Gestión de Cambios y Procedimientos en la Librería y el Hospital

En esta documentación, se detalla la implementación de procedimientos almacenados y triggers en las bases de datos de la librería y el hospital para realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar) en las tablas correspondientes, así como el registro de cambios en tablas de control de cambios.

Procedimientos Almacenados para la Librería:

Se han creado 4 procedimientos almacenados que permiten realizar operaciones en la tabla Editorial

Procedimiento para agregar un nuevo registro en la tabla `Editorial`:

```
DELIMITER //

CREATE PROCEDURE agregar_editorial(
    IN p_nombre VARCHAR(50),
    IN p_ciudad VARCHAR(30),
    IN p_complemento VARCHAR(100),
    IN p_telefono VARCHAR(20)
)
BEGIN
    INSERT INTO Editorial (nombre, ciudad, complemento, Telefono
    VALUES (p_nombre, p_ciudad, p_complemento, p_telefono);
END //
```

```
DELIMITER ;
```

Procedimiento para actualizar un registro en la tabla `Editorial`:

```
DELIMITER //

CREATE PROCEDURE actualizar_editorial(
    IN p_nombre VARCHAR(50),
    IN p_ciudad VARCHAR(30),
    IN p_complemento VARCHAR(100),
    IN p_telefono VARCHAR(20)
)
BEGIN
    UPDATE Editorial
    SET ciudad = p_ciudad, complemento = p_complemento, Telefono
    WHERE nombre = p_nombre;
END //

DELIMITER ;
```

Procedimiento para borrar un registro de la tabla `Editorial`:

```
DELIMITER //

CREATE PROCEDURE borrar_editorial(
    IN p_nombre VARCHAR(50)
)
BEGIN
    DELETE FROM Editorial WHERE nombre = p_nombre;
END //

DELIMITER ;
```

Procedimiento para actualizar un registro de la tabla `Editorial`:

```
DELIMITER //

CREATE PROCEDURE actualizar_editorial(
    IN p_nombre VARCHAR(50),
    IN p_ciudad VARCHAR(30),
    IN p_complemento VARCHAR(100),
    IN p_telefono VARCHAR(20)
)
BEGIN
    UPDATE Editorial
    SET ciudad = p_ciudad, complemento = p_complemento, Telefono
    WHERE nombre = p_nombre;
END //

DELIMITER ;
```

Estos procedimientos proporcionan una interfaz segura y controlada para realizar operaciones en la tabla de la librería.

Control de Cambios en la Librería:

Se ha creado una nueva tabla llamada "control_de_cambios_librería" que registra cambios en la tabla de la librería. Esta tabla tiene tres columnas: usuario, acción y fecha. Además, se han implementado 2 triggers para registrar el nombre del usuario que realizó una acción (agregar o eliminar un registro) en la tabla de la librería.

```
-- Crear la tabla control_de_cambios_librería
CREATE TABLE IF NOT EXISTS control_de_cambios_librería (
    id INT AUTO_INCREMENT PRIMARY KEY,
    usuario VARCHAR(100),
    accion VARCHAR(100),
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Triggers

```
-- Trigger para insertar un registro en control_de_cambios_libre
DELIMITER //
CREATE TRIGGER after_insert_editorial
AFTER INSERT ON Editorial
FOR EACH ROW
BEGIN
    INSERT INTO control_de_cambios_librería (usuario, accion) VA
END //
DELIMITER ;

-- Trigger para insertar un registro en control_de_cambios_libre
DELIMITER //
CREATE TRIGGER after_delete_editorial
AFTER DELETE ON Editorial
FOR EACH ROW
BEGIN
    INSERT INTO control_de_cambios_librería (usuario, accion) VA
END //
DELIMITER ;
```

Procedimientos Almacenados para el Hospital:

Se han desarrollado 4 procedimientos almacenados similares para realizar operaciones en una de la tabla paciente del hospital. Estos procedimientos permiten agregar, actualizar, consultar y eliminar registros en la tabla correspondiente del hospital.

Procedimiento almacenado para agregar un registro tabla Paciente:

```
DELIMITER //

CREATE PROCEDURE sp_agregar_registro (
    IN p_nombre VARCHAR(255),
    IN p_apellido VARCHAR(255),
    IN p_telefono VARCHAR(20),
    IN p_direccion VARCHAR(255)
```

```
)
BEGIN
    INSERT INTO tb_paciente (nombre_paciente, apellido_paciente,
    VALUES (p_nombre, p_apellido, p_telefono, p_direccion);
END //

DELIMITER ;
```

Procedimiento almacenado para actualizar un registro tabla Paciente:

```
DELIMITER //

CREATE PROCEDURE sp_actualizar_registro (
    IN p_id INT,
    IN p_nombre VARCHAR(255),
    IN p_apellido VARCHAR(255),
    IN p_telefono VARCHAR(20),
    IN p_direccion VARCHAR(255)
)
BEGIN
    UPDATE tb_paciente
    SET nombre_paciente = p_nombre, apellido_paciente = p_apelli
    WHERE id_paciente = p_id;
END //

DELIMITER ;
```

Procedimiento almacenado para consultar un registro tabla Paciente:

```
DELIMITER //

CREATE PROCEDURE sp_consultar_registro (
    IN p_id INT
)
BEGIN
    SELECT * FROM tb_paciente WHERE id_paciente = p_id;
```

```
END //

DELIMITER ;
```

Procedimiento almacenado para eliminar un registro tabla Paciente:

```
DELIMITER //

CREATE PROCEDURE sp_borrar_registro (
    IN p_id INT
)
BEGIN
    DELETE FROM tb_paciente WHERE id_paciente = p_id;
END //

DELIMITER ;
```

Control de Cambios en el Hospital:

Al igual que en la librería, se ha creado una nueva tabla llamada "control_de_cambios_hospital" para registrar cambios en la tabla del hospital. Esta tabla también tiene tres columnas: usuario, acción y fecha. Se han implementado 2 triggers para registrar el nombre del usuario que realizó una acción (agregar o eliminar un registro) en la tabla del hospital.

```
CREATE TABLE IF NOT EXISTS control_de_cambios_hospital (
    id INT AUTO_INCREMENT PRIMARY KEY,
    usuario VARCHAR(100),
    accion VARCHAR(100),
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Triggers

```
DELIMITER //
```

```
CREATE TRIGGER tr_insertar_control_insert AFTER INSERT ON tb_pac
FOR EACH ROW
BEGIN
    -- Registra inserción en control_de_cambios_hospital
    INSERT INTO control_de_cambios_hospital (usuario, accion, fe
END //

CREATE TRIGGER tr_insertar_control_delete AFTER DELETE ON tb_pac
FOR EACH ROW
BEGIN
    -- Registra eliminación en control_de_cambios_hospital
    INSERT INTO control_de_cambios_hospital (usuario, accion, fe
END //

DELIMITER ;
```

Estos procedimientos almacenados y triggers proporcionan un mecanismo robusto para realizar y auditar cambios en las bases de datos de la librería y el hospital, garantizando la integridad y la trazabilidad de los datos. Además, permiten un seguimiento detallado de las operaciones realizadas por los usuarios en ambas bases de datos.