SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNM TECNOLOGICO NACIONAL DE MEXICO

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de Morelia

"Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos"

INSTITUTO TECNOLÓGICO DE MORELIA

División De Estudios Profesionales

Departamento De Sistemas Y Computación

TALLER DE BASE DE DATOS

Unidad V: MYSQL PROCEDURAL

PRACTICA 1: TRIGGERS Y FUNCIONES

PROFESOR

I.S.C. Rubén Lara Barcenas

ALUMNOS

Diego Ulises Martínez Aguilar / Miguel Magdaleno Rosales

MORELIA, MICHOACÁN

Fecha de entrega: viernes, 25 de mayo del 2018

Tabla de contenido

Introducción

Como se vio anteriormente, cuando hablamos de las bases de datos debemos de incluir diversas operaciones a las que se pueden someter, siendo estas las básicas y principales para manipularlas de manera correcta. Sin embargo, existen otras operaciones diferentes las cuales sirven como complemento para las básicas, ya que se pueden encargar de realizar más de una operación al mismo tiempo o establecer diferentes parámetros para que se cumpla un requerimiento, así que en esta ocasión nos enfocaremos en estas.

Primero hay que hacer mención sobre un elemento importante que son los **triggers**. Los triggers o disparadores son objetos de una base de datos, los cuales están asociados con tablas. Este se puede ejecutar cuando se efectúa una de las operaciones básicas de una base de datos: INSERT, UPDATE, DELETE. Estos se pueden invocar antes o después de dicha operación, modificando así determinada información. A continuación veremos un ejemplo de la creación de un disparador:

```
. DELIMITER //

CREATE TRIGGER APTS_I  AFTER INSERT ON APUNTES

FOR EACH ROW
BEGIN

INSERT INTO SALDO SET
                SALDO.CUENTA=NEW.CUENTA,
                SALDO.ANO=YEAR(NEW.FECHA),
                SALDO.MES=MONTH(NEW.FECHA),
                SALDO.DEBE=NEW.DEBE,
                SALDO.HABER=NEW.HABER
         ON DUPLICATE KEY UPDATE
                SALDO.DEBE=SALDO.DEBE+NEW.DEBE,
                SALDO.HABER=SALDO.HABER+NEW.HABER ;

END;//

DELIMITER ;
```

El ejemplo anterior se trata sobre la actualización de diferentes datos **después** de que se haya realizado una inserción en la tabla de APUNTES. Con esto se puede observar uno de las diferentes aplicaciones que puede tomar un disparador.

Otro elemento importante de los que hay que hacer mención son los procedimientos almacenados.

Podemos definir un **procedimiento almacenado** podemos definirlo como un bloque de código en la que interviene la sentencia de código **CREATE PROCEDURE**, la cual tendrá el código (denominado cuerpo) del procedimiento almacenado.

Algunas de las situaciones en que los procedimientos almacenados pueden ser útiles son:

- Cuando múltiples aplicaciones cliente se escriben en distintos lenguajes o funcionan en distintas plataformas, pero necesitan realizar la misma operación en la base de datos.
- Cuando la seguridad es muy importante. Los bancos, por ejemplo, usan procedimientos almacenados para todas las operaciones comunes. Esto proporciona un entorno seguro y consistente, y los procedimientos pueden asegurar que cada operación se loguea apropiadamente.

Los procedimientos almacenados pueden mejorar el rendimiento ya que se necesita enviar menos información entre el servidor y el cliente.

```
mysql> delimiter //

mysql> CREATE PROCEDURE simpleproc (OUT param1 INT)
    -> BEGIN
    ->   SELECT COUNT(*) INTO param1 FROM t;
    -> END
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;

mysql> CALL simpleproc(@a);
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @a;
+------+
| @a   |
+------+
| 3    |
+------+
1 row in set (0.00 sec)
```

Por último, hay que mencionar otra de las características de MySQL, las cuales son las **funciones**. MySQL dispone de una gran cantidad de estas.

Las funciones de esta categoría son:

IF	Elección en función de una expresión booleana
IFNULL	Elección en función de si el valor de una expresión es *NULL*
NULLIF	Devuelve *NULL* en función del valor de una expresión

Algunas de las funciones para tratamiento de cadenas de caracteres son:

ASCII	Valor de código ASCII de un carácter
BIN	Conversión a binario
BIT_LENGTH	Cálculo de longitud de cadena en bits
CHAR	Convierte de ASCII a carácter
CHAR_LENGTH o CHARACTER_LENGTH	Cálculo de longitud de cadena en caracteres
COMPRESS	Comprime una cadena de caracteres
CONCAT	Concatena dos cadenas de caracteres
CONCAT_WS	Concatena cadenas con separadores
CONV	Convierte números entre distintas bases
ELT	Elección entre varias cadenas
EXPORT_SET	Expresiones binarias como conjuntos
FIELD	Busca el índice en listas de cadenas
FIND_IN_SET	Búsqueda en listas de cadenas
HEX	Conversión de números a hexadecimal
INSERT	Inserta una cadena en otra

"Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos"

Material

- MYSQL en Ubuntu Server
- phpMyAdmin
- JDBC
- IDE para JAVA

Contenido

En el desarrollo de esta práctica, se logró implementar de manera exitosa las nuevas operaciones propuestas por el profesor, con las cuales no estábamos muy familiarizados.

Cada una de las operaciones (disparadores, procedimientos almacenados y funciones) se llevaron a cabo desde nuestro manejador de la base de datos, en este caso **phpmyadmin**. Además de crearlas y ejecutarlas directamente desde nuestro sistema gestor, se implementaron en nuestra aplicación de escritorio para poder ver los resultados de manera más real y así sacar las conclusiones con un enfoque definido.

Desarrollo

Imagen 1 Creación de los procedimientos para inserción, modificación, eliminación y consultas

"Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos"

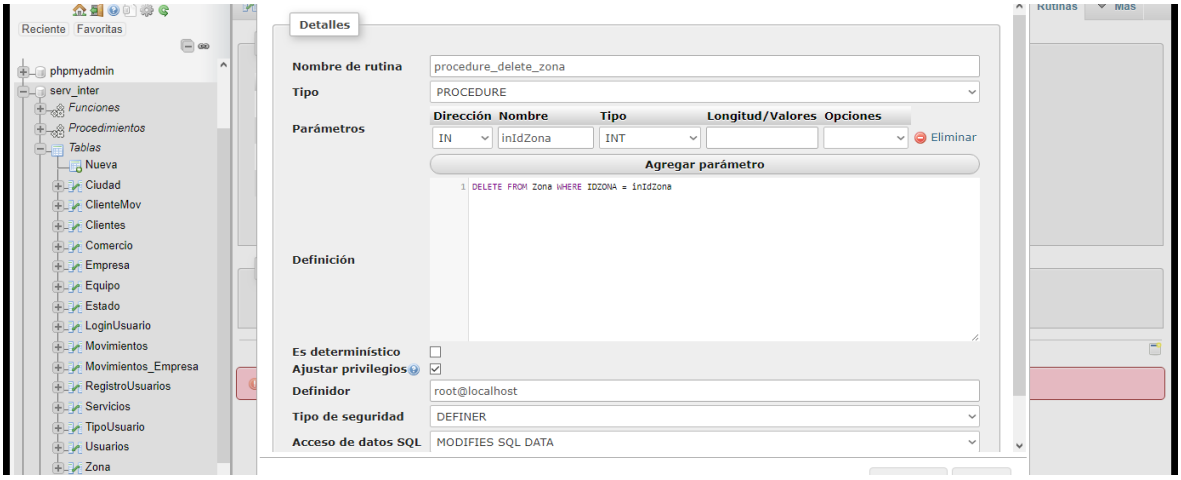

Imagen 2 **Procedimiento de inserción**

```
1469    private void procedureDeleteZona(int IDZONA) throws SQLException {
1470        cStmt = conexion.prepareCall("{call procedure_delete_zona (?)}");
1471        cStmt.setInt(1,IDZONA);
1472        cStmt.execute();
1473    }
```

Imagen 3 **Implementación del procedimiento de eliminación en el código de la aplicación de escritorio utilizando el método call**

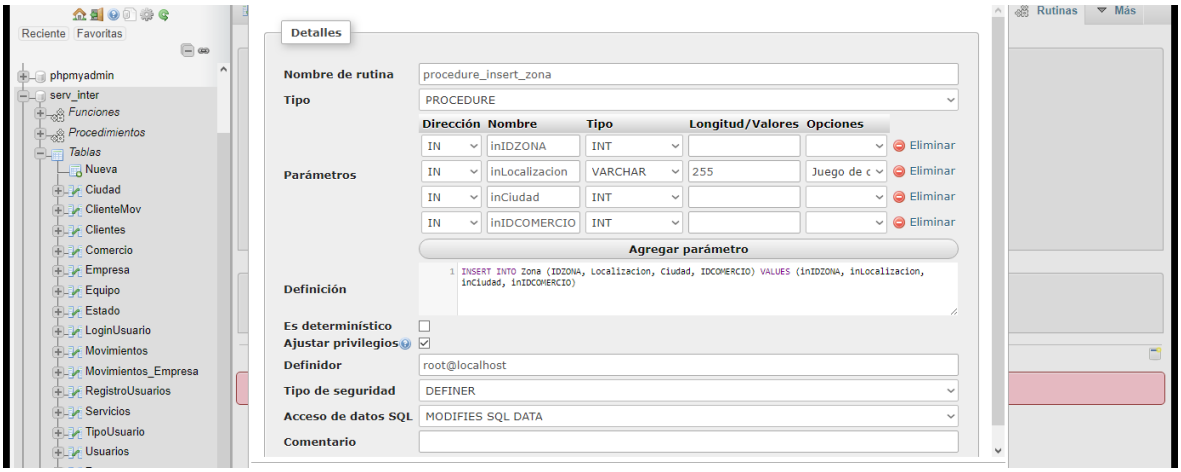

Imagen 4 **Creación del procedimiento de inserción desde el manejador con sus respectivos atributos**

"Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos"

```
1440        //Procedimientos Almacenados tabla zona
1441        CallableStatement cStmt;
1442
1443        private void procedureInsertZona(int IDZONA, String Localizacion, int Ciudad, int IDECOMERCIO) throws SQLException{
1444
1445            cStmt = conexion.prepareCall("{call procedure_insert_zona (?,?,?,?)}");
1446            cStmt.setInt(1,IDZONA);
1447            cStmt.setString(2, Localizacion);
1448            cStmt.setInt(3, Ciudad);
1449            cStmt.setInt(4,IDECOMERCIO);
1450    ////        cStmt.registerOutParameter("inOutParam", Types.INTEGER);
1451            cStmt.execute();
1452    ////        final ResultSet rs = cStmt.getResultSet();
1453    ////        cStmt.execute();
1454    ////        final ResultSet rs = cStmt.getResultSet();
1455    ////        while(rs.next()){
1456    ////            System.out.println("Cadena de caracteres pasada como prametro de entrada = "+rs.getString("inputParam"));
1457    ////        }
1458    ////        int outputValue = cStmt.getInt("inOutParam");
1459    ////        System.out.println("Parametro de salida incrementado = "+outputValue);
1460        }
```

Imagen 5 **Implementación del procedimiento de inserción en el código de la aplicación de escritorio**

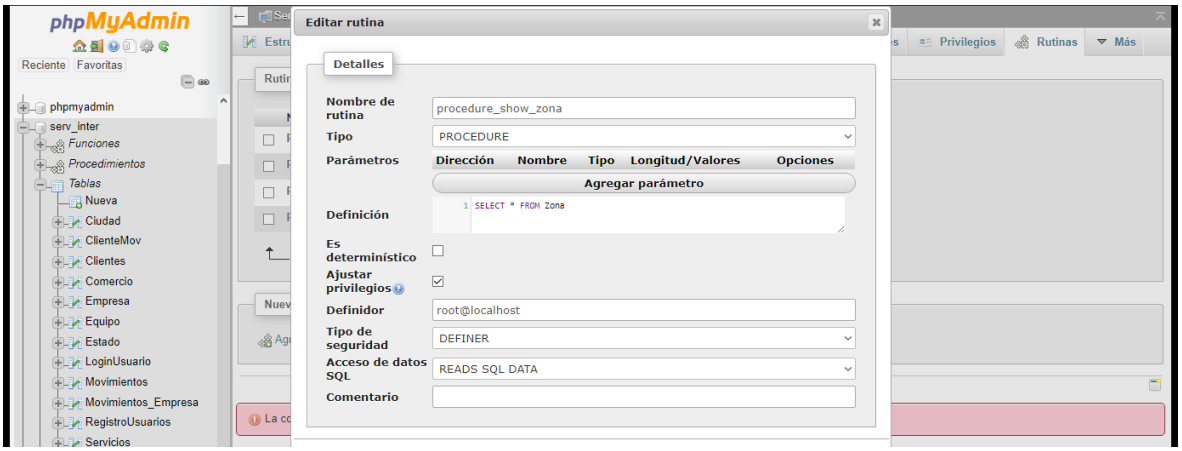

Imagen 6 **Creación del procedimiento de inserción en el manejador**

"Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos"

```
1479    private ArrayList<CLASEZONA> procedureShowZona(){
1480        try {
1481            cStmt = conexion.prepareCall("{call procedure_show_zona ()}");
1482            cStmt.execute();
1483            final ResultSet rs = cStmt.getResultSet();
1484            ArrayList<CLASEZONA> lista = new ArrayList<>();
1485            while (rs.next()) {
1486
1487                CLASEZONA obj = new CLASEZONA();
1488
1489                obj.setIDZONA(rs.getInt("IDZONA"));
1490                obj.setLocalizacion(rs.getString("Localizacion"));
1491                obj.setCiudad(rs.getInt("Ciudad"));
1492                obj.setIDComercio(rs.getInt("IDComercio"));
1493
1494                lista.add(obj);
1495            }
1496            rs.close();
1497            return lista;
1498        } catch (SQLException ex) {
1499            Logger.getLogger(Handler.class.getName()).log(Level.SEVERE, null, ex);
1500            return null;
1501        }
1502    }
1503 }
```

Imagen 7 **Implementación del procedimiento de consulta en el código de la aplicación**

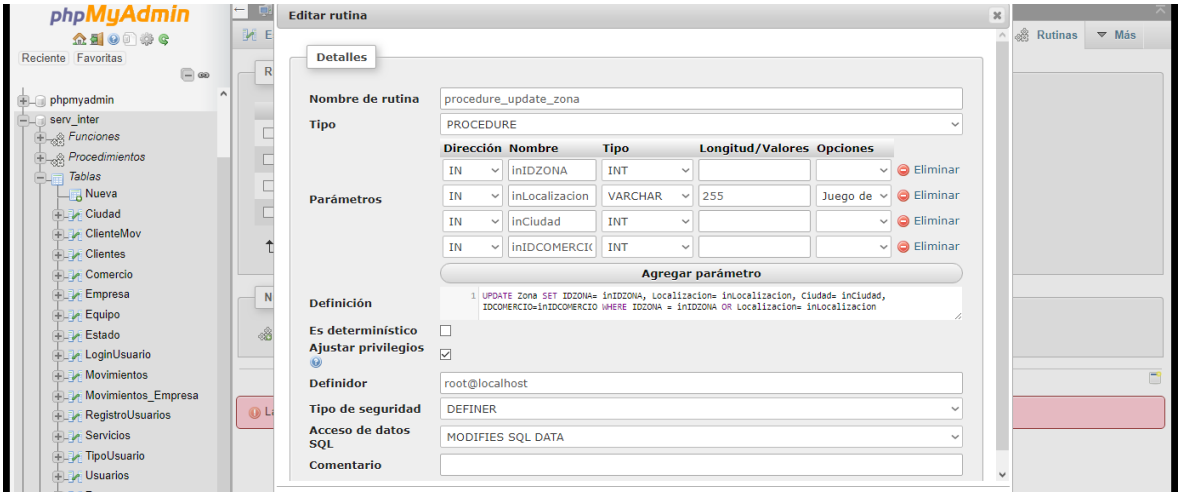

Imagen 8 **Creación del procedimiento de actualización en el manejador**

"Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos"

```
1468    private void procedureUpdateZona(int IDZONA, String Localizacion, int Ciudad, int IDECOMERCIO) throws SQLException {
1469        cStmt = conexion.prepareCall("{call procedure_update_zona (?,?,?,?)}");
1470
1471        cStmt.setInt(1,IDZONA);
1472        cStmt.setString(2, Localizacion);
1473        cStmt.setInt(3, Ciudad);
1474        cStmt.setInt(4,IDECOMERCIO);
1475
1476        cStmt.execute();
1477    }
```

Imagen 9 Implementación del procedimiento en el código de la aplicación

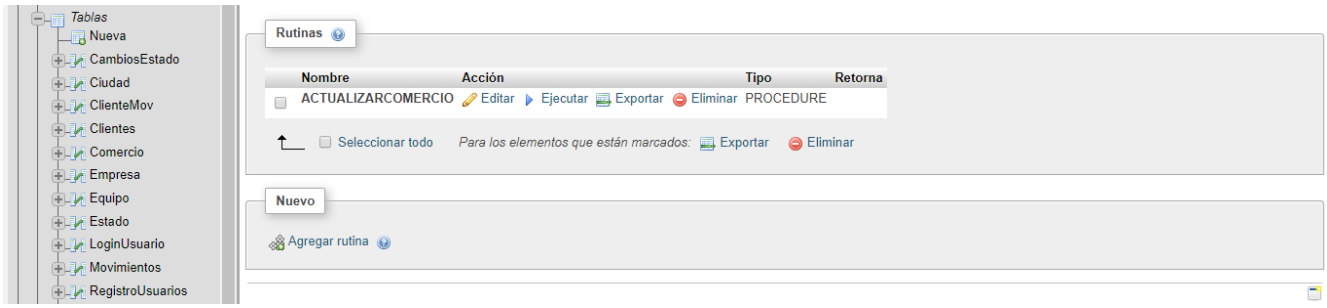

Imagen 10 Procedimiento utilizado para actualizar nombres comerciales

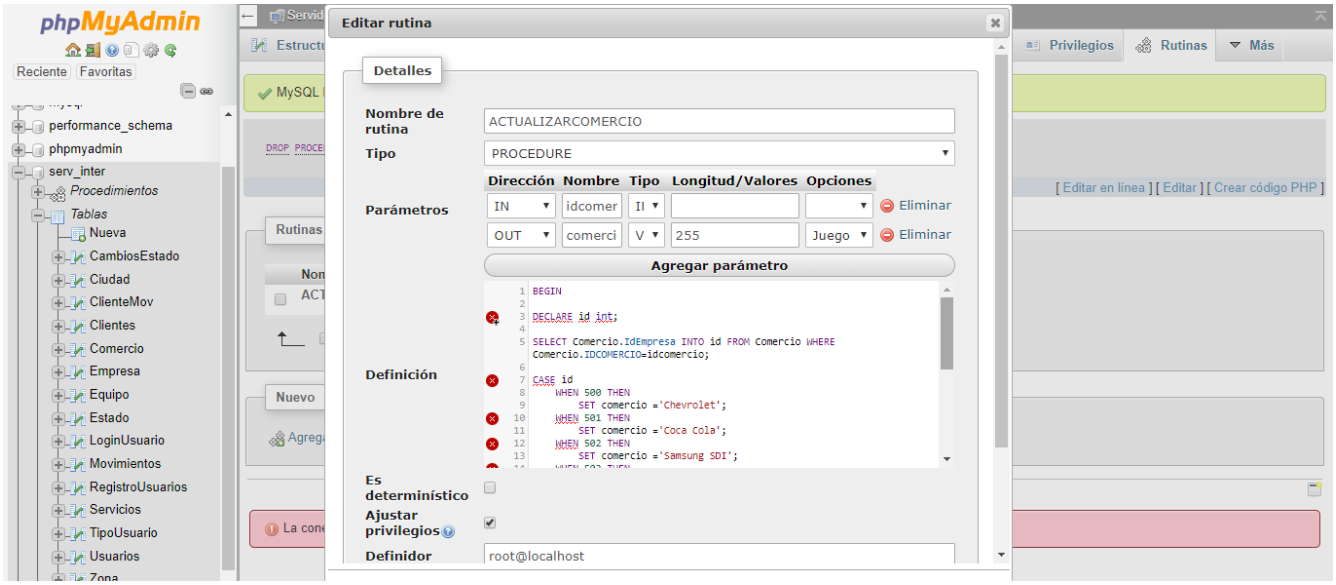

Imagen 11 Creación del procedimiento que cambiará el nombre comercial dependiendo del que se elimine con sus respectivos atributos

"Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos"

```
291
292     public void BajaComercio(int IdComercio, String comercio) {
293
294         try {
295             CallableStatement procedimiento = conexion.prepareCall("{call actualizarcomercio(?,?)}");
296             procedimiento.setInt(1, IdComercio);
297             procedimiento.setString(2, comercio);
298             procedimiento.execute();
299
300             Commit();
            } catch (Exception e) {
302             JOptionPane.showMessageDialog(null, "Error en la eliminación");
303             Logger.getLogger(Handler.class.getName()).log(Level.SEVERE, null, e);
304             Rollback();
305         }
306     }
```

Imagen 12 Implementación del procedimiento de cambio de nombre en el método de eliminación en la aplicación

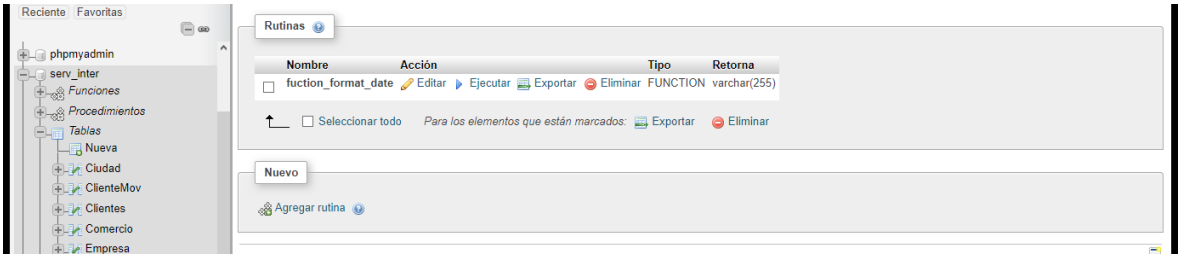

Imagen 13 Función que se utilizará para dar formato a la fecha que se ingrese desde la aplicación

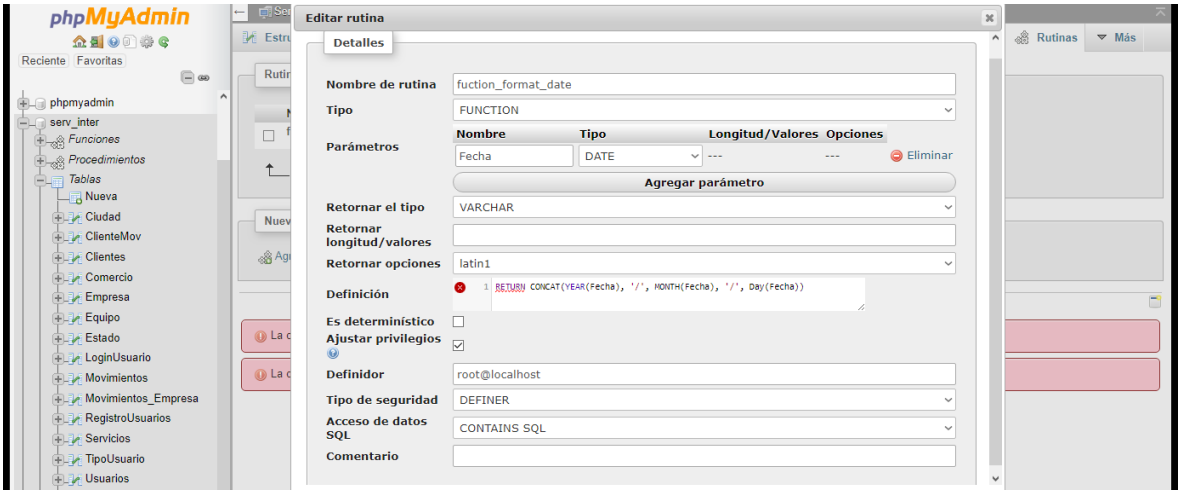

Imagen 14 Creación de la función que se encargará de dar formato a la fecha

"Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos"

```
764     public ArrayList<ClaseReportAllClientes> showRepAllClientes() {
765         String qry = "SELECT Empresa.IDEMPRESA, Comercio.NombreComercial, Empresa.Nombre, Clientes.Nombre, "
766                 + "Clientes.ApellidoP, Clientes.ApellidoM, fuction_format_date(Movimientos.Fecha) AS Fecha"
767                 + " FROM ((((Movimientos INNER JOIN Equipo ON Movimientos.NoSerie=Equipo.NOSERIE) INNER JOIN"
768                 + " Servicios ON Equipo.NoSERIE=Servicios.NoSERIE)"
769                 + "INNER JOIN Clientes ON Servicios.NoContrato=Clientes.NoContrato)INNER JOIN Empresa ON "
770                 + "Clientes.IDEMPRESA=Empresa.IDEMPRESA)INNER JOIN Comercio ON Empresa.IDEMPRESA=Comercio.IdEmpresa;";
771         try {
772             ResultSet rs = sentencia.executeQuery(qry);
773             ArrayList<ClaseReportAllClientes> lista = new ArrayList<>();
774             while (rs.next()) {
775
776                 ClaseReportAllClientes obj = new ClaseReportAllClientes();
777                 obj.setIDEMPRESA(rs.getInt("Empresa.IDEMPRESA"));
778                 obj.setNombreComercial(rs.getString("NombreComercial"));
779                 obj.setNombre(rs.getString("Empresa.Nombre"));
780                 obj.setNombreCliente(rs.getString("Clientes.Nombre"));
781                 obj.setApellidoP(rs.getString("Clientes.ApellidoP"));
782                 obj.setApellidoM(rs.getString("Clientes.ApellidoM"));
783                 obj.setFecha(rs.getString("Fecha"));
784                 lista.add(obj);
785
786             }
787             rs.close();
```

Imagen 15 Implementación de la función creada anteriormente en el código de la aplicación en la sección de la FECHA para el respectivo reporte

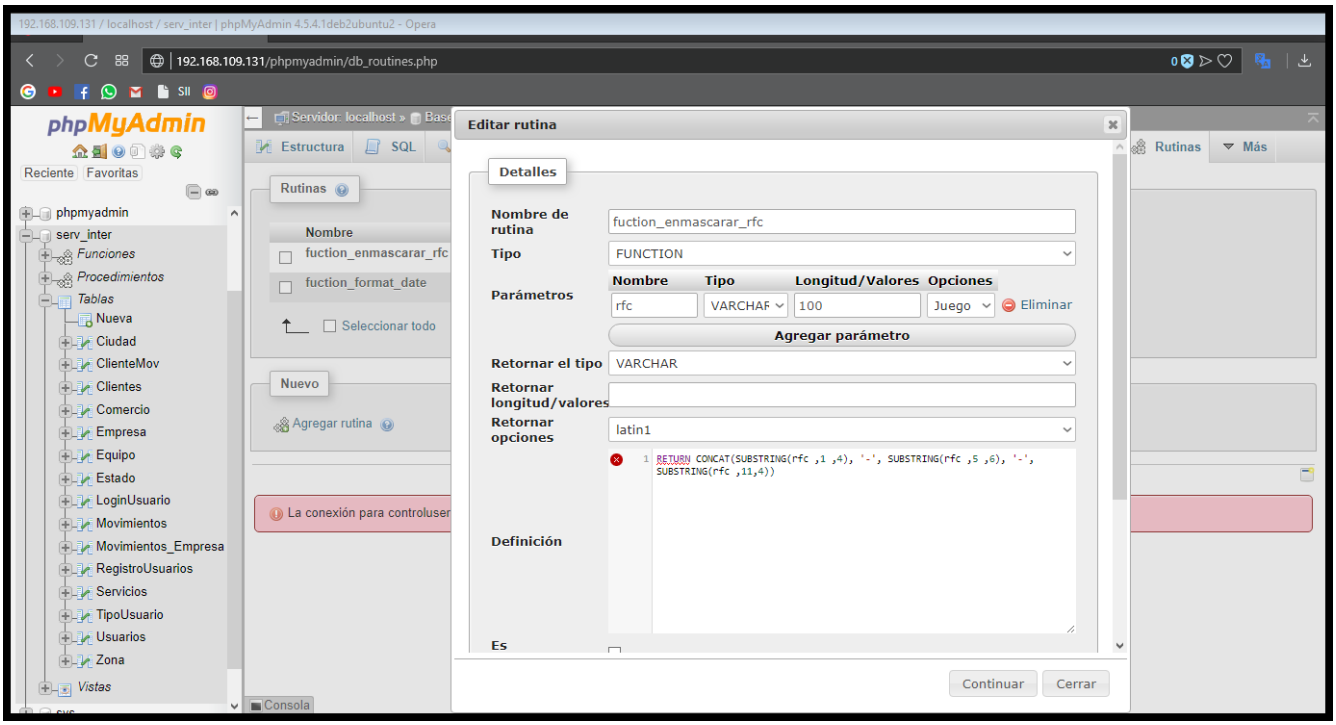

Imagen 16 Creación de la función que dará formato al RFC de determinada empresa

"Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos"

```
1526    public Object RFC(String rfc) {
1527        try {
1528
1529            Statement sentencia2 = conexion.createStatement();
1530            String qry = "fuction_enmascarar_rfc(" + rfc + ");";
1531
1532            ResultSet rs = sentencia.executeQuery(qry);
1533            return rs.getString("RFC");
1534        } catch (SQLException ex) {
1535            Logger.getLogger(Handler.class.getName()).log(Level.SEVERE, null, ex);
1536            return null;
1537        }
1538    }
```

Imagen 17 **Implementación de la función en el código de la aplicación**

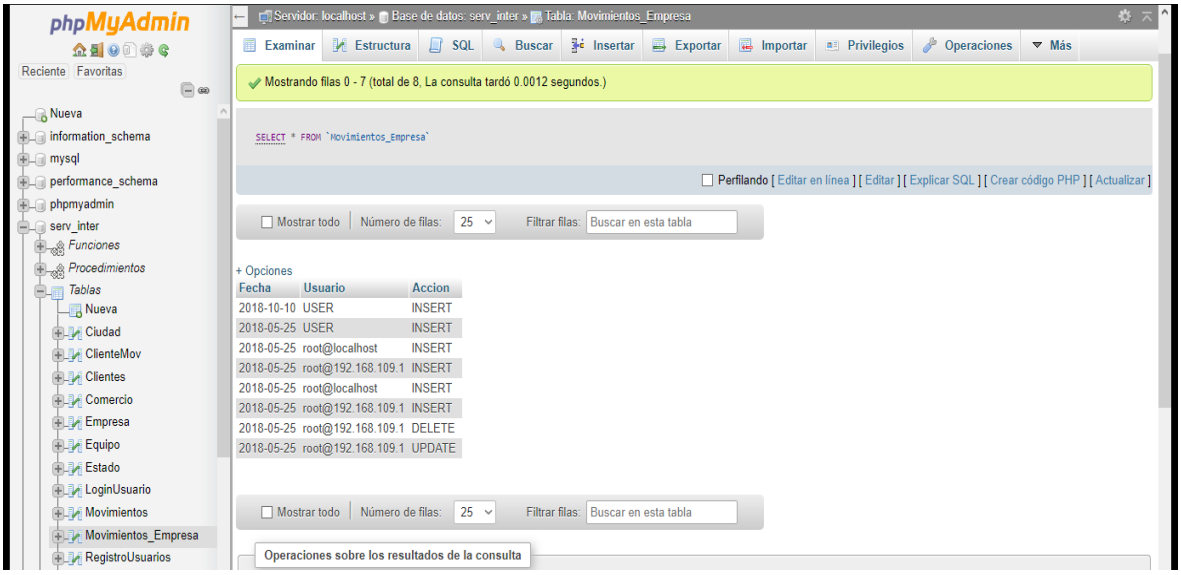

Imagen 18 **Creación de la tabla MovimientosEmpresa donde se registraran las actividades definidas por los triggers**

"Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos"

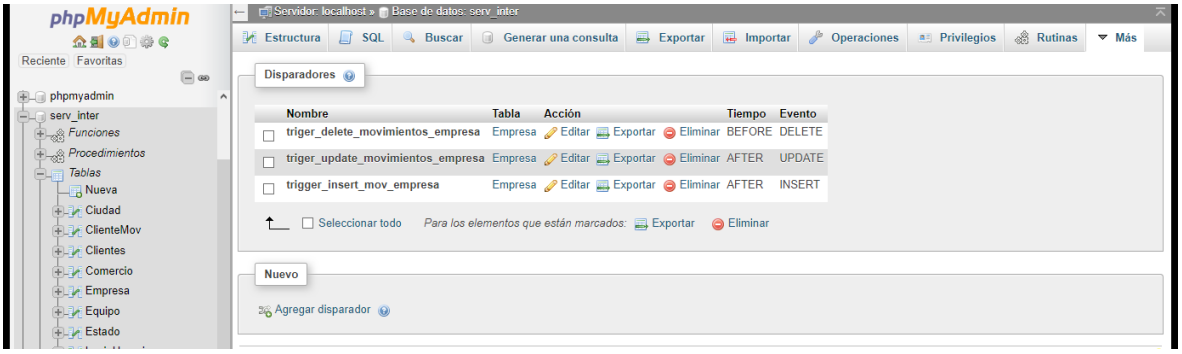

Imagen 19 Lista de cada uno de los triggers que se ejecutarán dependiendo de la operación realizada

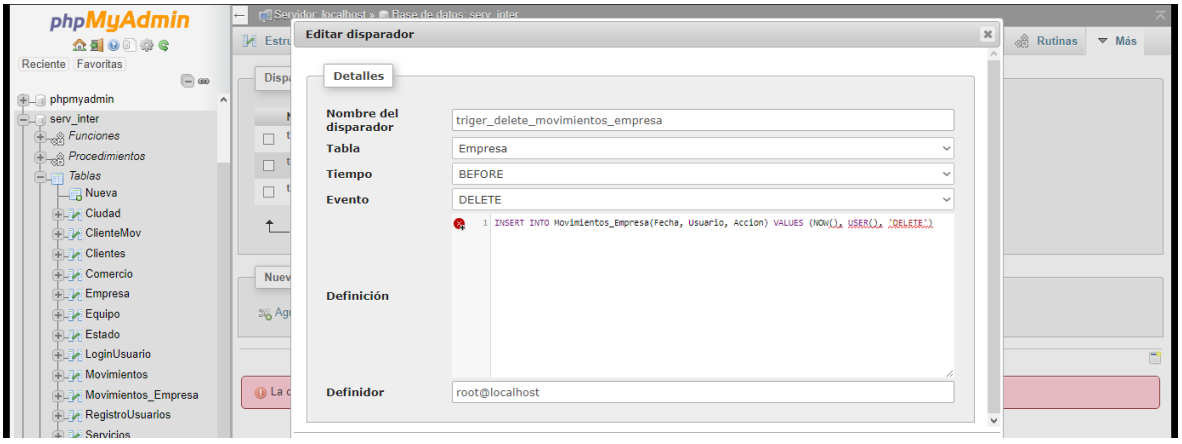

Imagen 20 Estructura del disparador que se encarga de registrar en la tabla de Movimientos_Empresa después de una eliminación

"Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos"

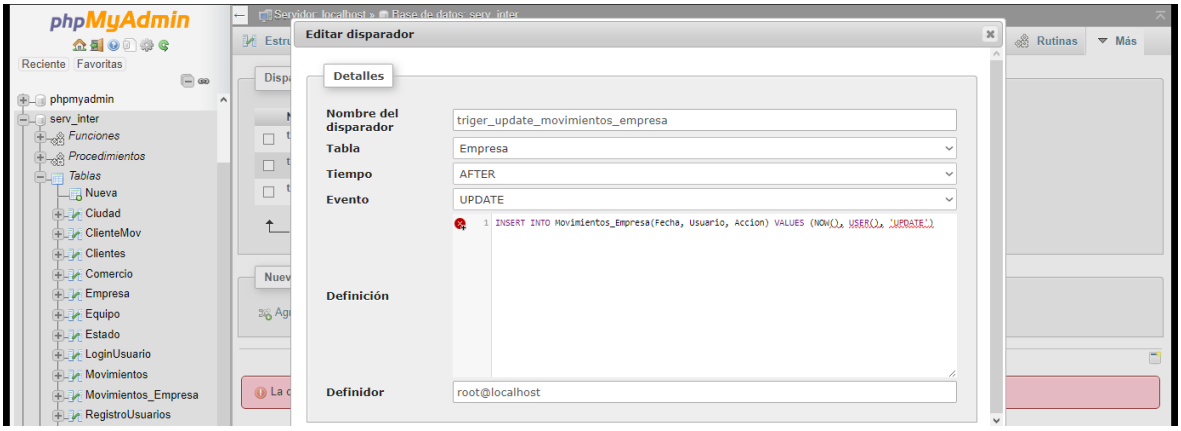

Imagen 21 Estructura del disparador que se encarga de registrar en la tabla de Movimientos_Empresa después de una actualización

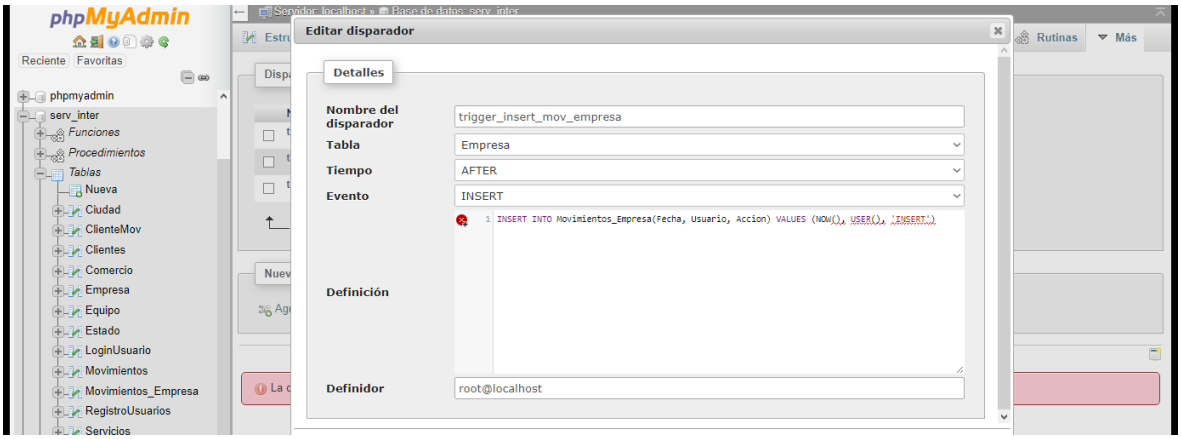

Imagen 22 Estructura del disparador que se encarga de registrar en la tabla de Movimientos_Empresa después de una inserción

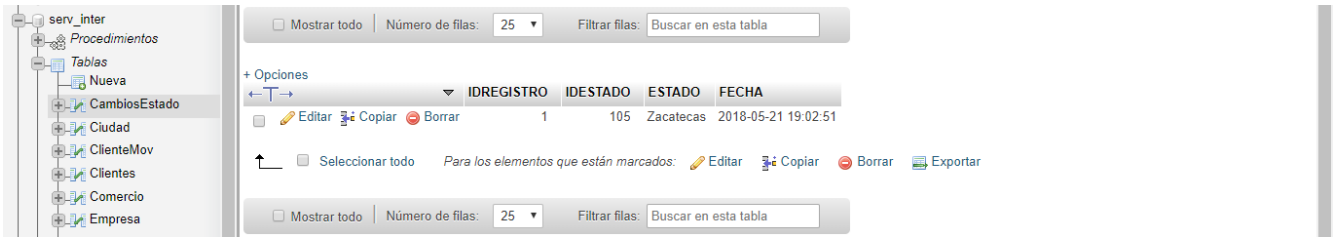

Imagen 23 Creación de la tabla que se encargará de registrar los movimientos realizados en la tabla de Estados

"Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos"

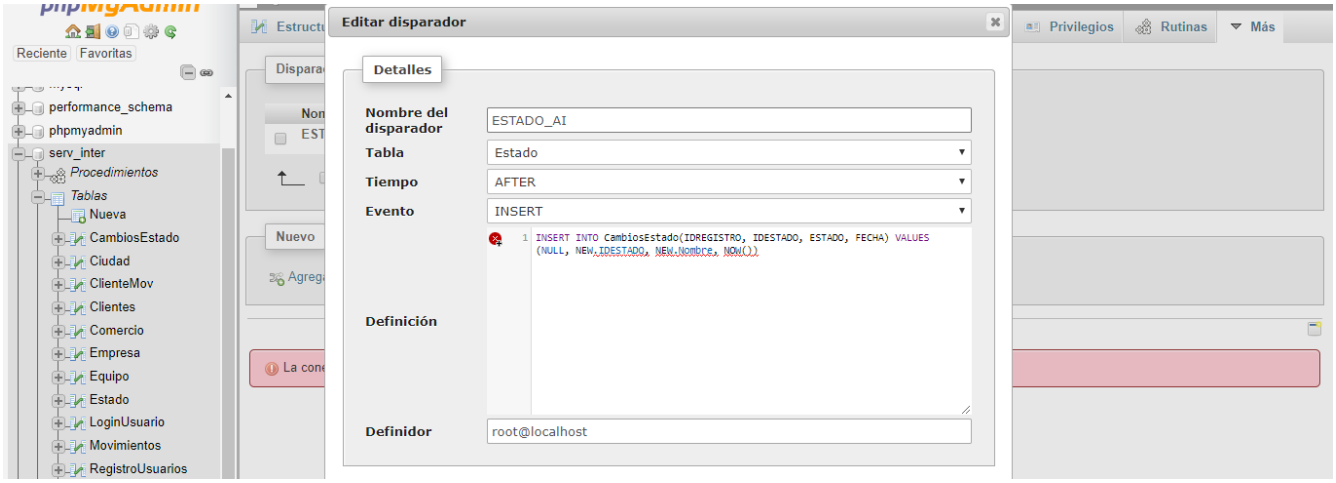

Imagen 24 Estructura el disparador que registrará en la tabla de CambioEstado después de que se realice una inserción

Conclusiones

Miguel Magdaleno Rosales

A pesar de las diversas complicaciones que surgieron al desarrollar la práctica, se logró implementar cada uno de los elementos propuestos de manera correcta, obteniendo resultados positivos. Con esto además se logró ampliar el repertorio sobre las diferentes características con las que puede contar una base de datos para así lograr un desempeño óptimo, además de contener la información de acuerdo a los estándares de una organización o cliente para su posterior manejo y respaldo.

Diego Ulises Martínez Aguilar

Siendo ésta práctica aquella con la que terminamos la materia de taller de base de datos, obtuvimos un producto totalmente funcional en el cual integramos todos los conocimientos adquiridos, en ésta ocasión se presentaron diversos problemas dado que se tenían muchas entregas finales en otras materias lo que afectó considerablemente la demora de la entrega, finalmente cabe mencionar que adquirimos las competencias y habilidades propuestas en la materia, se llevaron cada una de las prácticas en un entorno de trabajo real, además que contribuye a tener un producto eficiente y eficaz con un entorno amigable al usuario final, también se tiene un buen control de seguridad con respecto a nuestro servidor.

Referencias

https://manuales.guebs.com/mysql-5.0/stored-procedures.html#create-procedure

http://mysql.conclase.net/curso/?cap=011#

https://www.siteground.es/kb/que-son-triggers-mysql-uso/

Apuntes, curso FBD.