1. Sobre la bd banco implementa un trigger tal que imprime un mensaje en el cliente cuando alguien intenta cambiar datos en la tabla tiene.

-- Sobre la bd banco implementa un trigger tal que imprime un mensaje en el cliente cuando alguien intenta cambiar datos en la tabla tiene.

USE banco

GO

CREATE TRIGGER AVISO\_banco111

ON tiene

AFTER UPDATE, DELETE, INSERT

AS

BEGIN

SET NOCOUNT ON;--para que te no te aparezca filas afectadas--

RAISERROR('MODIFICACION DATOSpor datos',16,1)

PRINT 'cliente cambio datos'

END

GO

insert tiene (dni,cod\_cuenta)

values (117,15151515)

1. Sobre la bd compras, creamos un disparador para que no permita eliminar más de un registro a la vez de la tabla Proveedores.

--Sobre la bd compras, creamos un disparador para que no permita eliminar más de un registro a la vez de la tabla Proveedores.

use compras;

go

CREATE TRIGGER BORRAR\_REGISTROS

ON Proveedores

FOR DELETE AS --after delete tambien vale

begin

IF (SELECT COUNT(\*) FROM deleted) > 1

BEGIN

PRINT 'No permite borrar más de un registro a la vez'

ROLLBACK-- PARA NO DEJARLE EJECUTAR ESA ORDEN

end

END

GO

DELETE FROM Proveedores

WHERE emailpro IS NULL

1. Sobre la bd Jardinería crear un *TRIGGER* que bloquee actualizar el Código del producto en el proceso de actualización.

-- Sobre la bd Jardinería crear un TRIGGER que bloquee actualizar el Código del producto en el proceso de actualización.

use jardineria

go

CREATE TRIGGER bloquearActualizacion

ON Productos

for UPDATE AS --after delete tambien vale

begin

IF UPDATE(CodigoProducto)

BEGIN

PRINT 'No permite actualizar Código del producto en el proceso de actualización'

ROLLBACK transaction-- PARA NO DEJARLE EJECUTAR ESA ORDEN

end

END

GO

1. Sobre la BD Compras crear un trigger tal que cuando se vaya a introducir o modificar una línea de un artículo descatalogado no lo permita y devuelva el mensaje de error:

'No puede seleccionar un artículo descatalogado'

--Sobre la BD Compras crear un trigger tal que cuando se vaya a introducir o modificar una línea

--de un artículo descatalogado no lo permita y devuelva el mensaje de error:

--'No puede seleccionar un artículo descatalogado'

--si la fechad ee baja de articulos esta con una fehca esta descatalogado

use compras

go

create TRIGGER articulosDESCATALOGADOS

ON Lineas

AFTER UPDATE, INSERT AS

begin

IF (SELECT COUNT(\*) FROM Articulos ,inserted i WHERE Articulos.codigart = i.codigart AND fecbaja IS NOT NULL) > 0

BEGIN

PRINT 'No puede seleccionar un artículo descatalogado'

ROLLBACK transaction-- PARA NO DEJARLE EJECUTAR ESA ORDEN

end

END

GO

Corregir

use compras

go

create TRIGGER articulosDESCATALOGADOS

ON Lineas

AFTER UPDATE, INSERT AS

begin

IF (SELECT COUNT(\*) FROM Articulos ,inserted i WHERE Articulos.codigart = i.codigart AND fecbaja IS NOT NULL) > 0

BEGIN

set @erromesage = 'No puede seleccionar un artículo descatalogado'

raiserrror (@erromesage, 16.1)

ROLLBACK transaction-- PARA NO DEJARLE EJECUTAR ESA ORDEN

end

END

GO

--se `peude usar if exist

1. **Sobre la bd de compras implementar el código necesario tal que cuando se intenten vender más de la mitad de las unidades que tengamos en stock de un artículo de una vez no se permita.**

--Sobre la bd de compras implementar el código necesario tal que cuando se intenten vender más de la mitad

--de las unidades que tengamos en stock de un artículo de una vez no se permita.

use compras

go

create TRIGGER mitadUnidades

ON Articulos

AFTER UPDATE, INSERT AS

begin

IF ((SELECT stockart FROM inserted ) > ((SELECT stockart FROM inserted )/2) )

BEGIN

PRINT 'Sobrepasa la mitad de las unidades'

ROLLBACK transaction-- PARA NO DEJARLE EJECUTAR ESA ORDEN

end

END

GO

CORRECCION GEMMA

--Sobre la bd de compras implementar el código necesario tal que cuando se intenten vender más de la mitad

--de las unidades que tengamos en stock de un artículo de una vez no se permita.

use compras

go

create TRIGGER mitadUnidades

ON Articulos

AFTER UPDATE AS

begin

IF EXISTS (SELECT \* FROM inserted i JOIN deleted d ON i.codigart = d.codigart

WHERE(d.stockart - i.stockart)>d.stockart/2 AND d.stockart-i.stockart >0)

BEGIN

SET NOCOUNT ON;--para que te no te aparezca filas afectadas--

RAISERROR('Sobrepasa la mitad de las unidades',16,1)

ROLLBACK transaction-- PARA NO DEJARLE EJECUTAR ESA ORDEN

END

END

GO

--inserted valor nuevo

---deleted valor antiguo

--- valor nuevo comparado entre el valor antiguo dividido por 2

UPDATE Articulos SET stockart = 0 WHERE codigart like '0011'

go

1. Sobre la bd Compras, crear una tabla eventos con los campos, fecha char(6) y motivo varchar(200). Dicha tabla se rellenará cada vez que en la tabla artículos el stock de un producto alcanza el mínimo establecido.

--Sobre la bd Compras, crear una tabla eventos con los campos, fecha char(6) y motivo varchar(200).

-- Dicha tabla se rellenará cada vez que en la tabla artículos el stock de un producto alcanza el mínimo establecido.

use compras

GO

CREATE TABLE eventos

(

fecha date,

motivo varchar(200)

);

GO

create TRIGGER EVENTOSminimoSTCOK

ON Articulos

AFTER UPDATE, INSERT AS

begin

IF (SELECT stockart FROM inserted ) = (SELECT stockmin FROM inserted )

BEGIN

INSERT eventos (fecha,motivo )

VALUES ( GETDATE(),'Ha alcanzado el stock mínimo establecido')

end

END

GO

ESTE ESTA MAL

--Sobre la bd Compras, crear una tabla eventos con los campos, fecha char(6) y motivo varchar(200).

-- Dicha tabla se rellenará cada vez que en la tabla artículos el stock de un producto alcanza el mínimo establecido.

use compras

GO

CREATE TABLE eventos

(

fecha char(6),

motivo varchar(200)

);

GO

create TRIGGER EVENTOSminimoSTCOK

ON Articulos

AFTER UPDATE AS

begin

IF EXISTS (SELECT \* FROM inserted i JOIN deleted d ON i.codigart = d.codigart

WHERE i.stockart <= i.stockmin)

BEGIN

INSERT eventos (fecha,motivo )

VALUES ( GETDATE(),'Ha alcanzado el stock mínimo establecido')

end

END

GO

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CORRECIÓN GEMMA

--Sobre la bd Compras, crear una tabla eventos con los campos, fecha char(6) y motivo varchar(200).

-- Dicha tabla se rellenará cada vez que en la tabla artículos el stock de un producto alcanza el mínimo establecido.

use compras

GO

CREATE TABLE eventos

(

fecha char(6),

motivo varchar(200)

);

GO

create TRIGGER EVENTOSminimoSTCOK

ON Articulos

AFTER UPDATE AS

begin

IF UPDATE(stockart)

BEGIN

INSERT INTO eventos (fecha,motivo )

SELECT GETDATE(),'Ha alcanzado el stock mínimo establecido', codigart FROM inserted

WHERE stockart <= stockmin

end

END

GO

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Sobre la bd compras, cuando se quiera borra un artículo se sustituya por actualizar la fecha de baja en la tabla artículos.

--Sobre la bd compras, cuando se quiere borrar un artículo se sustituye por actualizar la fecha de baja en la tabla artículos.

1. En la bd Jardinería trigger que avisa al usuario que se ha eliminado un registro de la tabla pagos.

--En la bd Jardinería hacer un trigger que avisa al usuario que se ha eliminado un registro de la tabla pagos.

USE compras

GO

CREATE TRIGGER actualizarFECHAarticulo

ON Pagos

AFTER UPDATE, INSERT, DELETE AS

BEGIN

IF (SELECT COUNT(Pagos) FROM deleted) > 1

BEGIN

PRINT 'Se ha eliminado un registro de la tabla pagos'

END

END

GO

1. Sobre la bd Banco crear un trigger que cree un registro en la tabla numrojos con los campos id\_fila autoincremental, dni\_cliente, id\_cuenta, fecha y saldo cada vez que algún cliente se quede en números rojos en alguna de sus cuentas.

Suponer dos situaciones: que sólo se actualice la tabla cuenta de uno en uno, o que pueda actualizarse con varios a la vez.

--Sobre la bd Banco crear un trigger que cree un registro en la tabla numrojos con los

--campos id\_fila autoincremental, dni\_cliente, id\_cuenta, fecha y saldo cada vez que algún cliente se quede

--en números rojos en alguna de sus cuentas.

--Suponer dos situaciones: que sólo se actualice la tabla cuenta de uno en uno, o que pueda actualizarse con varios a la vez.

USE banco

GO

CREATE TABLE numrojos

(

id\_fila INT IDENTITY(1,1),

dni\_cliente INT,

id\_cuenta INT,

fecha DATETIME,

saldo INT

)

GO

CREATE TRIGGER numerosROJOS

ON Cuenta

AFTER UPDATE, INSERT, DELETE AS

BEGIN

--sólo se actualice la tabla cuenta de uno en uno

IF (SELECT cuenta.saldo FROM cuenta) < 0

BEGIN

PRINT 'Un cliente está en números rojos'

END

--pueda actualizarse con varios a la vez

ELSE IF (SELECT cuenta.saldo FROM cuenta) < 0

BEGIN

PRINT 'Varios clientes está en números rojos'

END

END

GO

1. Hacer lo necesario para que cada vez que un cliente de la bd banco ingrese más de 1000 euros se le bonifique con 100, sólo para clientes con cuentas que superen los tres años de antigüedad entre el 1 de enero del 2015 y el 31 de Julio del 2015. Hacerlo sobre la tabla movimiento.

Para resolverlo creo un trigger y una función( para calcular la antigüedad de la cuenta). Suponer que se produce sólo un movimiento cada vez.

1. Crear un desencadenador para la tabla productos en la bd Jardinería, que se active cada vez que se inserte un registro o se actualice la columna precio, la condición para aceptar al inserción o la actualización es que el precio costo no debe ser mayor que el precio venta.
2. Crear trigger asociado a la tabla Pedidos, de manera que cuando queramos borrar un registro de la tabla pedidos elimine todos los registros relacionados de la tabla de detallepedidos. Hay que tener en cuenta que puedo borrar varios pedidos a la vez

/\*Creo la Tabla Empleado\*/  
  
create table Empleado(  
nombre varchar(20),  
apellido varchar(20))  
  
  
/\*En esta se guardará automaticamente lo que se guardo en   
la Tabla Empleado\*/  
  
create table Empleado\_Log(  
nombre varchar(20),  
apellido varchar(20),  
direccion varchar(100),  
edad int)  
  
  
/\*Creo Trigger, se ejecutará después de insertar o actualizar la   
tabla Empleado e insertará los neuvos datos insertados en Empleado  
en la tabla Empleado\_Log\*/  
  
Create trigger trg\_Empleado\_Log  
on Empleado  
after insert,update  
as  
BEGIN  
  
 set nocount on;  
  
 declare  
 @nombre varchar(20),  
 @apellido varchar(20)  
  
 select @nombre = nombre,@apellido = apellido   
 from inserted  
  
 declare  
 @direccion varchar(100),  
 @edad int  
  
 set @direccion='Av 15, Santa Marina'  
 set @edad = 22  
  
 begin  
 insert into Empleado\_Log values(@nombre,@apellido,@direccion,@edad )  
 end  
END  
  
  
  
/\*Probando\*/  
  
insert into Empleado values('Pedro','Casas')  
select \* from Empleado  
select \* from Empleado\_Log  
  
update Empleado set nombre = 'Carlos'  
select \* from Empleado  
select \* from Empleado\_Log