Implementación SQL

Scripts para creación de tablas y restricciones

```
CREATE TABLE paciente (
    id_paciente INT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    fecha_nacimiento DATE NOT NULL,
    telefono VARCHAR(20),
    email VARCHAR(100),
    genero VARCHAR(1)
);

CREATE TABLE clinica (
    id_clinica INT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    ciudad VARCHAR(50) NOT NULL,
    direccion VARCHAR(200)
);

CREATE TABLE doctor (
    id_doctor INT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    especialidad VARCHAR(100) NOT NULL,
    id_clinica INT NOT NULL,
    FOREIGN KEY (id_clinica) REFERENCES clinica(id_clinica)
);

CREATE TABLE cita (
```

```
    id_cita INT PRIMARY KEY,
    id_paciente INT NOT NULL,
    id_doctor INT NOT NULL,
    fecha_cita DATETIME NOT NULL,
    motivo VARCHAR(200) NOT NULL,
    estado ENUM('PENDIENTE', 'COMPLETADA', 'CANCELADA') NOT NULL,
    FOREIGN KEY (id_paciente) REFERENCES paciente(id_paciente),
    FOREIGN KEY (id_doctor) REFERENCES doctor(id_doctor)
);

CREATE TABLE historial_medico (
    id_historial INT PRIMARY KEY,
    id_paciente INT NOT NULL,
    descripcion TEXT NOT NULL,
    fecha_registro TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
    FOREIGN KEY (id_paciente) REFERENCES paciente(id_paciente)
);

CREATE TABLE receta (
    id_receta INT PRIMARY KEY,
    id_cita INT NOT NULL,
    fecha_emision DATE NOT NULL DEFAULT (CURRENT_DATE),
    observaciones VARCHAR(500),
    FOREIGN KEY (id_cita) REFERENCES cita(id_cita)
);

CREATE TABLE medicamento (
    id_medicamento INT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    descripcion VARCHAR(200),
```

```
    laboratorio VARCHAR(100)
);

-- Tabla receta_medicamento (tabla de relación)
CREATE TABLE receta_medicamento (
    id_receta INT NOT NULL,
    id_medicamento INT NOT NULL,
    dosis VARCHAR(50) NOT NULL,
    frecuencia VARCHAR(100) NOT NULL,
    duracion VARCHAR(50) NOT NULL,
    PRIMARY KEY (id_receta, id_medicamento),
    FOREIGN KEY (id_receta) REFERENCES receta(id_receta),
    FOREIGN KEY (id_medicamento) REFERENCES medicamento(id_medicamento)
);

CREATE TABLE bitacora (
    id_bitacora INT AUTO_INCREMENT PRIMARY KEY,
    tabla VARCHAR(50) NOT NULL,
    operacion VARCHAR(10) NOT NULL CHECK (operacion IN ('INSERT',
'UPDATE', 'DELETE')),
    id_registro INT NOT NULL,
    datos TEXT,
    origen VARCHAR(10),
    replicado TINYINT(1) NOT NULL,
    intentos INT,
    ultimo_error TEXT
);
```

Procedimientos almacenados, disparadores y vistas

Procedimientos en Oracle

```
CREATE OR REPLACE
PROCEDURE aplicar_json_replica (
   p_tabla       VARCHAR2,
   p_operacion   VARCHAR2,
   p_id_registro NUMBER,
   p_datos       VARCHAR2
) IS
BEGIN
   -- PACIENTE
   IF p_tabla = 'paciente' THEN
      IF p_operacion = 'INSERT' THEN
         INSERT INTO paciente (id_paciente, nombre, fecha_nacimiento, telefono,
email, genero)
         SELECT id_paciente, nombre, TO_DATE(fecha_nacimiento,
'YYYY-MM-DD'), telefono, email, genero
         FROM JSON_TABLE(p_datos, '$'
            COLUMNS (
```

```
            id_paciente NUMBER PATH '$.id_paciente',
            nombre VARCHAR2(100) PATH '$.nombre',
            fecha_nacimiento VARCHAR2(20) PATH '$.fecha_nacimiento',
            telefono VARCHAR2(20) PATH '$.telefono',
            email VARCHAR2(100) PATH '$.email',
            genero VARCHAR2(10) PATH '$.genero'
        )
    );

ELSIF p_operacion = 'UPDATE' THEN
    UPDATE paciente SET
        nombre = (SELECT nombre FROM JSON_TABLE(p_datos, '$' COLUMNS
(nombre VARCHAR2(100) PATH '$.nombre'))),
        fecha_nacimiento = (SELECT TO_DATE(fecha_nacimiento,
'YYYY-MM-DD') FROM JSON_TABLE(p_datos, '$' COLUMNS (fecha_nacimiento
VARCHAR2(20) PATH '$.fecha_nacimiento'))),
        telefono = (SELECT telefono FROM JSON_TABLE(p_datos, '$' COLUMNS
(telefono VARCHAR2(20) PATH '$.telefono'))),
        email = (SELECT email FROM JSON_TABLE(p_datos, '$' COLUMNS
(email VARCHAR2(100) PATH '$.email'))),
```

```
            genero = (SELECT genero FROM JSON_TABLE(p_datos, '$' COLUMNS
(genero VARCHAR2(10) PATH '$.genero')))
        WHERE id_paciente = p_id_registro;

    ELSIF p_operacion = 'DELETE' THEN
        DELETE FROM paciente WHERE id_paciente = p_id_registro;
    END IF;

-- CLINICA
ELSIF p_tabla = 'clinica' THEN
    IF p_operacion = 'INSERT' THEN
        INSERT INTO clinica (id_clinica, nombre, ciudad, direccion)
        SELECT id_clinica, nombre, ciudad, direccion
        FROM JSON_TABLE(p_datos, '$'
            COLUMNS (
                id_clinica NUMBER PATH '$.id_clinica',
                nombre VARCHAR2(100) PATH '$.nombre',
                ciudad VARCHAR2(100) PATH '$.ciudad',
                direccion VARCHAR2(200) PATH '$.direccion'
```

```
            )
        );

    ELSIF p_operacion = 'UPDATE' THEN
        UPDATE clinica SET
            nombre = (SELECT nombre FROM JSON_TABLE(p_datos, '$' COLUMNS
(nombre VARCHAR2(100) PATH '$.nombre'))),
            ciudad = (SELECT ciudad FROM JSON_TABLE(p_datos, '$' COLUMNS
(ciudad VARCHAR2(100) PATH '$.ciudad'))),
            direccion = (SELECT direccion FROM JSON_TABLE(p_datos, '$'
COLUMNS (direccion VARCHAR2(200) PATH '$.direccion')))
        WHERE id_clinica = p_id_registro;

    ELSIF p_operacion = 'DELETE' THEN
        DELETE FROM clinica WHERE id_clinica = p_id_registro;
    END IF;

-- DOCTOR
ELSIF p_tabla = 'doctor' THEN
    IF p_operacion = 'INSERT' THEN
```

```
        INSERT INTO doctor (id_doctor, nombre, especialidad, id_clinica)
        SELECT id_doctor, nombre, especialidad, id_clinica
        FROM JSON_TABLE(p_datos, '$'
          COLUMNS (
            id_doctor NUMBER PATH '$.id_doctor',
            nombre VARCHAR2(100) PATH '$.nombre',
            especialidad VARCHAR2(100) PATH '$.especialidad',
            id_clinica NUMBER PATH '$.id_clinica'
          )
        );

    ELSIF p_operacion = 'UPDATE' THEN
        UPDATE doctor SET
          nombre = (SELECT nombre FROM JSON_TABLE(p_datos, '$' COLUMNS
(nombre VARCHAR2(100) PATH '$.nombre'))),
          especialidad = (SELECT especialidad FROM JSON_TABLE(p_datos, '$'
COLUMNS (especialidad VARCHAR2(100) PATH '$.especialidad'))),
          id_clinica = (SELECT id_clinica FROM JSON_TABLE(p_datos, '$'
COLUMNS (id_clinica NUMBER PATH '$.id_clinica')))
        WHERE id_doctor = p_id_registro;
```

```
    ELSIF p_operacion = 'DELETE' THEN
        DELETE FROM doctor WHERE id_doctor = p_id_registro;
    END IF;

  -- CITA
  ELSIF p_tabla = 'cita' THEN
    IF p_operacion = 'INSERT' THEN
        INSERT INTO cita (id_cita, id_paciente, id_doctor, fecha_cita, motivo, estado)
        SELECT id_cita, id_paciente, id_doctor, TO_DATE(fecha_cita,
'YYYY-MM-DD'), motivo, estado
        FROM JSON_TABLE(p_datos, '$'
            COLUMNS (
                id_cita NUMBER PATH '$.id_cita',
                id_paciente NUMBER PATH '$.id_paciente',
                id_doctor NUMBER PATH '$.id_doctor',
                fecha_cita VARCHAR2(20) PATH '$.fecha_cita',
                motivo VARCHAR2(200) PATH '$.motivo',
                estado VARCHAR2(50) PATH '$.estado'
```

```
            )
        );

    ELSIF p_operacion = 'UPDATE' THEN
        UPDATE cita SET
            id_paciente = (SELECT id_paciente FROM JSON_TABLE(p_datos, '$'
COLUMNS (id_paciente NUMBER PATH '$.id_paciente'))),
            id_doctor = (SELECT id_doctor FROM JSON_TABLE(p_datos, '$'
COLUMNS (id_doctor NUMBER PATH '$.id_doctor'))),
            fecha_cita = (SELECT TO_DATE(fecha_cita, 'YYYY-MM-DD') FROM
JSON_TABLE(p_datos, '$' COLUMNS (fecha_cita VARCHAR2(20) PATH
'$.fecha_cita'))),
            motivo = (SELECT motivo FROM JSON_TABLE(p_datos, '$' COLUMNS
(motivo VARCHAR2(200) PATH '$.motivo'))),
            estado = (SELECT estado FROM JSON_TABLE(p_datos, '$' COLUMNS
(estado VARCHAR2(50) PATH '$.estado')))
        WHERE id_cita = p_id_registro;

    ELSIF p_operacion = 'DELETE' THEN
        DELETE FROM cita WHERE id_cita = p_id_registro;
```

```
    END IF;


  -- HISTORIAL_MEDICO
  ELSIF p_tabla = 'historial_medico' THEN
    IF p_operacion = 'INSERT' THEN
      INSERT INTO historial_medico (id_historial, id_paciente, descripcion,
fecha_registro)
      SELECT id_historial, id_paciente, descripcion, TO_DATE(fecha_registro,
'YYYY-MM-DD')
      FROM JSON_TABLE(p_datos, '$'
        COLUMNS (
          id_historial NUMBER PATH '$.id_historial',
          id_paciente NUMBER PATH '$.id_paciente',
          descripcion CLOB PATH '$.descripcion',
          fecha_registro VARCHAR2(20) PATH '$.fecha_registro'
        )
      );


    ELSIF p_operacion = 'UPDATE' THEN
```

```
        UPDATE historial_medico SET
            id_paciente = (SELECT id_paciente FROM JSON_TABLE(p_datos, '$' COLUMNS (id_paciente NUMBER PATH '$.id_paciente'))),
            descripcion = (SELECT descripcion FROM JSON_TABLE(p_datos, '$' COLUMNS (descripcion CLOB PATH '$.descripcion'))),
            fecha_registro = (SELECT TO_DATE(fecha_registro, 'YYYY-MM-DD') FROM JSON_TABLE(p_datos, '$' COLUMNS (fecha_registro VARCHAR2(20) PATH '$.fecha_registro')))
        WHERE id_historial = p_id_registro;

    ELSIF p_operacion = 'DELETE' THEN
        DELETE FROM historial_medico WHERE id_historial = p_id_registro;
    END IF;

  -- RECETA
  ELSIF p_tabla = 'receta' THEN
    IF p_operacion = 'INSERT' THEN
        INSERT INTO receta (id_receta, id_cita, fecha_emision, observaciones)
        SELECT id_receta, id_cita, TO_DATE(fecha_emision, 'YYYY-MM-DD'), observaciones
```

```
        FROM JSON_TABLE(p_datos, '$'
            COLUMNS (
                id_receta NUMBER PATH '$.id_receta',
                id_cita NUMBER PATH '$.id_cita',
                fecha_emision VARCHAR2(20) PATH '$.fecha_emision',
                observaciones CLOB PATH '$.observaciones'
            )
        );

    ELSIF p_operacion = 'UPDATE' THEN
        UPDATE receta SET
            id_cita = (SELECT id_cita FROM JSON_TABLE(p_datos, '$' COLUMNS
(id_cita NUMBER PATH '$.id_cita'))),
            fecha_emision = (SELECT TO_DATE(fecha_emision, 'YYYY-MM-DD')
FROM JSON_TABLE(p_datos, '$' COLUMNS (fecha_emision VARCHAR2(20) PATH
'$.fecha_emision'))),
            observaciones = (SELECT observaciones FROM JSON_TABLE(p_datos, '$'
COLUMNS (observaciones CLOB PATH '$.observaciones')))
        WHERE id_receta = p_id_registro;
```

```
    ELSIF p_operacion = 'DELETE' THEN
        DELETE FROM receta WHERE id_receta = p_id_registro;
    END IF;

  -- MEDICAMENTO
  ELSIF p_tabla = 'medicamento' THEN
    IF p_operacion = 'INSERT' THEN
        INSERT INTO medicamento (id_medicamento, nombre, descripcion,
laboratorio)
        SELECT id_medicamento, nombre, descripcion, laboratorio
        FROM JSON_TABLE(p_datos, '$'
          COLUMNS (
            id_medicamento NUMBER PATH '$.id_medicamento',
            nombre VARCHAR2(100) PATH '$.nombre',
            descripcion CLOB PATH '$.descripcion',
            laboratorio VARCHAR2(100) PATH '$.laboratorio'
          )
        );
```

```
        ELSIF p_operacion = 'UPDATE' THEN
            UPDATE medicamento SET
                nombre = (SELECT nombre FROM JSON_TABLE(p_datos, '$' COLUMNS
(nombre VARCHAR2(100) PATH '$.nombre'))),
                descripcion = (SELECT descripcion FROM JSON_TABLE(p_datos, '$'
COLUMNS (descripcion CLOB PATH '$.descripcion'))),
                laboratorio = (SELECT laboratorio FROM JSON_TABLE(p_datos, '$'
COLUMNS (laboratorio VARCHAR2(100) PATH '$.laboratorio')))
            WHERE id_medicamento = p_id_registro;

        ELSIF p_operacion = 'DELETE' THEN
            DELETE FROM medicamento WHERE id_medicamento = p_id_registro;
        END IF;

    -- RECETA_MEDICAMENTO
    ELSIF p_tabla = 'receta_medicamento' THEN
        IF p_operacion = 'INSERT' THEN
            INSERT INTO receta_medicamento (id_receta, id_medicamento, dosis,
frecuencia, duracion)
            SELECT id_receta, id_medicamento, dosis, frecuencia, duracion
```

```
        FROM JSON_TABLE(p_datos, '$'
            COLUMNS (
                id_receta NUMBER PATH '$.id_receta',
                id_medicamento NUMBER PATH '$.id_medicamento',
                dosis VARCHAR2(50) PATH '$.dosis',
                frecuencia VARCHAR2(50) PATH '$.frecuencia',
                duracion VARCHAR2(50) PATH '$.duracion'
            )
        );
    ELSIF p_operacion = 'UPDATE' THEN
        UPDATE receta_medicamento SET
            dosis = (SELECT dosis FROM JSON_TABLE(p_datos, '$' COLUMNS (dosis
VARCHAR2(50) PATH '$.dosis'))),
            frecuencia = (SELECT frecuencia FROM JSON_TABLE(p_datos, '$'
COLUMNS (frecuencia VARCHAR2(50) PATH '$.frecuencia'))),
            duracion = (SELECT duracion FROM JSON_TABLE(p_datos, '$'
COLUMNS (duracion VARCHAR2(50) PATH '$.duracion')))
        WHERE id_receta = (SELECT id_receta FROM JSON_TABLE(p_datos, '$'
COLUMNS (id_receta NUMBER PATH '$.id_receta')))
```

```
            AND id_medicamento = (SELECT id_medicamento FROM
JSON_TABLE(p_datos, '$' COLUMNS (id_medicamento NUMBER PATH
'$.id_medicamento')));

        ELSIF p_operacion = 'DELETE' THEN

            DELETE FROM receta_medicamento

            WHERE id_receta = (SELECT id_receta FROM JSON_TABLE(p_datos, '$'
COLUMNS (id_receta NUMBER PATH '$.id_receta')))

            AND id_medicamento = (SELECT id_medicamento FROM
JSON_TABLE(p_datos, '$' COLUMNS (id_medicamento NUMBER PATH
'$.id_medicamento')));

        END IF;

    END IF;

    COMMIT;

EXCEPTION

    WHEN OTHERS THEN

        ROLLBACK;

        RAISE;

END aplicar_json_replica;
```

```
CREATE OR REPLACE

PROCEDURE replicar_mysql_a_oracle IS

  CURSOR c_bitacora_mysql IS

    SELECT id_bitacora, tabla, operacion, id_registro, datos

    FROM bitacora@mysqlcli

    WHERE replicado = 0 AND origen = 'MYSQL';



BEGIN

 replicacion_ctx.replicando := TRUE;



  FOR rec IN c_bitacora_mysql LOOP

    BEGIN

      aplicar_json_replica(

        rec.tabla, rec.operacion, rec.id_registro, rec.datos

      );



      marcar_replicado_mysql(rec.id_bitacora);



    EXCEPTION WHEN OTHERS THEN
```

```
      DECLARE
        v_error VARCHAR2(4000);
      BEGIN
        v_error := SQLERRM;
        UPDATE bitacora@mysqlcli
          SET intentos = intentos + 1
        WHERE id_bitacora = rec.id_bitacora;


    END;
    END;
  END LOOP;


  COMMIT;
   replicacion_ctx.replicando := FALSE;


END;
```

```
CREATE OR REPLACE

PROCEDURE insertar_en_bitacora_mysql(

   p_tabla       VARCHAR2,

   p_operacion   VARCHAR2,

   p_id_registro NUMBER,

   p_datos       VARCHAR2,

   p_origen      VARCHAR2

) IS

  PRAGMA AUTONOMOUS_TRANSACTION;

BEGIN

  INSERT INTO bitacora@mysqlcli (

     tabla,

     operacion,

     id_registro,

     datos,

     origen,

     replicado,

     intentos

  )
```

```
  VALUES (
      p_tabla,
      p_operacion,
      p_id_registro,
      p_datos,
      p_origen,
      0,
      0
  );

  COMMIT;
END;
```

Triggers

```
-- AFTER INSERT, UPDATE, DELETE para cada tabla
-- Checan IF replicacion_ctx.replicando THEN RETURN; END IF;
-- Generan entrada en bitacora en operaciones de usuario
-- TRIGGER PARA PACIENTE
DELIMITER //
CREATE TRIGGER tr_paciente_validar_insert
BEFORE INSERT ON paciente
```

```
FOR EACH ROW
BEGIN
    DECLARE v_count INT;
-- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;


    -- Validar que la PK no sea NULL
    IF NEW.id_paciente IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: El id_paciente no puede ser NULL.';
    END IF;
    -- Validar que la PK no exista ya
    SELECT COUNT(*) INTO v_count
    FROM paciente
    WHERE id_paciente = NEW.id_paciente;
    IF v_count > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: El id_paciente ya existe.';
    END IF;

END//
DELIMITER ;
```

```
DELIMITER //
CREATE TRIGGER tr_paciente_insert
AFTER INSERT ON paciente
FOR EACH ROW
BEGIN
    DECLARE v_json JSON;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger
(modo replicador activado)';
  END IF;


    SET v_json = JSON_OBJECT(
        'id_paciente', NEW.id_paciente,
        'nombre', NEW.nombre,
        'fecha_nacimiento', DATE_FORMAT(NEW.fecha_nacimiento, '%Y-%m-%d'),
        'telefono', NEW.telefono,
        'email', NEW.email,
        'genero', NEW.genero
    );


    -- Registrar en bitácora
    INSERT INTO bitacora (tabla, operacion, id_registro, datos,
        origen, replicado, intentos, ultimo_error
) VALUES ('paciente', 'INSERT', NEW.id_paciente, v_json,
```

```
        'MYSQL', 0, 0, NULL
    );
END//
DELIMITER ;


DELIMITER //
CREATE TRIGGER tr_paciente_validar_update
BEFORE UPDATE ON paciente
FOR EACH ROW
BEGIN
-- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    -- Validar que no se modifique la PK
    IF NEW.id_paciente != OLD.id_paciente THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: No se permite modificar el id_paciente.';
    END IF;
END//
DELIMITER ;
DELIMITER //
CREATE TRIGGER tr_paciente_update
AFTER UPDATE ON paciente
```

```
FOR EACH ROW
BEGIN
    DECLARE v_json JSON;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    SET v_json = JSON_OBJECT(
        'id_paciente', NEW.id_paciente,
        'nombre', NEW.nombre,
        'fecha_nacimiento', DATE_FORMAT(NEW.fecha_nacimiento, '%Y-%m-%d'),
        'telefono', NEW.telefono,
        'email', NEW.email,
        'genero', NEW.genero,
        'campos_actualizados', JSON_ARRAY(
            IF(OLD.nombre != NEW.nombre, 'nombre', NULL),
            IF(DATE_FORMAT(OLD.fecha_nacimiento, '%Y-%m-%d') != DATE_FORMAT(NEW.fecha_nacimiento, '%Y-%m-%d'), 'fecha_nacimiento', NULL),
            IF(OLD.telefono != NEW.telefono, 'telefono', NULL),
            IF(OLD.email != NEW.email, 'email', NULL),
            IF(OLD.genero != NEW.genero, 'genero', NULL)
        )
    );
    INSERT INTO bitacora (
        tabla, operacion, id_registro, datos,
```

```
        origen, replicado, intentos, ultimo_error
    ) VALUES (
        'paciente', 'UPDATE', NEW.id_paciente, v_json,
        'MYSQL', 0, 0, NULL
    );
END//
DELIMITER ;


DELIMITER //
CREATE TRIGGER tr_paciente_validar_delete
BEFORE DELETE ON paciente
FOR EACH ROW
BEGIN
    DECLARE v_citas INT;
    DECLARE v_historial INT;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger
(modo replicador activado)';
  END IF;
    -- Verificar si el paciente tiene citas
    SELECT COUNT(*) INTO v_citas
    FROM cita
    WHERE id_paciente = OLD.id_paciente;
    -- Verificar si el paciente tiene historial médico
    SELECT COUNT(*) INTO v_historial
```

```
    FROM historial_medico
    WHERE id_paciente = OLD.id_paciente;
    IF v_citas > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No se puede eliminar el paciente: tiene citas
registradas.';
    END IF;
    IF v_historial > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No se puede eliminar el paciente: tiene historial médico
registrado.';
    END IF;
END;//
DELIMITER ;
DELIMITER //
CREATE TRIGGER tr_paciente_delete
AFTER DELETE ON paciente
FOR EACH ROW
BEGIN
    DECLARE v_json JSON;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger
(modo replicador activado)';
  END IF;
    SET v_json = JSON_OBJECT(
        'id_paciente', OLD.id_paciente,
```

```
        'nombre', OLD.nombre,
        'fecha_nacimiento', DATE_FORMAT(OLD.fecha_nacimiento, '%Y-%m-%d'),
        'telefono', OLD.telefono,
        'email', OLD.email,
        'genero', OLD.genero
    );
    INSERT INTO bitacora (
        tabla, operacion, id_registro, datos,
        origen, replicado, intentos, ultimo_error
    ) VALUES (
        'paciente', 'DELETE', OLD.id_paciente, v_json,
        'MYSQL', 0, 0, NULL
    );
END//
DELIMITER ;

-- TRIGGERS PARA CLINICA
DELIMITER //
CREATE TRIGGER tr_clinica_validar_insert
BEFORE INSERT ON clinica
FOR EACH ROW
BEGIN
    DECLARE v_count INT;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;

  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
```

```
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    -- Validar que la PK no sea NULL
    IF NEW.id_clinica IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: El id_clinica no puede ser NULL.';
    END IF;
    -- Validar que la PK no exista ya
    SELECT COUNT(*) INTO v_count
    FROM clinica
    WHERE id_clinica = NEW.id_clinica;

    IF v_count > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: El id_clinica ya existe.';
    END IF;
END;//
DELIMITER ;
DELIMITER //
CREATE TRIGGER tr_clinica_insert
AFTER INSERT ON clinica
FOR EACH ROW
BEGIN
    DECLARE v_json JSON;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;

  -- Si replicando está activo, salimos del trigger
```

```
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    SET v_json = JSON_OBJECT(
        'id_clinica', NEW.id_clinica,
        'nombre', NEW.nombre,
        'ciudad', NEW.ciudad,
        'direccion', NEW.direccion
    );
    INSERT INTO bitacora (
        tabla, operacion, id_registro, datos,
        origen, replicado, intentos, ultimo_error
    ) VALUES (
        'clinica', 'INSERT', NEW.id_clinica, v_json,
        'MYSQL', 0, 0, NULL
    );
END;//
DELIMITER ;

DELIMITER //
CREATE TRIGGER tr_clinica_validar_update
BEFORE UPDATE ON clinica
FOR EACH ROW
BEGIN
-- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;

  -- Si replicando está activo, salimos del trigger
```

```
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    -- Validar que no se modifique la PK
    IF NEW.id_clinica != OLD.id_clinica THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: No se permite modificar el id_clinica.';
    END IF;
END;//
DELIMITER ;
DELIMITER //
CREATE TRIGGER tr_clinica_update
AFTER UPDATE ON clinica
FOR EACH ROW
BEGIN
    DECLARE v_json JSON;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;

  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    SET v_json = JSON_OBJECT(
        'id_clinica', NEW.id_clinica,
        'nombre', NEW.nombre,
        'ciudad', NEW.ciudad,
```

```
        'direccion', NEW.direccion,
        'campos_actualizados', JSON_ARRAY(
            IF(OLD.nombre != NEW.nombre, 'nombre', NULL),
            IF(OLD.ciudad != NEW.ciudad, 'ciudad', NULL),
            IF(OLD.direccion != NEW.direccion, 'direccion', NULL)
        )
    );
    INSERT INTO bitacora (
        tabla, operacion, id_registro, datos,
        origen, replicado, intentos, ultimo_error
    ) VALUES (
        'clinica', 'UPDATE', NEW.id_clinica, v_json,
        'MYSQL', 0, 0, NULL
    );
END;//
DELIMITER ;

DELIMITER //
CREATE TRIGGER tr_clinica_validar_delete
BEFORE DELETE ON clinica
FOR EACH ROW
BEGIN
    DECLARE v_doctores INT;
    DECLARE v_citas INT;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;

  -- Si replicando está activo, salimos del trigger
```

```
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    -- Verificar si la clínica tiene doctores asociados
    SELECT COUNT(*) INTO v_doctores
    FROM doctor
    WHERE id_clinica = OLD.id_clinica;
    -- Verificar si la clínica tiene citas asociadas a través de doctores
    SELECT COUNT(*) INTO v_citas
    FROM cita c
    JOIN doctor d ON c.id_doctor = d.id_doctor
    WHERE d.id_clinica = OLD.id_clinica;
    IF v_doctores > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No se puede eliminar la clínica: tiene doctores asociados.';
    END IF;
    IF v_citas > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No se puede eliminar la clínica: tiene citas asociadas a sus doctores.';
    END IF;
END;//
DELIMITER ;
DELIMITER //
CREATE TRIGGER tr_clinica_delete
AFTER DELETE ON clinica
FOR EACH ROW
BEGIN
    DECLARE v_json JSON;
    -- Verificar bandera
```

```
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;

  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    SET v_json = JSON_OBJECT(
        'id_clinica', OLD.id_clinica,
        'nombre', OLD.nombre,
        'ciudad', OLD.ciudad,
        'direccion', OLD.direccion
    );
    INSERT INTO bitacora (
        tabla, operacion, id_registro, datos,
        origen, replicado, intentos, ultimo_error
    ) VALUES (
        'clinica', 'DELETE', OLD.id_clinica, v_json,
        'MYSQL', 0, 0, NULL
    );
END;//
DELIMITER ;

-- TRIGGERS PARA DOCTOR
DELIMITER //
CREATE TRIGGER tr_doctor_validar_insert
BEFORE INSERT ON doctor
FOR EACH ROW
BEGIN
```

```
    DECLARE v_count INT;
    DECLARE v_clinica_existente INT;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger
(modo replicador activado)';
  END IF;
    IF NEW.id_doctor IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: El id_doctor no puede ser NULL.';
    END IF;
    SELECT COUNT(*) INTO v_count
    FROM doctor
    WHERE id_doctor = NEW.id_doctor;
    IF v_count > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: El id_doctor ya existe.';
    END IF;
    SELECT COUNT(*) INTO v_clinica_existente
    FROM clinica
    WHERE id_clinica = NEW.id_clinica;
    IF v_clinica_existente = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: La clínica asignada no existe.';
    END IF;
END;//
```

```
DELIMITER ;
DELIMITER //
CREATE TRIGGER tr_doctor_insert
AFTER INSERT ON doctor
FOR EACH ROW
BEGIN
    DECLARE v_json JSON;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger
(modo replicador activado)';
  END IF;
    SET v_json = JSON_OBJECT(
        'id_doctor', NEW.id_doctor,
        'nombre', NEW.nombre,
        'especialidad', NEW.especialidad,
        'id_clinica', NEW.id_clinica
    );
    INSERT INTO bitacora (
        tabla, operacion, id_registro, datos,
        origen, replicado, intentos, ultimo_error
    ) VALUES (
        'doctor', 'INSERT', NEW.id_doctor, v_json,
        'MYSQL', 0, 0, NULL
    );
END;//
```

```
DELIMITER ;


DELIMITER //
CREATE TRIGGER tr_doctor_validar_update
BEFORE UPDATE ON doctor
FOR EACH ROW
BEGIN
        DECLARE v_clinica_existente INT;
   -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    IF NEW.id_doctor != OLD.id_doctor THEN
       SIGNAL SQLSTATE '45000'
       SET MESSAGE_TEXT = 'Error en validación: No se permite modificar el id_doctor.';
    END IF;
    -- Validar que la clínica referenciada exista
    SELECT COUNT(*) INTO v_clinica_existente
    FROM clinica
    WHERE id_clinica = NEW.id_clinica;
    IF v_clinica_existente = 0 THEN
       SIGNAL SQLSTATE '45000'
       SET MESSAGE_TEXT = 'Error en validación: La clínica asignada no existe.';
    END IF;
```

```
END;//
DELIMITER ;
DELIMITER //
CREATE TRIGGER tr_doctor_update
AFTER UPDATE ON doctor
FOR EACH ROW
BEGIN
    DECLARE v_json JSON;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    SET v_json = JSON_OBJECT(
        'id_doctor', NEW.id_doctor,
        'nombre', NEW.nombre,
        'especialidad', NEW.especialidad,
        'id_clinica', NEW.id_clinica,
        'campos_actualizados', JSON_ARRAY(
            IF(OLD.nombre != NEW.nombre, 'nombre', NULL),
            IF(OLD.especialidad != NEW.especialidad, 'especialidad', NULL),
            IF(OLD.id_clinica != NEW.id_clinica, 'id_clinica', NULL)
        )
    );
    INSERT INTO bitacora (
        tabla, operacion, id_registro, datos,
```

```
        origen, replicado, intentos, ultimo_error
    ) VALUES (
        'doctor', 'UPDATE', NEW.id_doctor, v_json,
        'MYSQL', 0, 0, NULL
    );
END;//
DELIMITER ;


DELIMITER //
CREATE TRIGGER tr_doctor_validar_delete
BEFORE DELETE ON doctor
FOR EACH ROW
BEGIN
    DECLARE v_citas INT;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    -- Validar que no existan citas asociadas al doctor
    SELECT COUNT(*) INTO v_citas
    FROM cita
    WHERE id_doctor = OLD.id_doctor;
    IF v_citas > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No se puede eliminar el doctor: tiene citas asociadas.';
```

```
    END IF;
END;//
DELIMITER ;
DELIMITER //
CREATE TRIGGER tr_doctor_delete
AFTER DELETE ON doctor
FOR EACH ROW
BEGIN
    DECLARE v_json JSON;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger
(modo replicador activado)';
  END IF;
    SET v_json = JSON_OBJECT(
        'id_doctor', OLD.id_doctor,
        'nombre', OLD.nombre,
        'especialidad', OLD.especialidad,
        'id_clinica', OLD.id_clinica
    );
    INSERT INTO bitacora (
        tabla, operacion, id_registro, datos,
        origen, replicado, intentos, ultimo_error
    ) VALUES (
        'doctor', 'DELETE', OLD.id_doctor, v_json,
        'MYSQL', 0, 0, NULL
```

```
    );
END;//
DELIMITER ;

-- TRIGGERS PARA CITA
DELIMITER //
CREATE TRIGGER tr_cita_validar_insert
BEFORE INSERT ON cita
FOR EACH ROW
BEGIN
    DECLARE v_count INT;
    DECLARE v_paciente INT;
    DECLARE v_doctor INT;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;

  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    IF NEW.id_cita IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: El id_cita no puede ser NULL.';
    END IF;
    SELECT COUNT(*) INTO v_count
    FROM cita
    WHERE id_cita = NEW.id_cita;
    IF v_count > 0 THEN
```

```
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: El id_cita ya existe.';
    END IF;
    SELECT COUNT(*) INTO v_paciente
    FROM paciente
    WHERE id_paciente = NEW.id_paciente;
    IF v_paciente = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: El paciente no existe.';
    END IF;
    SELECT COUNT(*) INTO v_doctor
    FROM doctor
    WHERE id_doctor = NEW.id_doctor;
    IF v_doctor = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: El doctor no existe.';
    END IF;
END;//
DELIMITER ;
DELIMITER //
CREATE TRIGGER tr_cita_insert
AFTER INSERT ON cita
FOR EACH ROW
BEGIN
    DECLARE v_json JSON;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;
```

```
  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    SET v_json = JSON_OBJECT(
        'id_cita', NEW.id_cita,
        'id_paciente', NEW.id_paciente,
        'id_doctor', NEW.id_doctor,
        'fecha_cita', DATE_FORMAT(NEW.fecha_cita, '%Y-%m-%d'),
        'motivo', NEW.motivo,
        'estado', NEW.estado
    );
    INSERT INTO bitacora (
        tabla, operacion, id_registro, datos,
        origen, replicado, intentos, ultimo_error
    ) VALUES (
        'cita', 'INSERT', NEW.id_cita, v_json,
        'MYSQL', 0, 0, NULL
    );
END;//
DELIMITER ;

DELIMITER //
CREATE TRIGGER tr_cita_validar_update
BEFORE UPDATE ON cita
FOR EACH ROW
BEGIN
    DECLARE v_paciente INT;
    DECLARE v_doctor INT;
    -- Verificar bandera
```

```
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    IF NEW.id_cita != OLD.id_cita THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: No se permite modificar el id_cita.';
    END IF;
    SELECT COUNT(*) INTO v_paciente
    FROM paciente
    WHERE id_paciente = NEW.id_paciente;
    IF v_paciente = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: El paciente no existe.';
    END IF;
    SELECT COUNT(*) INTO v_doctor
    FROM doctor
    WHERE id_doctor = NEW.id_doctor;
    IF v_doctor = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: El doctor no existe.';
    END IF;
END;//
DELIMITER ;
```

```
DELIMITER //
CREATE TRIGGER tr_cita_update
AFTER UPDATE ON cita
FOR EACH ROW
BEGIN
    DECLARE v_json JSON;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    SET v_json = JSON_OBJECT(
        'id_cita', NEW.id_cita,
        'id_paciente', NEW.id_paciente,
        'id_doctor', NEW.id_doctor,
        'fecha_cita', DATE_FORMAT(NEW.fecha_cita, '%Y-%m-%d'),
        'motivo', NEW.motivo,
        'estado', NEW.estado,
        'campos_actualizados', JSON_ARRAY(
            IF(OLD.id_paciente != NEW.id_paciente, 'id_paciente', NULL),
            IF(OLD.id_doctor != NEW.id_doctor, 'id_doctor', NULL),
            IF(DATE_FORMAT(OLD.fecha_cita, '%Y-%m-%d') != DATE_FORMAT(NEW.fecha_cita, '%Y-%m-%d'), 'fecha_cita', NULL),
            IF(OLD.motivo != NEW.motivo, 'motivo', NULL),
            IF(OLD.estado != NEW.estado, 'estado', NULL)
        )
```

```
    );
    INSERT INTO bitacora (
        tabla, operacion, id_registro, datos,
        origen, replicado, intentos, ultimo_error
    ) VALUES (
        'cita', 'UPDATE', NEW.id_cita, v_json,
        'MYSQL', 0, 0, NULL
    );
END;//
DELIMITER ;


DELIMITER //
CREATE TRIGGER tr_cita_validar_delete
BEFORE DELETE ON cita
FOR EACH ROW
BEGIN
    DECLARE v_recetas INT;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger
(modo replicador activado)';
  END IF;
    SELECT COUNT(*) INTO v_recetas
    FROM receta
    WHERE id_cita = OLD.id_cita;
    IF v_recetas > 0 THEN
```

```
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No se puede eliminar la cita: tiene recetas asociadas.';
    END IF;
END;//
DELIMITER ;
DELIMITER //
CREATE TRIGGER tr_cita_delete
AFTER DELETE ON cita
FOR EACH ROW
BEGIN
    DECLARE v_json JSON;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    SET v_json = JSON_OBJECT(
        'id_cita', OLD.id_cita,
        'id_paciente', OLD.id_paciente,
        'id_doctor', OLD.id_doctor,
        'fecha_cita', DATE_FORMAT(OLD.fecha_cita, '%Y-%m-%d'),
        'motivo', OLD.motivo,
        'estado', OLD.estado
    );
    INSERT INTO bitacora (
        tabla, operacion, id_registro, datos,
```

```
        origen, replicado, intentos, ultimo_error
    ) VALUES (
        'cita', 'DELETE', OLD.id_cita, v_json,
        'MYSQL', 0, 0, NULL
    );
END;//
DELIMITER ;

-- TRIGGERS PARA HISTORIAL MEDICO
DELIMITER //
CREATE TRIGGER tr_historial_validar_insert
BEFORE INSERT ON historial_medico
FOR EACH ROW
BEGIN
    DECLARE v_count INT;
    DECLARE v_paciente INT;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;

  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    IF NEW.id_historial IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: El id_historial no puede ser NULL.';
    END IF;
```

```
    SELECT COUNT(*) INTO v_count
    FROM historial_medico
    WHERE id_historial = NEW.id_historial;
    IF v_count > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: El id_historial ya existe.';
    END IF;
    SELECT COUNT(*) INTO v_paciente
    FROM paciente
    WHERE id_paciente = NEW.id_paciente;
    IF v_paciente = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: El paciente no existe.';
    END IF;
END;//
DELIMITER ;
DELIMITER //
CREATE TRIGGER tr_historial_insert
AFTER INSERT ON historial_medico
FOR EACH ROW
BEGIN
    DECLARE v_json JSON;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
```

```
  END IF;
    SET v_json = JSON_OBJECT(
        'id_historial', NEW.id_historial,
        'id_paciente', NEW.id_paciente,
        'descripcion', NEW.descripcion,
        'fecha_registro', DATE_FORMAT(NEW.fecha_registro, '%Y-%m-%d')
    );
    INSERT INTO bitacora (
        tabla, operacion, id_registro, datos,
        origen, replicado, intentos, ultimo_error
    ) VALUES (
        'historial_medico', 'INSERT', NEW.id_historial, v_json,
        'MYSQL', 0, 0, NULL
    );
END;//
DELIMITER ;

DELIMITER //
CREATE TRIGGER tr_historial_validar_update
BEFORE UPDATE ON historial_medico
FOR EACH ROW
BEGIN
DECLARE v_paciente INT;
-- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;

  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
```

```
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    IF NEW.id_historial != OLD.id_historial THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: No se permite modificar el id_historial.';
    END IF;
    SELECT COUNT(*) INTO v_paciente
    FROM paciente
    WHERE id_paciente = NEW.id_paciente;
    IF v_paciente = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: El paciente no existe.';
    END IF;
END;//
DELIMITER ;
DELIMITER //
CREATE TRIGGER tr_historial_update
AFTER UPDATE ON historial_medico
FOR EACH ROW
BEGIN
    DECLARE v_json JSON;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;

  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
```

```
  END IF;
    SET v_json = JSON_OBJECT(
        'id_historial', NEW.id_historial,
        'id_paciente', NEW.id_paciente,
        'descripcion', NEW.descripcion,
        'fecha_registro', DATE_FORMAT(NEW.fecha_registro, '%Y-%m-%d'),
        'campos_actualizados', JSON_ARRAY(
            IF(OLD.id_paciente != NEW.id_paciente, 'id_paciente', NULL),
            IF(OLD.descripcion != NEW.descripcion, 'descripcion', NULL),
            IF(DATE_FORMAT(OLD.fecha_registro, '%Y-%m-%d') !=
DATE_FORMAT(NEW.fecha_registro, '%Y-%m-%d'), 'fecha_registro', NULL)
        )
    );
    INSERT INTO bitacora (
        tabla, operacion, id_registro, datos,
        origen, replicado, intentos, ultimo_error
    ) VALUES (
        'historial_medico', 'UPDATE', NEW.id_historial, v_json,
        'MYSQL', 0, 0, NULL
    );
END;//
DELIMITER ;

DELIMITER //
CREATE TRIGGER tr_historial_delete
AFTER DELETE ON historial_medico
FOR EACH ROW
BEGIN
    DECLARE v_json JSON;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
```

```
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    SET v_json = JSON_OBJECT(
        'id_historial', OLD.id_historial,
        'id_paciente', OLD.id_paciente,
        'descripcion', OLD.descripcion,
        'fecha_registro', DATE_FORMAT(OLD.fecha_registro, '%Y-%m-%d')
    );
    INSERT INTO bitacora (
        tabla, operacion, id_registro, datos,
        origen, replicado, intentos, ultimo_error
    ) VALUES (
        'historial_medico', 'DELETE', OLD.id_historial, v_json,
        'MYSQL', 0, 0, NULL
    );
END;//
DELIMITER ;


-- TRIGGERS PARA RECETA
DELIMITER //
CREATE TRIGGER tr_receta_validar_insert
BEFORE INSERT ON receta
FOR EACH ROW
BEGIN
    DECLARE v_count INT;
```

```
    DECLARE v_cita INT;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    IF NEW.id_receta IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: El id_receta no puede ser NULL.';
    END IF;
    SELECT COUNT(*) INTO v_count
    FROM receta
    WHERE id_receta = NEW.id_receta;
    IF v_count > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: El id_receta ya existe.';
    END IF;
    SELECT COUNT(*) INTO v_cita
    FROM cita
    WHERE id_cita = NEW.id_cita;
    IF v_cita = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: La cita asignada no existe.';
    END IF;
END;//
DELIMITER ;
```

```
DELIMITER //
CREATE TRIGGER tr_receta_insert
AFTER INSERT ON receta
FOR EACH ROW
BEGIN
    DECLARE v_json JSON;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    SET v_json = JSON_OBJECT(
        'id_receta', NEW.id_receta,
        'id_cita', NEW.id_cita,
        'fecha_emision', DATE_FORMAT(NEW.fecha_emision, '%Y-%m-%d'),
        'observaciones', NEW.observaciones
    );
    INSERT INTO bitacora (
        tabla, operacion, id_registro, datos,
        origen, replicado, intentos, ultimo_error
    ) VALUES (
        'receta', 'INSERT', NEW.id_receta, v_json,
        'MYSQL', 0, 0, NULL
    );
END;//
DELIMITER ;
```

```
DELIMITER //
CREATE TRIGGER tr_receta_validar_update
BEFORE UPDATE ON receta
FOR EACH ROW
BEGIN
        DECLARE v_cita INT;
   -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    IF NEW.id_receta != OLD.id_receta THEN
       SIGNAL SQLSTATE '45000'
       SET MESSAGE_TEXT = 'Error en validación: No se permite modificar el id_receta.';
    END IF;
    SELECT COUNT(*) INTO v_cita
    FROM cita
    WHERE id_cita = NEW.id_cita;
    IF v_cita = 0 THEN
       SIGNAL SQLSTATE '45000'
       SET MESSAGE_TEXT = 'Error en validación: La cita asignada no existe.';
    END IF;
END;//
DELIMITER ;
```

```
DELIMITER //
CREATE TRIGGER tr_receta_update
AFTER UPDATE ON receta
FOR EACH ROW
BEGIN
    DECLARE v_json JSON;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    SET v_json = JSON_OBJECT(
        'id_receta', NEW.id_receta,
        'id_cita', NEW.id_cita,
        'fecha_emision', DATE_FORMAT(NEW.fecha_emision, '%Y-%m-%d'),
        'observaciones', NEW.observaciones,
        'campos_actualizados', JSON_ARRAY(
            IF(OLD.id_cita != NEW.id_cita, 'id_cita', NULL),
            IF(DATE_FORMAT(OLD.fecha_emision, '%Y-%m-%d') != DATE_FORMAT(NEW.fecha_emision, '%Y-%m-%d'), 'fecha_emision', NULL),
            IF(OLD.observaciones != NEW.observaciones, 'observaciones', NULL)
        )
    );
    INSERT INTO bitacora (
        tabla, operacion, id_registro, datos,
        origen, replicado, intentos, ultimo_error
```

```
    ) VALUES (
        'receta', 'UPDATE', NEW.id_receta, v_json,
        'MYSQL', 0, 0, NULL
    );
END;//
DELIMITER ;


DELIMITER //
CREATE TRIGGER tr_receta_validar_delete
BEFORE DELETE ON receta
FOR EACH ROW
BEGIN
    DECLARE v_med INT;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    SELECT COUNT(*) INTO v_med
    FROM receta_medicamento
    WHERE id_receta = OLD.id_receta;
    IF v_med > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No se puede eliminar la receta: tiene medicamentos asociados.';
    END IF;
```

```
END;//
DELIMITER ;
DELIMITER //
CREATE TRIGGER tr_receta_delete
AFTER DELETE ON receta
FOR EACH ROW
BEGIN
   DECLARE v_json JSON;
   -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger
(modo replicador activado)';
  END IF;
   SET v_json = JSON_OBJECT(
      'id_receta', OLD.id_receta,
      'id_cita', OLD.id_cita,
      'fecha_emision', DATE_FORMAT(OLD.fecha_emision, '%Y-%m-%d'),
      'observaciones', OLD.observaciones
   );
   INSERT INTO bitacora (
      tabla, operacion, id_registro, datos,
      origen, replicado, intentos, ultimo_error
   ) VALUES (
      'receta', 'DELETE', OLD.id_receta, v_json,
      'MYSQL', 0, 0, NULL
   );
```

```
END;//
DELIMITER ;


-- TRIGGERS PARA MEDICAMENTO
DELIMITER //
CREATE TRIGGER tr_medicamento_validar_insert
BEFORE INSERT ON medicamento
FOR EACH ROW
BEGIN
   DECLARE v_count INT;
   -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
   IF NEW.id_medicamento IS NULL THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Error en validación: El id_medicamento no puede ser NULL.';
   END IF;
   SELECT COUNT(*) INTO v_count
   FROM medicamento
   WHERE id_medicamento = NEW.id_medicamento;
   IF v_count > 0 THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Error en validación: El id_medicamento ya existe.';
```

```
    END IF;
END;//
DELIMITER ;
DELIMITER //
CREATE TRIGGER tr_medicamento_insert
AFTER INSERT ON medicamento
FOR EACH ROW
BEGIN
    DECLARE v_json JSON;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger
(modo replicador activado)';
  END IF;
    SET v_json = JSON_OBJECT(
        'id_medicamento', NEW.id_medicamento,
        'nombre', NEW.nombre,
        'descripcion', NEW.descripcion,
        'laboratorio', NEW.laboratorio
    );
    INSERT INTO bitacora (
        tabla, operacion, id_registro, datos,
        origen, replicado, intentos, ultimo_error
    ) VALUES (
        'medicamento', 'INSERT', NEW.id_medicamento, v_json,
        'MYSQL', 0, 0, NULL
```

```
    );
END;//
DELIMITER ;


DELIMITER //
CREATE TRIGGER tr_medicamento_validar_update
BEFORE UPDATE ON medicamento
FOR EACH ROW
BEGIN
-- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    IF NEW.id_medicamento != OLD.id_medicamento THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: No se permite modificar el id_medicamento.';
    END IF;
END;//
DELIMITER ;
DELIMITER //
CREATE TRIGGER tr_medicamento_update
AFTER UPDATE ON medicamento
FOR EACH ROW
BEGIN
```

```
    DECLARE v_json JSON;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger
(modo replicador activado)';
  END IF;
    SET v_json = JSON_OBJECT(
        'id_medicamento', NEW.id_medicamento,
        'nombre', NEW.nombre,
        'descripcion', NEW.descripcion,
        'laboratorio', NEW.laboratorio,
        'campos_actualizados', JSON_ARRAY(
            IF(OLD.nombre != NEW.nombre, 'nombre', NULL),
            IF(OLD.descripcion != NEW.descripcion, 'descripcion', NULL),
            IF(OLD.laboratorio != NEW.laboratorio, 'laboratorio', NULL)
        )
    );
    INSERT INTO bitacora (
        tabla, operacion, id_registro, datos,
        origen, replicado, intentos, ultimo_error
    ) VALUES (
        'medicamento', 'UPDATE', NEW.id_medicamento, v_json,
        'MYSQL', 0, 0, NULL
    );
END;//
DELIMITER ;
```

```
DELIMITER //
CREATE TRIGGER tr_medicamento_validar_delete
BEFORE DELETE ON medicamento
FOR EACH ROW
BEGIN
    DECLARE v_count INT;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    SELECT COUNT(*) INTO v_count
    FROM receta_medicamento
    WHERE id_medicamento = OLD.id_medicamento;
    IF v_count > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No se puede eliminar el medicamento: está asociado a una receta.';
    END IF;
END;//
DELIMITER ;
DELIMITER //
CREATE TRIGGER tr_medicamento_delete
AFTER DELETE ON medicamento
FOR EACH ROW
```

```
BEGIN
    DECLARE v_json JSON;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    SET v_json = JSON_OBJECT(
        'id_medicamento', OLD.id_medicamento,
        'nombre', OLD.nombre,
        'descripcion', OLD.descripcion,
        'laboratorio', OLD.laboratorio
    );
    INSERT INTO bitacora (
        tabla, operacion, id_registro, datos,
        origen, replicado, intentos, ultimo_error
    ) VALUES (
        'medicamento', 'DELETE', OLD.id_medicamento, v_json,
        'MYSQL', 0, 0, NULL
    );
END;//
DELIMITER ;

-- TRIGGERS RECETA MEDICAMENTO
DELIMITER //
CREATE TRIGGER tr_receta_medicamento_validar_insert
```

```
BEFORE INSERT ON receta_medicamento
FOR EACH ROW
BEGIN
    DECLARE v_receta INT;
    DECLARE v_med INT;
    DECLARE v_duplicado INT;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    IF NEW.id_receta IS NULL OR NEW.id_medicamento IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: La clave compuesta no puede tener valores NULL.';
    END IF;
    SELECT COUNT(*) INTO v_receta
    FROM receta
    WHERE id_receta = NEW.id_receta;
    IF v_receta = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: La receta no existe.';
    END IF;
    SELECT COUNT(*) INTO v_med
    FROM medicamento
    WHERE id_medicamento = NEW.id_medicamento;
```

```
    IF v_med = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: El medicamento no existe.';
    END IF;
    SELECT COUNT(*) INTO v_duplicado
    FROM receta_medicamento
    WHERE id_receta = NEW.id_receta AND id_medicamento = NEW.id_medicamento;
    IF v_duplicado > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: Ya existe esta combinación
receta-medicamento.';
    END IF;
END;//
DELIMITER ;
DELIMITER //
CREATE TRIGGER tr_receta_medicamento_insert
AFTER INSERT ON receta_medicamento
FOR EACH ROW
BEGIN
    DECLARE v_json JSON;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger
(modo replicador activado)';
  END IF;
    SET v_json = JSON_OBJECT(
```

```
        'id_receta', NEW.id_receta,
        'id_medicamento', NEW.id_medicamento,
        'dosis', NEW.dosis,
        'frecuencia', NEW.frecuencia,
        'duracion', NEW.duracion
    );
    INSERT INTO bitacora (
        tabla, operacion, id_registro, datos,
        origen, replicado, intentos, ultimo_error
    ) VALUES (
        'receta_medicamento',
        'INSERT',
        NULL,
        v_json,
        'MYSQL', 0, 0, NULL
    );
END;//
DELIMITER ;

DELIMITER //
CREATE TRIGGER tr_receta_medicamento_validar_update
BEFORE UPDATE ON receta_medicamento
FOR EACH ROW
BEGIN
-- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;

  -- Si replicando está activo, salimos del trigger
```

```
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    IF NEW.id_receta != OLD.id_receta OR NEW.id_medicamento != OLD.id_medicamento THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error en validación: No se permite modificar id_receta o id_medicamento.';
    END IF;
END;//
DELIMITER ;
DELIMITER //
CREATE TRIGGER tr_receta_medicamento_update
AFTER UPDATE ON receta_medicamento
FOR EACH ROW
BEGIN
    DECLARE v_json JSON;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;

  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    SET v_json = JSON_OBJECT(
        'id_receta', NEW.id_receta,
        'id_medicamento', NEW.id_medicamento,
        'dosis', NEW.dosis,
```

```
        'frecuencia', NEW.frecuencia,
        'duracion', NEW.duracion,
        'campos_actualizados', JSON_ARRAY(
            IF(OLD.dosis != NEW.dosis, 'dosis', NULL),
            IF(OLD.frecuencia != NEW.frecuencia, 'frecuencia', NULL),
            IF(OLD.duracion != NEW.duracion, 'duracion', NULL)
        )
    );
    INSERT INTO bitacora (
        tabla, operacion, id_registro, datos,
        origen, replicado, intentos, ultimo_error
    ) VALUES (
        'receta_medicamento',
        'UPDATE',
        NULL,
        v_json,
        'MYSQL', 0, 0, NULL
    );
END;//
DELIMITER ;

DELIMITER //
CREATE TRIGGER tr_receta_medicamento_delete
AFTER DELETE ON receta_medicamento
FOR EACH ROW
BEGIN
    DECLARE v_json JSON;
    -- Verificar bandera
DECLARE v_replicando TINYINT DEFAULT 0;
  -- Verificar bandera
```

```
  SELECT replicando INTO v_replicando FROM replicacion_control WHERE id = 1;


  -- Si replicando está activo, salimos del trigger
  IF v_replicando = 1 THEN
 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'no ejecutar nada en el trigger (modo replicador activado)';
  END IF;
    SET v_json = JSON_OBJECT(
        'id_receta', OLD.id_receta,
        'id_medicamento', OLD.id_medicamento,
        'dosis', OLD.dosis,
        'frecuencia', OLD.frecuencia,
        'duracion', OLD.duracion
    );
    INSERT INTO bitacora (
        tabla, operacion, id_registro, datos,
        origen, replicado, intentos, ultimo_error
    ) VALUES (
        'receta_medicamento',
        'DELETE',
        NULL,
        v_json,
        'MYSQL', 0, 0, NULL
    );
END;//
DELIMITER ;
```

Transacciones representativas con control de concurrencia

El control de concurrencia y de las transacciones fue manejado completamente por el SGBD integrado en las bases de datos. En caso de error, un rollback es probable.