Resoluciones (modelos) de Ejercicios Prácticos

Bases de Datos Activas – Práctico 1

Introducción

A continuación, se presentan las probables resoluciones para los casos prácticos de la guía 1, cada alumno puede incorporar las variantes que estimen necesarias, si están familiarizados con algún manejador de bases de datos, pueden ir adecuando a las sentencias SQL de cada uno de ellos.
En las definiciones de las tablas activas, incorporaremos restricciones, reglas de negocios y triggers. Las restricciones pueden ser dadas a ciertas columnas o a un conjunto de ellas, por ejemplo, para determinar la clave primaria, los valores por defecto que puedan asumir ciertos atributos o un rango de ellos, asignando el criterio de unicidad, restricción de valores nulos, claves externas para relacionar columnas de distintas tablas y la definición de los triggers siendo uno de los conceptos principales de este tipo de bases de datos.

1)

```
Create table Empleados
(
        Dni             Char(8) Primary Key,
        Legajo          Char(6) Unique,
        Ape_y_Nom       Varchar(60),
        Cargo           Int Check(cargo between 50 and 120),
        Sueldo          Dec(8,2) Check(sueldo <= 90000)
);
```

Alternativa: crear un tipo de datos de usuario a partir de un tipo de dato estándar, luego lo podemos referenciar, por ejemplo, cuando definimos una tabla.

```
Create Domain Documento Int(8) Check(Documento> 0);
```

Se modificaría en la definición de la tabla Empleados para el atributo Dni, que haría referencia al tipo de dato Documento creado previamente:

```
        Dni             Documento Primary Key,
```

```
Create Trigger Control_sueldo
Before Update of sueldo on Empleados   --Cuando se actualiza el atributo sueldo de la tabla Empleados
For each row                           --se activa este disparador, controlando que el sueldo no se
        Begin                          --incremente en un más 20% del valor del sueldo anterior
                If :new.sueldo> (:old.sueldo * 1.20)
                        Then   "Error-Debe ir sentencia que cancela la actualización"
                Endif;
        End;
```

```
Create Trigger Adicional_cargo
After Insert or Update of cargo on Empleados
For each row
        Begin
        If :new.cargo = 90
                Then
                :new.sueldo = :new.sueldo +
                (Select importe_basico from Importes_basicos where cargo = :new.cargo) * 0.15
        Endif:
        End;
```

Una forma de incorporar comentarios en las sentencias SQL, es mediante el uso de -- o /* */ :

```
-- Esto es un comentario
```

Y otra manera es:

```
        /*      Comentario
                en varias
                líneas
        */
```

2)

```
Create table Seguros_autos
(
        Nro_poliza      Char(8) Primary Key,
        Dni             Char(8) references Clientes(Dni),
        Marca_auto      Char(7) Check(marca In ('Ford','Renault','Fiat','Peugeot','VW', 'Toyota',
                                               'Nissan ')),
        Año_modelo      Char(4) default ('2019'),
        Nro_patente     Char(10),
        Nro_motor       Varchar(30),
        Periodo_pago    Char (6) default ('202001'),  -- Sólo consideramos el primer trimestre del año
        Imp_pagar       Dec(10,2) not null,
        Foreign Key(marca_auto,año_modelo) references Vehiculos(marca_auto,año_modelo)
);

Create Trigger Facturas_mensuales
After Insert on Seguros_autos
For each row

        Declare  importe_cuota Dec(10,2)    --Variable auxiliar declarada dentro del trigger
        Declare  nro_cuota integer
        Declare comprobante   Char (16)
        Declare fecha_pago_aux Char (8)
        Begin

  /* Insertamos las filas correspondientes a las facturas mensuales de la respectiva póliza de seguro,
     en la tabla Cuotas_seguros
*/
                importe_cuota= :new.imp_pagar / 3
                nro_cuota = 1
                While nro_cuota < 4
                Loop
                nro_comprobante = :new.Nro_poliza + :new.Periodo_pago +convert(char(2),
                                  nro_cuota))

                /*  Ejemplo de armar la fecha de pago: sabiendo que el día de vencimiento es el
                    día 15 de cada mes, donde el mes coincidiría con el número de cuota y del atributo
                    período de pago de la tabla Seguros_autos, recuperaremos los 2 últimos caracteres
                    del año que está contenido en ese atributo, en este caso rescatamos los caracteres
                    '20', todo esto lo concatenamos y luego al insertarlo en la tabla Cuotas_seguros, lo
                    convertimos a tipo fecha (date).  Recuerden que es sólo una alternativa de
                    resolución de este trigger, queda abierta a todas las propuestas de solución que
                    cada uno pueda aportar de acuerdo a su técnica de trabajo.
                */

                fecha_pago_aux =  ´15' + convert(char(2), nro_cuota)) +
                                  substring (:new.Periodo_pago,3,2)

                Insert into Cuotas_seguros  values (    nro_comprobante,
                                                        importe_cuota,
                                                        convert(fecha_pago_aux, date)
                                                   )
                nro_cuota = nro_cuota + 1
                End Loop
        End;
```

3)

```
Create table Pedidos
(       Nro_orden               Char(6) not null unique,
        Fecha_pedido            Date default today,
        Cuit_cliente            Char(11) not null,
        Cuit_proveedor          Char(11) not null,
        Cuit_cliente references Clientes(Cuit_cliente),
        Primary key(Cuit_cliente,nro_orden),
        Nro_producto            Number,
        Tipo_pedido             Char(1) Check(Tipo_pedido In ('M','C','G','H','N')),
        Cant_pedida             Int Check(Cant_pedida>0)
);
```

```
Create Trigger Control_pedido
Before Insert on Pedidos
For each row
        Begin

          /* La condición evalúa que la nueva cantidad pedida no sea mayor al valor del stock actual
        del producto que se rescata de la tabla Stock_productos  */

                If :new.cant_pedida > (Select stock_actual from Stock_productos
                                                        where producto = :new.nro_producto)
                        Then   "ERROR-Cantidad pedida superior al stock disponible-Cancelar pedido"
                Else
                        Update Stock_productos
                                Set stock_actual = stock_actual - :new.cant_pedida
                                where producto= :new.nro_producto
                Endif;
        End;
```

4)

Estas restricciones se agregarían a las sentencias del ejercicio 3), en primer lugar, al definir la tabla Pedidos tenemos:

```
        Alter table Pedidos add
                Constraint Clave_Producto Foreign Key (nro_producto,tipo_pedido) references
                Productos(nro_producto,tipo_pedido)

                Constraint CantidadPermitida check(cant_pedida between 10 and 2000)

        Alter table Pedidos with no check add
                Constraint ControlCuit check (Cuit_cliente<>Cuit_proveedor)
```

Restricción a incorporar a la tabla Stock_productos:

```
        Alter table Stock_productos add
                Constraint ControlStock check (stock_actual mayor o igual stock_minimo)
```

5)

```
Create Domain CodigoCliente Int(9) check (CodigoCliente> 0);

Create table Clientes
(
        Cod_cliente     CodigoCliente primary key,
        Nombre_cli      Varchar(40) not null,
        Direccion       Varchar(35) not null,
        Cod_postal      Char(5),
        Foreign key (cod_postal) references CodPostales(cod_postal),
        Importe_base    Dec(8,2) not null
);

Create table Facturas
(
        Nro_factura     Char(8) primary key,
        Fecha_fact      Date,
        Cliente_factura  CodigoCliente,
        Foreign key (cliente_factura)  references Clientes(cliente_factura),
        Tipo_descuento int(2) check(tipo_descuento between 5 and 20),
        Iva              int(2) check((iva=15) or (iva=21)),
        Importe_factura         Dec(10,2) not null
);

Create Trigger Calcular_importe_factura
After Insert or Update importe_base on Clientes
For each row
        Declare  resultado Dec(10,2)    --Variable auxiliar declarada dentro del trigger
        Begin
        -- Descontamos del importe base el respectivo importe de acuerdo al valor de descuento
```

```
        resultado = :new.importe_base – ((:new.tipo_descuento * :new.importe_base)/100)

    /* Al importe obtenido en el paso anterior que almacenamos en el campo auxiliar
       'resultado', debemos de incrementarlo de acuerdo al porcentaje del IVA a aplicar,
       teniendo en cuenta el valor del atributo iva.
    */

        resultado = resultado + ((:new.iva * resultado)/100)

        Update Facturas
            Set importe_factura = resultado
            Where cliente_factura = :new.cod_cliente
End;
```

6)

```
Create table Empleado_baja
(
    Dni         Char(8),
    Legajo      Char(6),
    Cargo       number,
    Usuario     Varchar2(15),
    Fecha       Date
);

Create trigger Empleado_eliminado   -- Se registran en una tabla, aquellos empleados dados de
After delete on Empleados           -- baja
For each row
    Begin
        Insert into Empleados_baja values (:old.Dni, :old.Legajo, :old.Cargo, User, Sysdate);
    End;

Create trigger Baja_usuario   -- Eliminamos el usuario del empleado, con el cual accedía al sistema
After delete on Empleados     -- de gestión de la empresa a la cual pertenecía
    Begin
        Delete from Usuarios where usuarios.dni= :old.Dni;
    End;
```

7)

```
Replace trigger Baja_usuario
After delete on Empleados
    Begin
        Delete from Usuarios where usuarios.legajo= :old.legajo;
    End;

Alter trigger Calcular_importe disable;
Alter table Consumos disable all triggers;
```

8)

Dependerá del manejador de base de datos que usemos, que permite distintos formatos para agregar columnas a una tabla.

Opción 1

```
Alter table Facturas Add fecha_vto date;
Alter table Facturas Add fecha_pago date;     Alter table Facturas Add intereses int;
```

Opción 2

```
Alter table Facturas Add fecha_vto date, fecha_pago date, intereses int;

Create Trigger Recargo_factura
After Update fecha_pago on Facturas
For each row
    Begin
        If  fecha_pago > fecha_vto then
        Update Facturas
            Set intereses = 5
            Where cliente_factura= :new.cod_cliente
        Endif;
    End;
```

9)

```
Create Trigger Importe_recargo
After Update intereses on Facturas
For each row
        Begin
                Update Facturas
                        Set importe_factura = importe_factura + ((importe_factura * intereses)/100)
                        Where cliente_factura= :new.cod_cliente
        End;
```

Comentarios

En los triggers, podemos hacer referencia a los valores que contienen los atributos al momento de actualizar o insertar las tablas, anteponiendo al nombre de cada atributo **:new.** y para hacer referencia a los valores de una fila, que vamos a eliminar de la tabla en ese caso anteponemos **:old.**, ejemplos:

:new.cantidad_pedida	Cantidad pedida de un cierto producto, que va a ser actualizada o insertada formando parte de una tupla
:old. Dni	Dni del empleado eliminado de la respectiva tabla

10)

```
Create table cursadas
(
        dni int primary key,
        cod_carrera int,
        cod_materia int,
        año_cursado int not null,
        condicion char(1) (check condicion in ('R', 'P', 'D')),
        nota_final int (check between 1 and 10),

        constraint control_materia foreign key (cod_carrera, cod_materia)
        references plan_carreras (cod_carrera, cod_materia)
);
```

```
Create trigger condicion_alumno

after insert on cursadas or update nota_final on cursadas

for each row
        begin
                if new.nota_final = 6 then
                        update cursadas
                        set condicion = 'R'
                        where dni = new.dni

                elseif  new.nota_final >= 7 then
                        update cursadas
                        set condicion = 'P'
                        where dni = new.dni

                        else
                                update cursadas
                                set condicion = 'D'
                                where dni = new.dni
                Endif;
        end;
```

Nota: Recuerden que estas resoluciones, pueden sufrir cambios de acuerdo a otra forma de plantearlos que cada uno desee incorporar, así como los formatos y las sintaxis de las sentencias, queda abierta a las distintas maneras de desarrollarlos que estimen conveniente, pero llegando siempre a las consignas requeridas en los ítems de los distintos casos prácticos.

Trabajo Práctico 1

Otra versión de resolución de los casos prácticos.

Ejercicio 1

```
CREATE DOMAIN DNI INT(8) CHECK (DNI > 0);

CREATE TABLE Empleados (
        Dni DNI,
        Legajo VARCHAR(6),
        ApellidoNombre VARCHAR(50),
        Cargo INT,
        Sueldo DEC(8,2),
        CONSTRAINT PK_Empleados PRIMARY KEY (Dni),
        CONSTRAINT UQ_Legajo UNIQUE (Legajo),
        CONSTRAINT CK_Cargo  CHECK (Cargo BETWEEN 50 AND 120),
        CONSTRAINT CK_Sueldo CHECK (SUELDO <= 90000)
);

CREATE TRIGGER Control_sueldo
BEFORE UPDATE of Sueldo ON Empleado
BEGIN
        FOR EACH ROW
        BEGIN
                IF :NEW.Sueldo > 1.2 * :OLD.Sueldo THEN
                ROLLBACK TRANSACTION
        ENDIF
END;

CREATE TRIGGER Adicional_cargo
BEFORE INSERT or UPDATE of Cargo ON Empleado
FOR EACH ROW
  BEGIN
    IF :NEW.Cargo = 90 THEN
      :NEW.Sueldo = :NEW.Sueldo + 0.15 *
        (SELECT importes_basicos FROM Importes_basicos WHERE cargo = :NEW.Cargo)
    ENDIF
  END;
```

Ejercicio2

```
CREATE TABLE Seguros_autos (
  NroPoliza INT,
  DniCliente INT,
  AñoVehiculo INT,
  Marca VARCHAR(10),
  Patente VARCHAR(7),
  NroMotor INT,
  Periodo INT,
  Importe DEC(8,2) NOT NULL,
  CONSTRAINT PK_Seguros_autos PRIMARY KEY (NroPoliza),
  CONSTRAINT FK_Seguros_autos_Clientes FOREIGN KEY (DniCliente)
  REFERENCES Clientes(DNI),
  CONSTRAINT DF_AñoVehiculo DEFAULT '2019' FOR Legajo,
  CONSTRAINT CK_Marca CHECK (Marca IN ('Ford', 'Renault', 'Fiat', 'Peugeout', 'VW', 'Toyota',
                                       Nissan')),
  CONSTRAINT DF_Periodo DEFAULT '202001' FOR Periodo,
  CONSTRAINT FK_Seguros_autos_Vehiculos FOREIGN KEY (Marca, AñoVehiculo)
  REFERENCES Vehiculos(marca, año_modelo)
);

CREATE TRIGGER Facturas_mensuales
AFTER INSERT ON Seguros_autos
FOR EACH ROW
  DECLARE CantidadCuotas INT
  DECLARE Comprobante VARCHAR(18)
```

```
  DECLARE ImporteCuota DEC(8,2)
  DECLARE FechaPago DATE
  DECLARE NroCuota INT
  BEGIN
    CantidadCuotas = 3
    NroCuota = 1
    ImporteCuota = :NEW.Importe / CantidadCuotas
    WHILE NroCuota <= CantidadCuotas LOOP
      Comprobante = CONCAT(
        CONVERT(VARCHAR(10), NroPoliza),
        CONVERT(VARCHAR(6), Periodo),
        CONVERT(VARCHAR(2), NroCuota)
      )
      FechaPago = CONVERT(DATE, CONCAT(
        CONVERT(VARCHAR(4), FLOOR(Periodo / 100)),
        CONVERT(VARCHAR(2), NroCuota),
        '15'
      ))
      INSERT INTO Cuotas_Seguros VALUES (Comprobante, ImporteCuota, FechaPago)
      NroCuota += 1
    ENDLOOP
  END;
```

Ejercicio 3

```
CREATE TABLE Pedidos (
  NroOrden INT,
  Fecha DATE,
  CuitCliente VARCHAR(11),
  CuitProveedor VARCHAR(11),
  NroProducto INT,
  Tipo INT,
  Cantidad INT,
  CONSTRAINT PK_Pedidos PRIMARY KEY (CuitCliente, NroOrden),
  CONSTRAINT UQ_NroOrder UNIQUE (NroOrden),
  CONSTRAINT DF_Fecha DEFAULT GETDATE() FOR Fecha,
  CONSTRAINT FK_Pedidos_Clientes FOREIGN KEY (CuitCliente) REFERENCES Clientes(Cuit),
  CONSTRAINT CK_Tipo CHECK (Tipo IN ('M', 'C', 'G', 'H', 'N')),
  CONSTRAINT CK_Cantidad CHECK (Cantidad > 0),);

CREATE TRIGGER Control_pedido
BEFORE INSERT ON Pedidos
FOR EACH ROW
  DECLARE Stock INT
  BEGIN
    Stock = (SELECT stock_actual FROM Stock_productos
             WHERE nro_producto = :NEW.NroProducto)
    IF :NEW.Cantidad > Stock THEN
      ROLLBACK TRANSACTION
    ELSE
      UPDATE Stock_productos
      SET stock_actual = (Stock - :NEW.Cantidad)
      WHERE nro_producto = :NEW.NroProducto
    ENDIF
  END;
```

Ejercicio 4

```
ALTER TABLE Pedidos ADD CONSTRAINT ClaveProducto FOREIGN KEY
(NroProductos, Tipo) REFERENCES Productos(nro_producto, tipo_pedido);
ALTER TABLE Pedidos ADD CONSTRAINT CantidadPermitida CHECK (Cantidad >=
10 AND Cantidad <= 2000);
ALTER TABLE Pedidos ADD CONSTRAINT ControlCUIT CHECK (CuitCliente !=
CuitProveedor);
ALTER TABLE Stock_productos ADD CONSTRAINT ControlStock CHECK
(stock_actual > stock_minimo);
```

Ejercicio 5

```
CREATE DOMAIN CodigoCliente INT(9) CHECK (CodigoCliente > 0);

CREATE TABLE Clientes (
  CodCliente CodigoCliente,
  Nombre VARCHAR(30) NOT NULL,
  Direccion VARCHAR(30) NOT NULL,
  CodigoPostal VARCHAR(10),
  ImporteBase DEC(8,2) NOT NULL,

  CONSTRAINT PK_Clientes PRIMARY KEY (CodCliente),
  CONSTRAINT FK_Clientes_CodPostales FOREIGN KEY (CodigoPostal)
  REFERENCES CodPostales(cod_postal)
);

CREATE TABLE Facturas (
  NroFactura INT,   Fecha DATE,
  cliente_factura CodigoCliente,
  TipoDescuento DEC(8,2),
  Iva DEC(8,2),
  Importe DEC(8,2) NOT NULL,
  CONSTRAINT PK_Facturas PRIMARY KEY (NroFactura),
  CONSTRAINT FK_Facturas_Clientes FOREIGN KEY (cliente_factura)
  REFERENCES Clientes(CodCliente),
  CONSTRAINT CK_TipoDescuento CHECK (TipoDescuento BETWEEN 5 AND 20),
  CONSTRAINT CK_Iva CHECK (Iva = 15 OR Iva = 21)
);

CREATE TRIGGER Calcular_importe_factura
AFTER INSERT or UPDATE of ImporteBase ON Clientes
FOR EACH ROW
  DECLARE NuevoImporte DEC(8,2)
  DECLARE Descuento DEC(8,2)
  DECLARE Iva DEC(8,2)
  BEGIN
    Descuento = (
      SELECT TipoDescuento
      FROM Facturas
      WHERE cliente_factura = :NEW.CodCliente
    )
    Iva = (
      SELECT Iva
      FROM Facturas
      WHERE cliente_factura = :NEW.CodCliente
    )
    -- Aplico el descuento
    NuevoImporte = :NEW.ImporteBase - (:NEW.ImporteBase * Descuento / 100)
    -- Aplico el iva
    NuevoImporte = NuevoImporte + (NuevoImporte * Iva / 100)

    UPDATE Facturas
    SET Importe = NuevoImporte
    WHERE cliente_factura = :NEW.CodCliente
  END;
```

Ejercicio 6

```
CREATE TABLE Empleados_baja (
  Dni DNI,
  Legajo VARCHAR(6),
  Cargo INT,
  Usuario VARCHAR(15),
  Fecha DATE
);
```

```
CREATE TRIGGER Empleado_eliminado
AFTER DELETE ON Empleados
FOR EACH ROW
  BEGIN
    INSERT INTO Empleados_baja VALUES
    (:OLD.Dni, :OLD.Legajo, :OLD.Cargo, User, Sysdate)
  END

CREATE TRIGGER Baja_usuario
AFTER DELETE ON Empleados
FOR EACH ROW
  BEGIN
    DELETE FROM Usuarios
    WHERE :OLD.Dni = DNI
  END
```

Ejercicio 7

```
REPLACE TRIGGER Baja_usuario
AFTER DELETE ON Empleados
FOR EACH ROW
  BEGIN
    DELETE FROM Usuarios
    WHERE :OLD.Legajo = Legajo
  END

-- Desactivar un trigger
ALTER TRIGGER Calcular_importe DISABLE
-- Desactivar todos los triggers asociados a una tabla
ALTER TABLE Consumos DISABLE ALL TRIGGERS
```

Ejercicio 8

```
ALTER TABLE Facturas ADD Fecha_vto DATE
ALTER TABLE Facturas ADD Fecha_pago DATE
ALTER TABLE Facturas ADD intereses INT

CREATE TRIGGER Recargo_factura
AFTER UPDATE of Fecha_pago ON Facturas
FOR EACH ROW
  BEGIN
    IF :NEW.Fecha_pago > Fecha_vto THEN
      UPDATE Facturas
      SET intereses = 5
      WHERE :NEW.cliente_factura = cliente_factura
    ENDIF
  END
```

Ejercicio 9

```
CREATE TRIGGER Importe_recargo
AFTER UPDATE of intereses ON Facturas
FOR EACH ROW
  BEGIN
    UPDATE Facturas
    SET Importe = Importe + (Importe * intereses / 100)
    WHERE :NEW.cliente_factura = cliente_factura
  END
```

Ejercicio 10

Pendiente para que agreguen su resolución.