Proyecto de Sistema de Información basado en Bases de Datos Relacionales

Nombre	Bryan Moreno Picamán
Tema	Almacen y Empresa de Informatica
Desarrollador/es	• Bryan Moreno Picamán • Aaron Rodriguez Bueno • Miguel Angel Rodriguez Serrano • David Bueno Gonzalez
Grupo Grande	Grupo C
Grupo Pequeño	C2
Tutor de proyecto	Daniel Sánchez

Nota: utilice para cada sección tantas páginas como sea necesario. Este documento será entregado al profesor junto con la implementación del sistema, para la evaluación final de la práctica, para la que se tendrá en cuenta asimismo el resultado de la defensa presencial de la práctica.

Descripción detallada del proyecto

Será necesario elaborar un documento de descripción del sistema que recoja detalladamente el comportamiento del sistema, los datos que recibe, trata, almacena y devuelve, y cualquier consideración sobre el funcionamiento del mismo. Es necesario identificar un área funcional por cada uno de los componentes del grupo, de una complejidad y volumen de trabajo similar.

Registro de la gestión del almacén y pedidos

El sistema de gestión de almacén debe mantener los datos correspondientes a los productos y sus características:

- idAlmacen
- Nombre
- Tipo
- Referencia
- Precio de compra
- Precio de venta
- IVA aplicado
- Proveedor
- Peso
- Volumen (embalado)

Se tendrán almacenados los equipos en stock que existen en el almacén, almacenando una información mínima de cada uno de ellos
Pueden existir mismos productos (mismo nombre y tipo) que provengan de distintos proveedores y con distinto precio, se consideraran equipos distintos y tendrán una referencia distinta (identificador)

Cada almacén tendrá una entrada para cada uno de los productos, siento el producto 1 del almacén 0 (central), diferente del producto 1 del almacén 1 (almacén pequeño), no obstante la referencia debe ser la misma para poder pedir de forma adecuada desde los almacenes pequeños al central.

Así mismo se debe disponer de una forma de ver la cantidad de unidades vendidas de cada uno de los equipos, para llevar una cuenta de los que se venden y en qué cantidades.

A parte de esto se necesita que el sistema avise cuando las unidades de un producto lleguen a 5, dando un aviso de stock bajo, así como un nuevo aviso cuando llegue a 1, con aviso de ultima unidad. Controlándose también que no se quede el stock a 0. Los pedidos se pasaran al sistema de gestión de Compras, encargado de realizar las operaciones correspondientes a este proceso.

El documento de análisis de requisitos consta de tres listas, donde habrá que distinguir los requisitos correspondientes a cada área funcional:

Análisis de requisitos

Requisitos Funcionales

RFA01.- Alta de producto
- E: RD01
- A/M: RD02
- S:

RFA02.- Consulta Historial Ventas
- E: RD03
- A/M:
- S:RD04

RFA03.- Modificar producto
- E: RD05
- A/M: RD06
- S:

RFA05.- Alta Almacén
- E: RD10
- A/M: RD11
- S:RD12

RFA06.- Buscar Productos
- E: RD13
- A/M:
- S:RD14

Requisitos de Datos

RDA01.- Nombre
Tipo
Referencia
Unidades
PCompra
PVenta
IVA Aplicado
Proveedor
Peso
Volumen

RDA02.- Nombre
Tipo
Referencia
Unidades
PCompra
PVenta
IVA Aplicado
Proveedor
Peso
Volumen

RDA03.- Botón consultar historial

RDA04.- Nombre
Referencia
Nº Unidades Vendidas

RDA05.- Referencia

RDA06.- Nombre
Tipo
Unidades
PCompra
PVenta
IVA Aplicado
Proveedor
Peso
Volumen

RDA10.- Botón "Alta de Almacen"

RDA11.- idAlmacen
Dirección

RDA12.-Mensaje "Alta realizada"

RDA13.- Cuadro búsqueda

RDA14.- Listado de productos que coinciden

Restricciones Semánticas

RSA01 - RDA01 - RFA01.- Referencia debe ser única

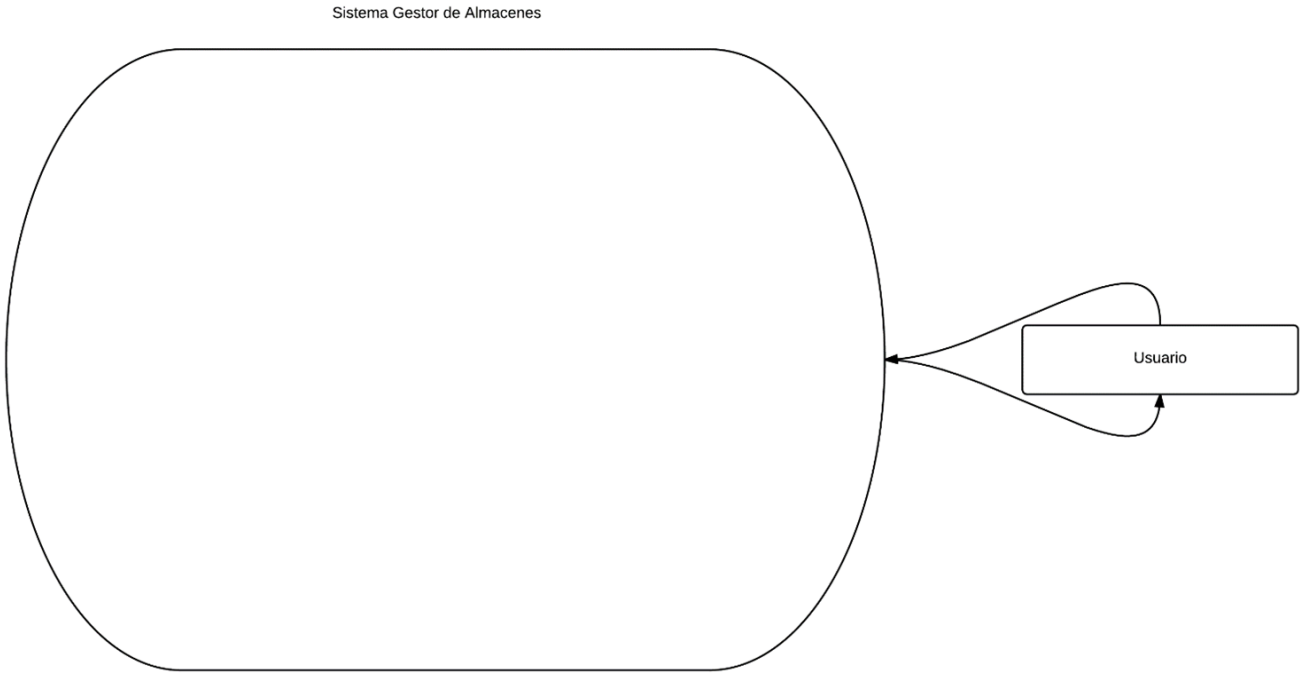
RSA02 – RDA01 - RFA01.- Pcompra no puede ser superior a Pventa

RSA03 – RDA06 – RFA03.- No se puede modificar la referencia

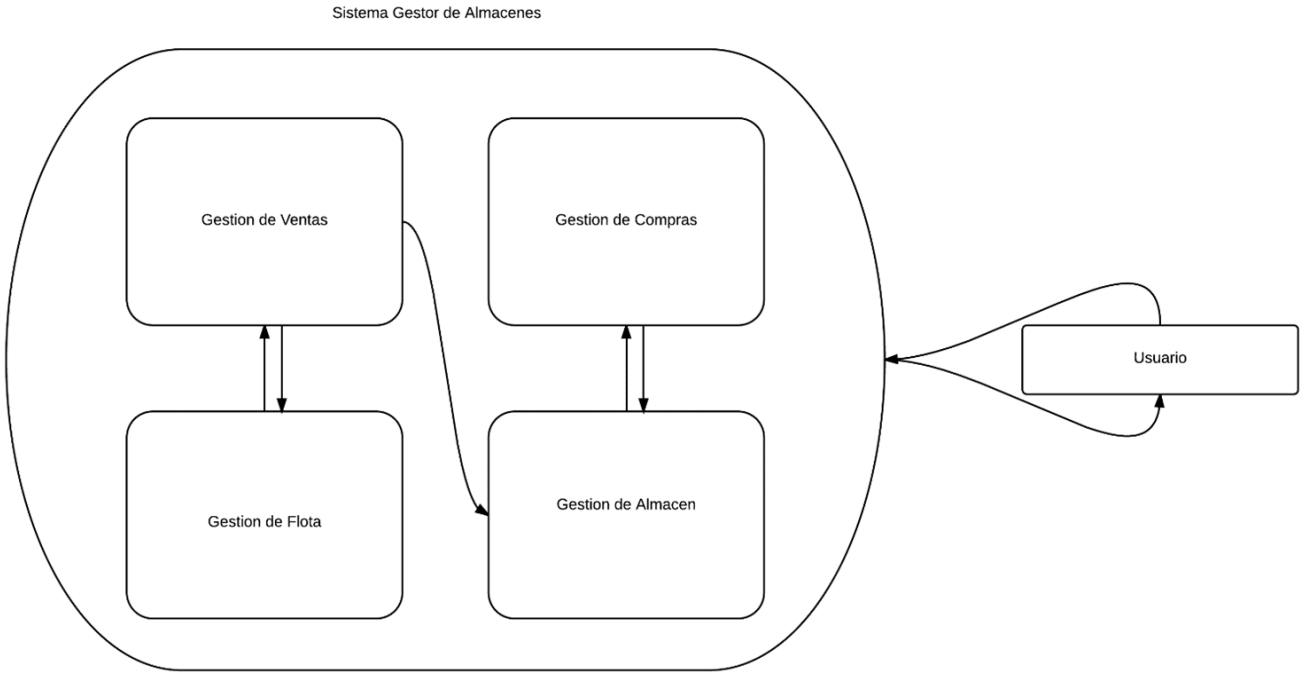
Diseño (análisis conjunto de datos y funciones orientado a las funciones)

Esquema de caja negra (a realizar conjuntamente por el equipo)

Esquema de caja negra.



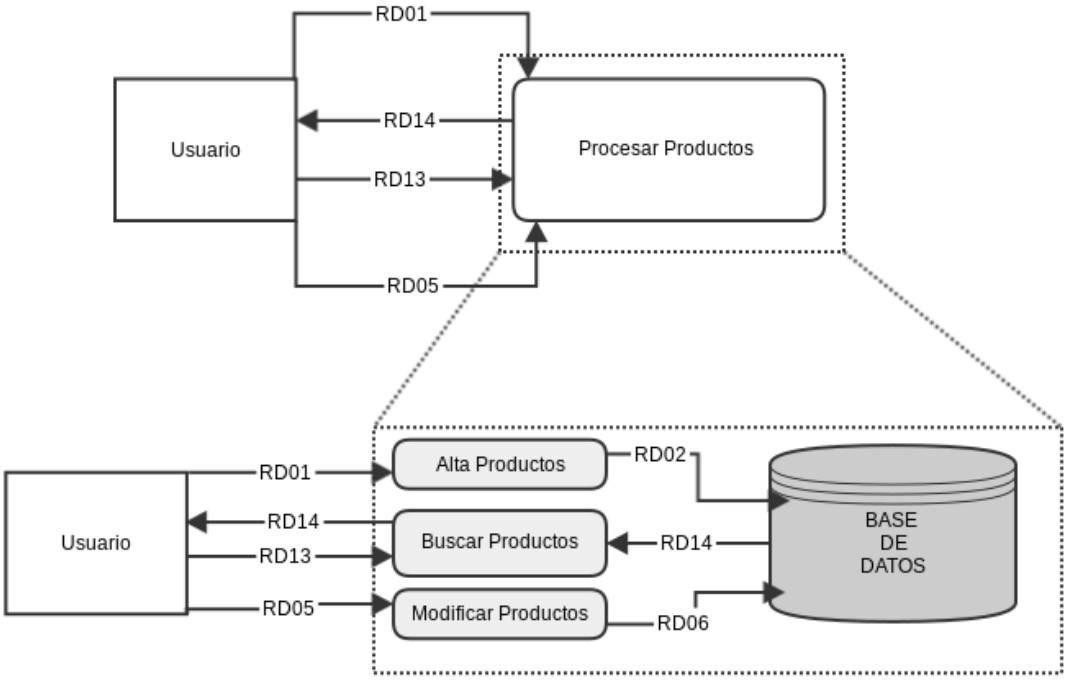
Esquema de caja negra completo.



Consta de:

Esquema armazón (a realizar conjuntamente por el equipo)

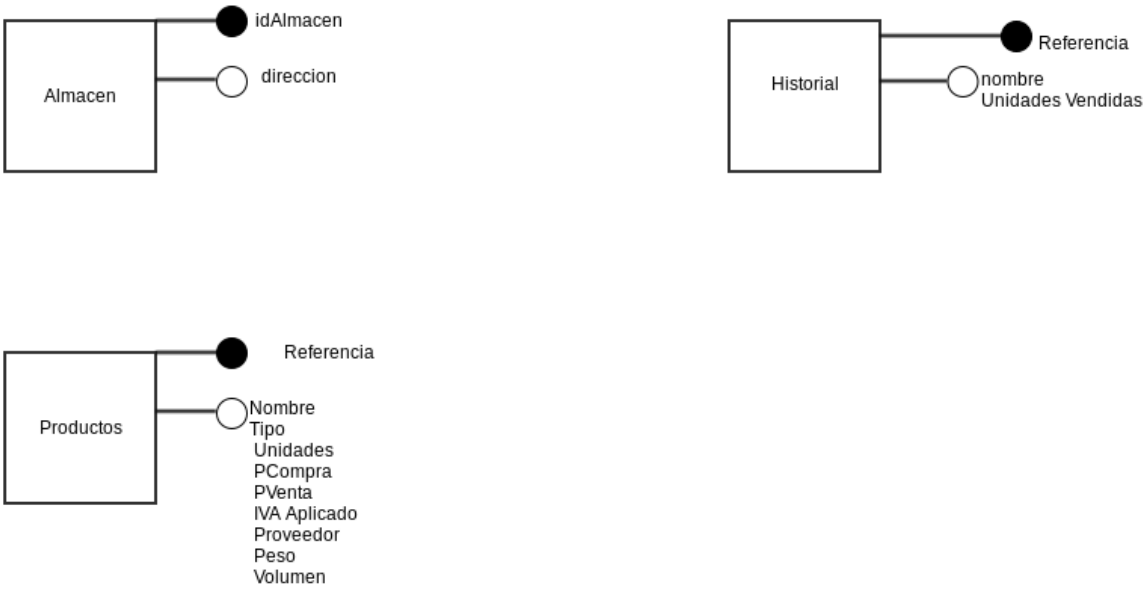
Diagrama de Flujo de Datos



Consta de:

Refinamientos (cada miembro del equipo hace el de su área funcional)

Esquema Externo



Operaciones de datos para el esquema final F (cada miembro del equipo hace las de su área funcional)

Operaciones de Datos

OA01.- Dar de alta un producto a partir de Referencia, Nombre, Tipo, Unidades, PCompra, Pventa, IVAAplicado, Proveedor, Peso y Volumen

OA02.- Buscar un producto por referencia, nombre o tipo

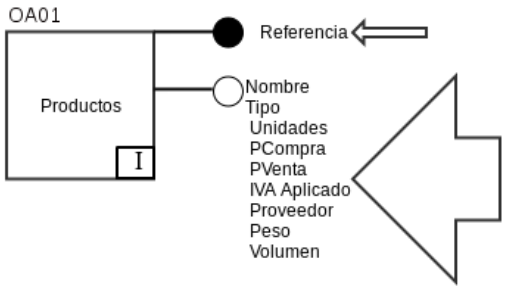
OA03.- Modificar un producto, dando la referencia y los datos a modificar (Nombre, Tipo, Unidades, PCompra, Pventa, IVAAplicado, Proveedor, Peso y Volumen)

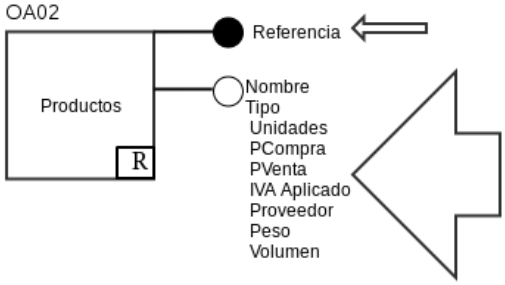
OA04.- Consultar Historial dando la referencia

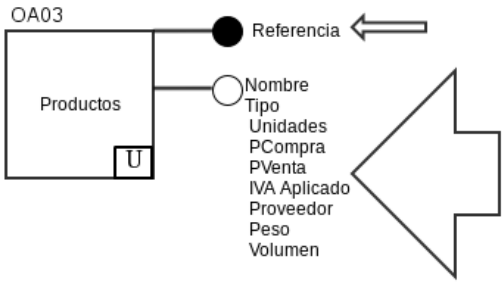
OA05.- Alta de almacén, dando la IdAlmacen y su dirección

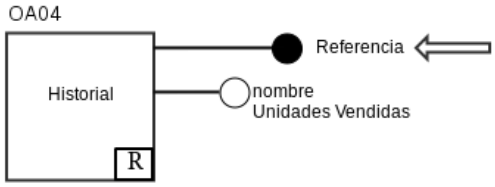
Esquemas de operación y navegación para las operaciones de datos (cada miembro del equipo hace los esquemas para su área funcional)

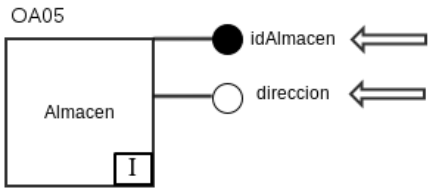
Esquemas de navegación.





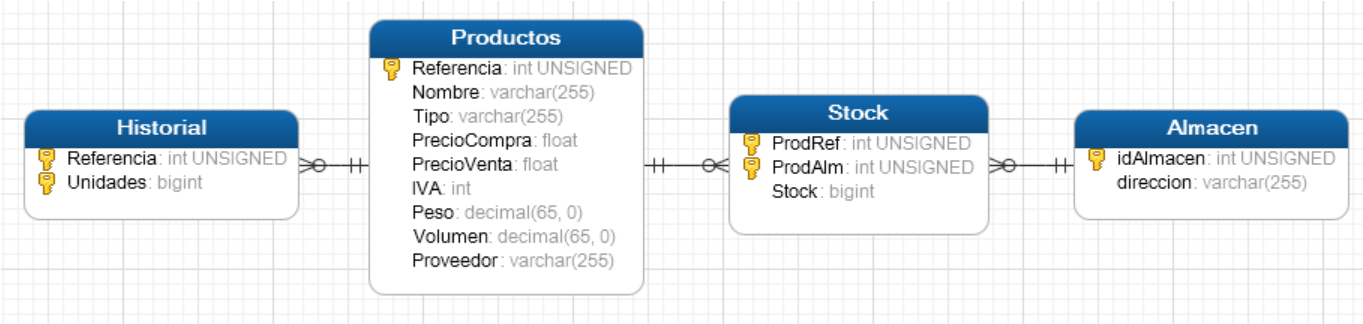






Diseño lógico relacional a partir del esquema conceptual final D

Paso a tablas del esquema E/R que representa la base de datos completa.



Falta el resto de la base.

Normalización de las tablas.

Es necesario indicar para todas las tablas las dependencias funcionales, todas las claves candidatas, y si están en FNBC (en caso contrario, dividir las tablas hasta conseguir que estén en FNBC)

Historial(Referencia,Unidades)

Solo dispone de las triviales. La clave candidata es Referencia,Unidades.

Esta en todas las formas normales.

Productos(Referencia,Nombre,Tipo,PrecioCompra,PrecioVenta,Iva,Peso,Volumen,Proveedor)

Referencia → Nombre

Referencia → Tipo

Referencia → PrecioCompra

Referencia → PrecioVenta

Referencia → IVA

Referencia → Peso

Referencia → Volumen

Referencia → Proveedor

Esta 1º forma normal. Lo atributos no primos dependen de forma completa de la clave candidata por lo que esta en 2º forma normal. No hay transitividades peligrosas por lo que esta en 3º forma normal. Todo determinante es clave por lo qe esta en FNByC

Stock(ProdRef,ProdAlm,Stock)

ProdRef ProdAlm → Stock

Esta 1º forma normal. Lo atributos no primos dependen de forma completa de la clave candidata por lo que esta en 2º forma normal. No hay transitividades peligrosas por lo que esta en 3º forma normal. Todo determinante es clave por lo qe esta en FNByC

Almacen(idAlmacen,direccion)

idAlmacen → direccion

Esta 1º forma normal. Lo atributos no primos dependen de forma completa de la clave candidata por lo que esta en 2º forma normal. No hay transitividades peligrosas por lo que esta en 3º forma normal. Todo determinante es clave por lo qe esta en FNByC

Diseño físico relacional

Sentencias SQL para crear las tablas y definir las restricciones semánticas de integridad.

También sentencias de inserción de algunos datos en las tablas para realizar pruebas posteriormente.

Asimismo, código de los disparadores programados.

```
SET FOREIGN_KEY_CHECKS=0;

-- ----------------------------
-- Table structure for Almacen
-- ----------------------------
DROP TABLE IF EXISTS `Almacen`;
CREATE TABLE `Almacen` (
  `idAlmacen` int(255) unsigned NOT NULL AUTO_INCREMENT,
  `direccion` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`idAlmacen`)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=utf8;

-- ----------------------------
-- Records of Almacen
-- ----------------------------
INSERT INTO `Almacen` VALUES ('1', 'C/Pepa nº1');
INSERT INTO `Almacen` VALUES ('2', 'C/Juana nº2 ');
INSERT INTO `Almacen` VALUES ('3', 'Poligono ALmacen 3');
INSERT INTO `Almacen` VALUES ('4', 'Edificio Colon Planta 4');
INSERT INTO `Almacen` VALUES ('5', null);

-- ----------------------------
-- Table structure for Historial
-- ----------------------------
DROP TABLE IF EXISTS `Historial`;
CREATE TABLE `Historial` (
  `Referencia` int(255) unsigned NOT NULL,
  `Unidades` bigint(255) NOT NULL,
  PRIMARY KEY (`Referencia`,`Unidades`),
  CONSTRAINT `Referencia` FOREIGN KEY (`Referencia`) REFERENCES `Productos` (`Referencia`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------
-- Records of Historial
-- ----------------------------
INSERT INTO `Historial` VALUES ('1', '120');
INSERT INTO `Historial` VALUES ('2', '1');
INSERT INTO `Historial` VALUES ('3', '4');
INSERT INTO `Historial` VALUES ('4', '6');

-- ----------------------------
```

```
-- Table structure for Productos
-- ----------------------------
DROP TABLE IF EXISTS `Productos`;
CREATE TABLE `Productos` (
  `Referencia` int(255) unsigned NOT NULL AUTO_INCREMENT,
  `Nombre` varchar(255) DEFAULT NULL,
  `Tipo` varchar(255) DEFAULT NULL,
  `PrecioCompra` float(255,0) DEFAULT NULL,
  `PrecioVenta` float(255,0) DEFAULT NULL,
  `IVA` int(255) DEFAULT '21',
  `Peso` decimal(65,0) DEFAULT '0',
  `Volumen` decimal(65,0) DEFAULT '0',
  `Proveedor` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`Referencia`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8;

-- ----------------------------
-- Records of Productos
-- ----------------------------
INSERT INTO `Productos` VALUES ('1', 'Raton', 'BT', '10', '15', '21', '0', '2', 'INFORMATICA SA');
INSERT INTO `Productos` VALUES ('2', 'Teclado', 'WiFi', '20', '25', '21', '1', '3', 'REPUESTOS SL');
INSERT INTO `Productos` VALUES ('3', 'Monitor', 'Cable', '12', '23', '21', '0', '0', 'INFORMATICA SA');
INSERT INTO `Productos` VALUES ('4', 'Torre', 'ATX', '59', '68', '21', '3', '5', 'MEGASUR SA');

-- ----------------------------
-- Table structure for Stock
-- ----------------------------
DROP TABLE IF EXISTS `Stock`;
CREATE TABLE `Stock` (
  `ProdRef` int(255) unsigned NOT NULL,
  `ProdAlm` int(255) unsigned NOT NULL,
  `Stock` bigint(255) DEFAULT NULL,
  PRIMARY KEY (`ProdRef`,`ProdAlm`),
  KEY `Alm` (`ProdAlm`),
  CONSTRAINT `Alm` FOREIGN KEY (`ProdAlm`) REFERENCES `Almacen` (`idAlmacen`),
  CONSTRAINT `Ref` FOREIGN KEY (`ProdRef`) REFERENCES `Productos` (`Referencia`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------
-- Records of Stock
-- ----------------------------
INSERT INTO `Stock` VALUES ('1', '2', '23');
INSERT INTO `Stock` VALUES ('1', '3', '12');
INSERT INTO `Stock` VALUES ('2', '2', '54');
INSERT INTO `Stock` VALUES ('2', '3', '10');
```

DISPARADORES:

Disparador 1.-

Cuando se inserta un producto, el precio de venta, debe ser superior al de compra.

```
CREATE TRIGGER
    insercionProducto BEFORE INSERT Productos

BEGIN

    if(:new.PrecioCompra>=:new.PrecioVenta)
        EXCEPTION
            DBMS_OUTPUT.PUT_LINE("Precio de Venta debe ser myor que precio de compra.");
        RAISE;
    else
        INSERT INTO Productos
            VALUES(:NEW.Referencia,:NEW.Nombre,:NEW.Tipo,:NEW.PrecioCompra,
                    :NEW.PrecioVenta,:NEW.IVA,:NEW.Peso,:NEW.Volumen,:NEW.Proveedor);
    end if
END;
```

Disparador 2.-

Cuando se modifica un producto, en el campo precio, se debe cumplir que el precio de venta debe ser superior al de compra (similar a disparador 1)

```
CREATE TRIGGER
    modificacionProducto before update on Productos
    BEGIN

    IF UPDATING ('PrecioCompra') THEN
        if(:new.PrecioCompra>=:old.PrecioVenta)
            EXCEPTION
            DBMS_OUTPUT.PUT_LINE("Precio de compra incorrecto.");
            RAISE;
        else
            :old.PrecioCompra=:new.PrecioCompra;
        end if;
    ELSIF UPDATING ('PrecioVenta') THEN
        if(:old.PrecioCompra>=:new.PrecioVenta)
            EXCEPTION
            DBMS_OUTPUT.PUT_LINE("Precio de venta incorrecto.");
            RAISE;
        else
            :old.PrecioVenta=:new.PrecioVenta;
        end if;
    END IF;
END;
```