo de acción realizada en la tabla “registro_actividad”, lo que permite llevar un historial completo de las modificaciones en las observaciones de especies o áreas protegidas. El disparador identifica el tipo de operación mediante condiciones internas para determinar si se trata de una

inserción, actualización o eliminación, y luego inserta un nuevo registro en "registro_actividad". En este registro se almacena el identificador de la observación modificada, el tipo de acción realizada y la fecha y hora exacta del cambio. Con esta configuración, el disparador asegura un seguimiento continuo y automático de todas las actividades en la tabla "observacion", facilitando un control exhaustivo de los cambios realizados.

```
CREATE OR REPLACE TRIGGER trg_registro_actividad_observacion
AFTER INSERT OR UPDATE OR DELETE ON observacion
FOR EACH ROW
DECLARE
    v_accion VARCHAR2(10 CHAR);
BEGIN
    -- Determinar la acción realizada
    IF INSERTING THEN
        v_accion := 'INSERT';
    ELSIF UPDATING THEN
        v_accion := 'UPDATE';
    ELSIF DELETING THEN
        v_accion := 'DELETE';
    END IF;

    -- Insertar registro de actividad en la tabla registro_actividad
    INSERT INTO registro_actividad (id, observacion_id, accion, fecha_actividad)
    VALUES (
        registro_actividad_seq.NEXTVAL,  -- ID autogenerado por la secuencia
        NVL(:OLD.id, :NEW.id),           -- ID de la observación afectada
        v_accion,                        -- Acción realizada
        SYSDATE                          -- Fecha y hora actual
    );
END;
```

Disparador para mantener actualizado el total de Observaciones por Especie: Este disparador se activará automáticamente cada vez que se inserte, actualice o elimine una Observación y actualizará el campo Total Observaciones en la tabla de Especies.

El disparador trg_actualizar_total_observaciones se diseñó para mantener actualizado el total de observaciones por especie en la base de datos. Para lograr esto, se agregó primero un nuevo campo, total_observaciones, en la tabla especie, que almacenará la cantidad total de observaciones asociadas a cada especie. Este campo se inicializa en cero y se actualizará

automáticamente cada vez que se realicen cambios en las observaciones relacionadas a los individuos de esa especie. El disparador se activa después de cualquier operación de inserción, actualización o eliminación en la tabla obsind, que es la tabla que establece la relación entre observaciones e individuos.

```
ALTER TABLE especie ADD total_observaciones INTEGER DEFAULT 0;
```

Dentro del cuerpo del disparador, se utiliza la cláusula DECLARE para definir variables que almacenarán el ID de la especie y el total de observaciones. Cuando se inserta o actualiza una entrada en obsind, se determina el ID de la especie asociado utilizando una consulta que une las tablas individuo, obsind, observacion y especie. En el caso de una eliminación, se sigue un proceso similar para obtener el ID de la especie afectada. Posteriormente, se realiza una consulta que cuenta las observaciones relacionadas con esa especie utilizando COUNT(DISTINCT o.id), lo que asegura que cada observación se cuente solo una vez. Finalmente, el disparador actualiza el campo total_observaciones en la tabla especie con el nuevo total, garantizando que los datos de la base de datos reflejen correctamente la cantidad de observaciones disponibles para cada especie en todo momento.

```
BEGIN
    -- Determinar el id de la especie relacionado con el individuo de la observación
    IF INSERTING OR UPDATING THEN
        SELECT e.id
        INTO v_especie_id
        FROM individuo i
        JOIN obsind o ON i.id = :NEW.individuo_id
        JOIN observacion obs ON obs.id = o.observacion_id
        JOIN especie e ON e.id = i.especie_id
        WHERE obs.id = :NEW.observacion_id;

    ELSIF DELETING THEN
        SELECT e.id
        INTO v_especie_id
        FROM individuo i
        JOIN obsind o ON i.id = :OLD.individuo_id
        JOIN observacion obs ON obs.id = o.observacion_id
        JOIN especie e ON e.id = i.especie_id
        WHERE obs.id = :OLD.observacion_id;
    END IF;

    -- Contar el total de observaciones para la especie
    SELECT COUNT(DISTINCT o.id)
    INTO v_total
    FROM observacion o
    JOIN obsind oi ON o.id = oi.observacion_id
    JOIN individuo i ON i.id = oi.individuo_id
    WHERE i.especie_id = v_especie_id;

    -- Actualizar el total de observaciones en la tabla especie
    UPDATE especie
    SET total_observaciones = v_total
    WHERE id = v_especie_id;
END;
```

Disparador para prevenir la eliminación de Especies con Observaciones asociadas: Este disparador evitará que se eliminen Especies que tienen Observaciones asociadas en la base de datos.

El disparador “trg_prevenir_eliminacion_especie” se configura para activarse antes de que se realice una eliminación en la tabla especie. Su objetivo es garantizar la integridad de los datos al prevenir la eliminación de especies que tienen observaciones asociadas en la base de datos. Para lograr esto, el disparador ejecuta una consulta que cuenta el número de observaciones relacionadas con la especie que se intenta eliminar. Esta consulta se realiza mediante un JOIN entre la tabla individuo y la tabla “obsind”, utilizando el identificador de la especie en la condición de búsqueda. Si se encuentra al menos una observación relacionada

(es decir, si el contador es mayor que cero), el disparador lanza una excepción a través de RAISE_APPLICATION_ERROR, deteniendo así el proceso de eliminación.

```
CREATE OR REPLACE TRIGGER trg_prevenir_eliminacion_especie
BEFORE DELETE ON especie
FOR EACH ROW
DECLARE
    v_count INTEGER;
BEGIN
    -- Contar el número de observaciones asociadas a la especie que se intenta eliminar
    SELECT COUNT(*)
    INTO v_count
    FROM individuo i
    JOIN obsind o ON i.id = o.individuo_id
    WHERE i.especie_id = :OLD.id;

    -- Si hay observaciones asociadas, lanzar una excepción
    IF v_count > 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'No se puede eliminar la especie porque tiene observaciones asociadas.');
    END IF;
END;
```

La implementación de este disparador es fundamental para mantener la consistencia en la base de datos, ya que las especies y sus observaciones están intrínsecamente relacionadas. Permitir la eliminación de una especie con observaciones asociadas podría resultar en datos huérfanos o inconsistentes, lo que afectaría la calidad de la información. Al lanzar una excepción con un mensaje claro, se proporciona a los usuarios de la base de datos un entendimiento inmediato del problema, lo que facilita la gestión de datos y asegura que las operaciones de eliminación se realicen de manera controlada y consciente.

4. Conclusiones

Integración y consistencia en la gestión de datos de biodiversidad. La implementación de una base de datos estructurada con restricciones y relaciones de integridad referencial permite asegurar la precisión y consistencia de la información sobre biodiversidad y actividades de conservación. Esto facilita el manejo de datos relacionados con especies, áreas protegidas, amenazas y observaciones, proporcionando una plataforma confiable para la toma de decisiones informadas en la gestión ambiental.

Automatización y control mediante procedimientos y disparadores. El uso de procedimientos almacenados y disparadores SQL ayuda a mantener la integridad y actualización de los datos en tiempo real. Los disparadores implementados, como el que previene la eliminación de especies con observaciones asociadas y los que actualizan automáticamente los totales de observaciones, permiten un control exhaustivo y garantizan que los cambios en los datos no afecten la calidad o precisión de la información**.**

Facilitación de análisis y generación de reportes para la conservación. La base de datos permite realizar consultas avanzadas que facilitan el análisis de la biodiversidad en áreas protegidas. Con procedimientos que calculan indicadores clave, como la media de extensión de las áreas protegidas o la especie con más observaciones, el sistema apoya la generación de reportes útiles para los gestores y conservacionistas, contribuyendo a una gestión más eficaz y sustentada en datos.

Referencias

Protected Planet. (s.f.). *World Database on Protected Areas (WDPA)*. United Nations Environment Programme. https://www.protectedplanet.net/en