Tecnológico Nacional de México Instituto Tecnológico de Culiacán



**Proyecto auditoría**

Unidad 6 Monitoreo y auditoría

**Materia:**

Administración de Bases de Datos

**Docente:**

M.C. Daniel Esparza Soto

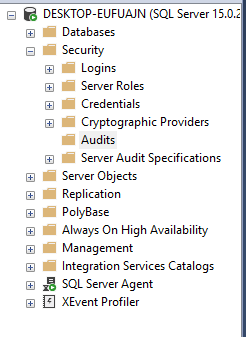
**Integrantes:** Astorga Ochoa César Alfredo Medrano Iturrios Ángel Raúl

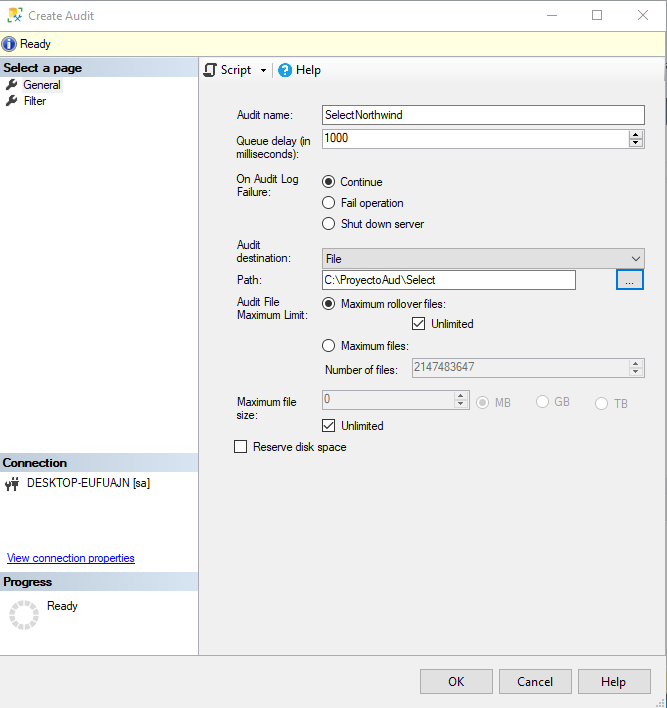
**Grupo:**

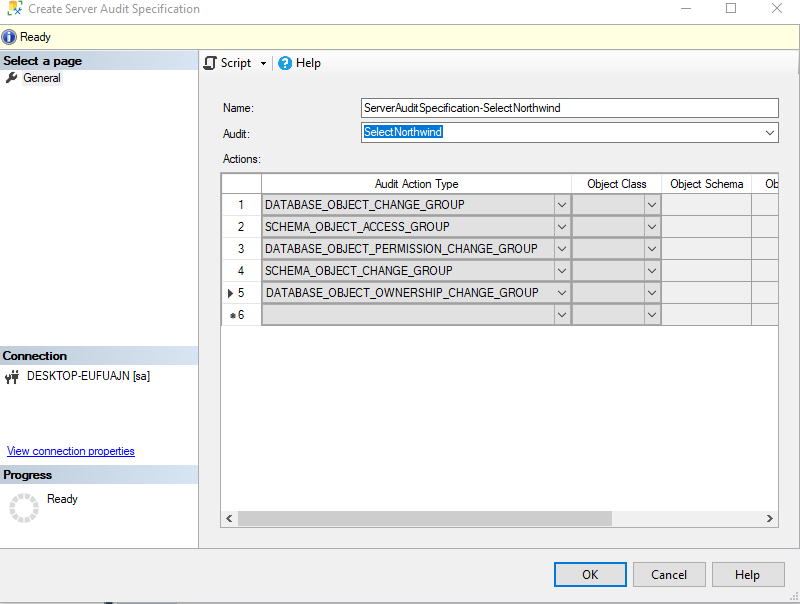
6:00 pm - 7:00 pm

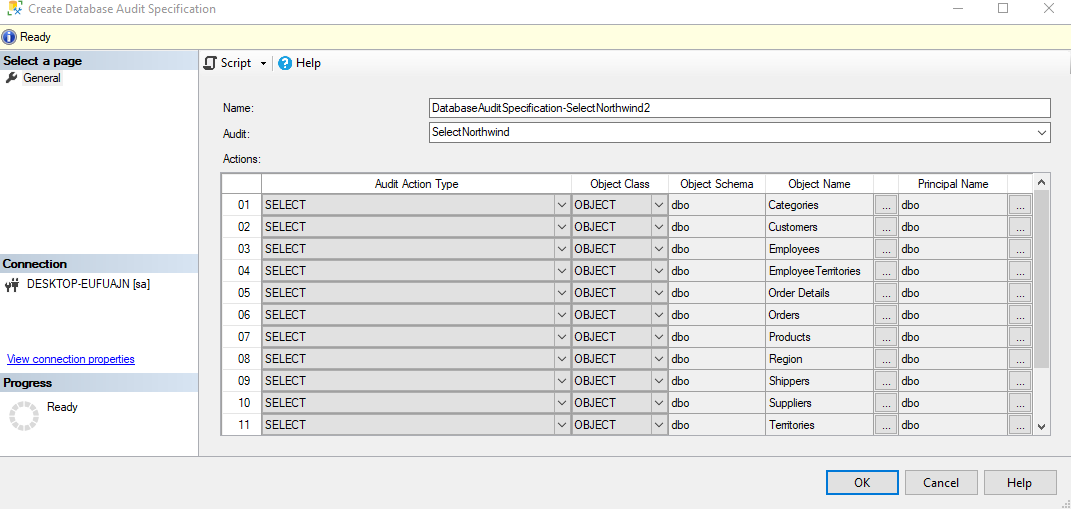
Fecha: 16 de mayo de 2022

1.- Proponer un método de Auditar el comando select sobre todas las tablas de la base de datos Norhwind. Realizar 5 consultas y exponga el método de revisión de la auditoria.









select\*from Categories select\*from Customers select\*from Employees select\*from EmployeeTerritories select\*from [Order Details] select\*from Orders

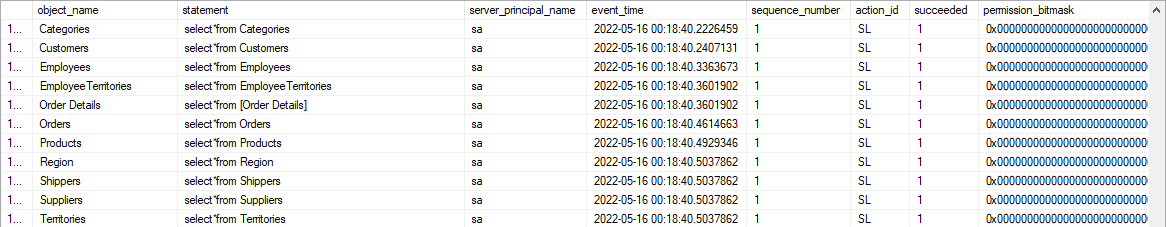
select\*from Products select\*from Region select\*from Shippers select\*from Suppliers select\*from Territories

select object\_name , statement, server\_principal\_name,\*

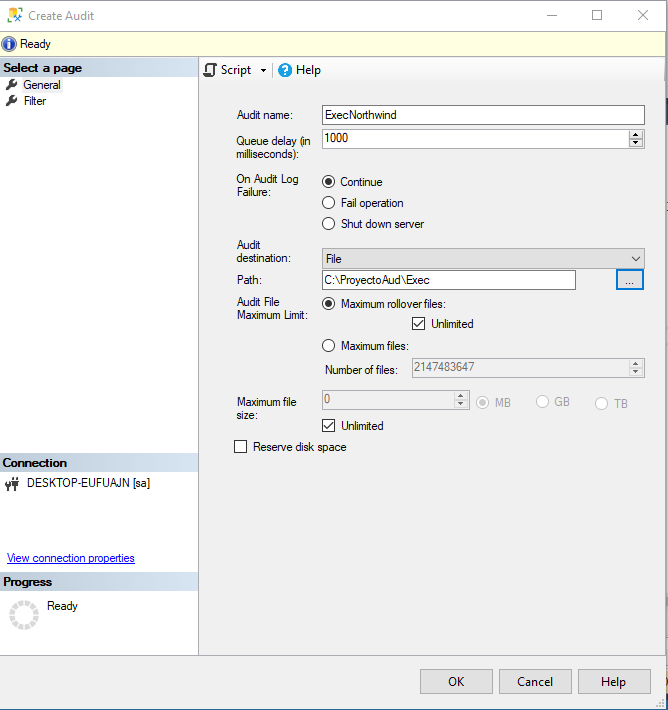
from fn\_get\_audit\_file('C:\ProyectoAud\Select\SelectNorthwind\_4DA475CE-5DFC-4E51- 9752-D8113075605A\_0\_132971326244290000.sqlaudit',

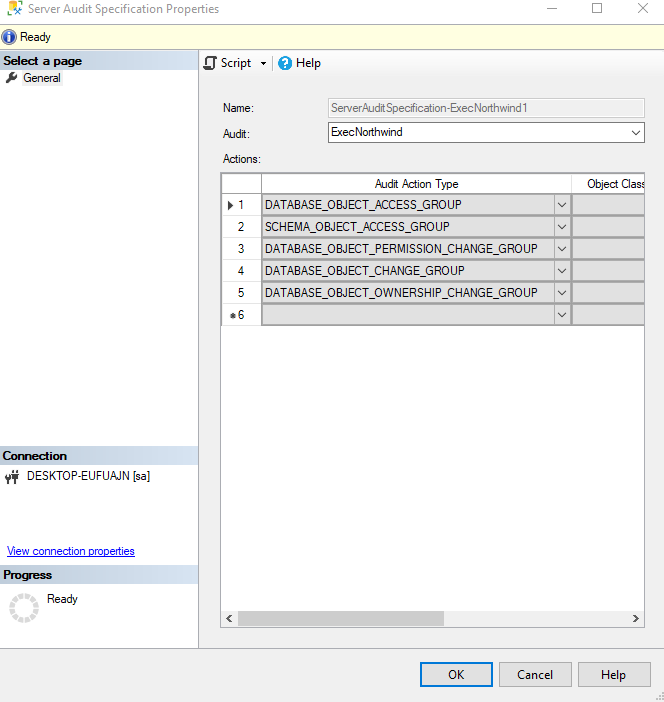
default, default)

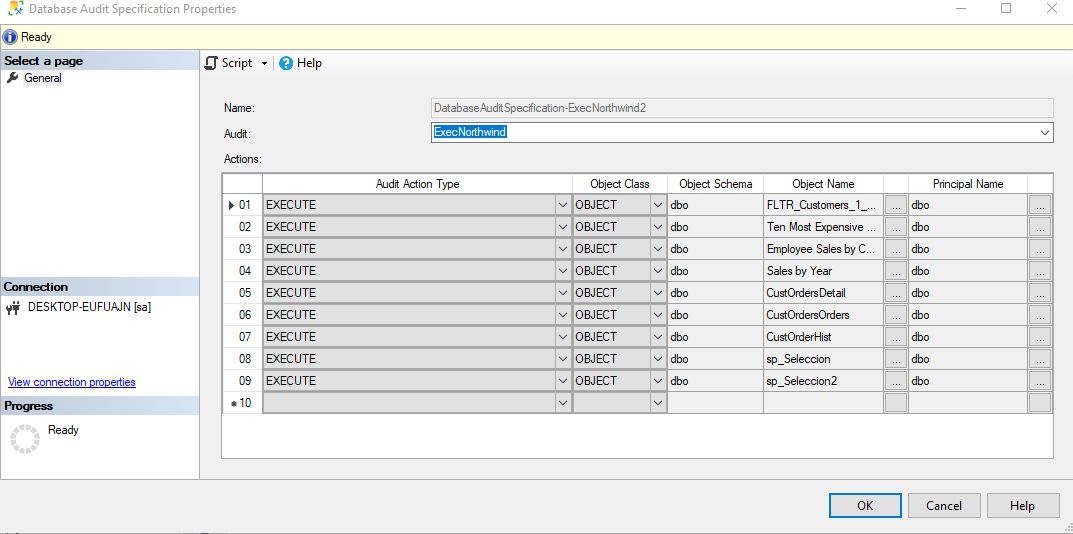
where object\_name is not null and statement like '%select%'



2.- Proponer un método para auditar la execución de todos los procedimientos almacenados de la bd Northwind. Crear dos procedimientos almacenado y exponga el método de revisión de la auditoría.







create proc sp\_Seleccion as

select\*from products where Categoryid = 1

create proc sp\_Seleccion2 as select\*from orders

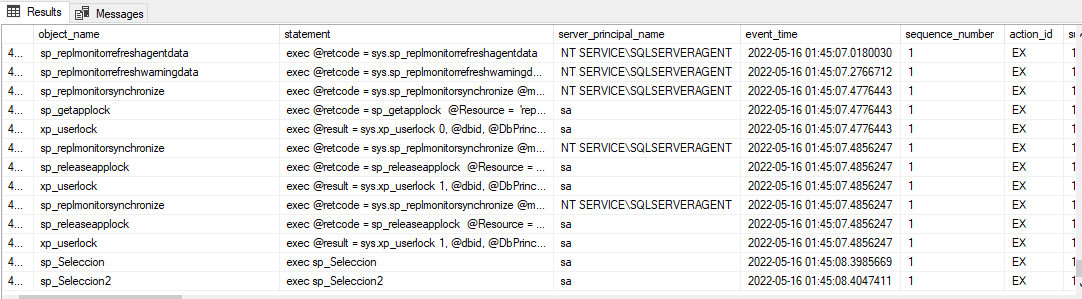
exec sp\_Seleccion exec sp\_Seleccion2

select object\_name , statement, server\_principal\_name,\*

from fn\_get\_audit\_file('C:\ProyectoAud\Exec\ExecNorthwind\_525F7D04-B524-4B90-86ED- 33C77ED78298\_0\_132971369253130000.sqlaudit',

default, default)

where object\_name is not null and statement like 'exec%'



3.- Utilizando trigger DDL a nivel servidor auditar la creación, eliminación o modificación de base de datos y de inicios de sesión. Grabar en una tabla:

* ID
* fecha del evento,
* inicio de sesión,
* acción,
* sentencia y
* detalle del evento XML.

Realizar 5 acciones sobre base de datos y 5 acciones sobre inicios de sesión.

--tabla que se llenara dentro del trigger DLL sobre base de datos create table CreEliMod(

ID int identity primary key,

Fecha datetime not null constraint DF\_LogCambiosDDL\_FechaCmabio default(getdate()), InicioSesion varchar(500) not null,

Accion varchar(500) not null, Sentencia varchar(max) not null, DetalleEvento xml not null);

go

--Trigger DLL sobre servidor: auditoria sobre creacion, eliminacion o modificacion de base de datos e inicios de sesion

create trigger tr\_AuditaCambios on all server

for

CREATE\_DATABASE, DROP\_DATABASE, ALTER\_DATABASE, CREATE\_LOGIN, DROP\_LOGIN, ALTER\_LOGIN

--DLL\_TABLE\_VIEW\_EVENTS

as begin

set nocount on;

--Declaracion de variables de proceso

declare @InfoEvento xml, @Accion varchar(500), @Sentencia varchar(max);

--1.-Captura de evento en variable principal set @InfoEvento = EVENTDATA();

--2.- Consumir informacion en variables secundarias usando XQuery

------Captura la accion DDL (si es CREATE, ALTER o DROP)

set @Accion = @InfoEvento.value('(/EVENT\_INSTANCE/EventType)[1]', 'varchar(500)');

------Captura la secuencia ejecutada

set @Sentencia = @InfoEvento.value('(/EVENT\_INSTANCE/TSQLCommand)[1]', 'nvarchar(max)')

end; go

--3.- Almacenar detalles de la accion en la tabla log

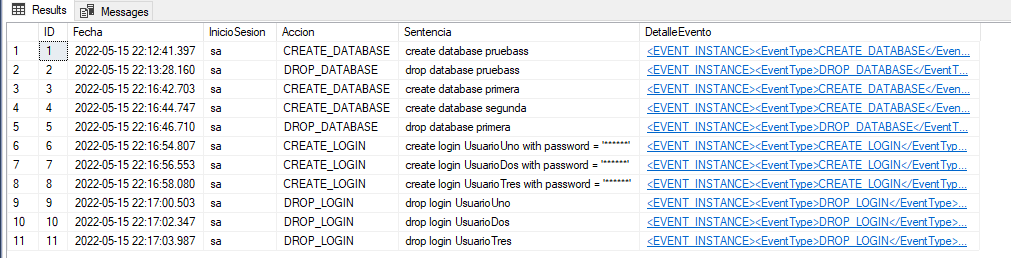
insert dbo.CreEliMod(Fecha,InicioSesion,Accion,Sentencia,DetalleEvento) values(getdate(), SUSER\_NAME(), @Accion, @Sentencia, @InfoEvento);

create database pruebass drop database pruebass create database primera create database segunda drop database primera

create login UsuarioUno with password = 'hola' create login UsuarioDos with password = 'hola' create login UsuarioTres with password = 'hola'

drop login UsuarioUno drop login UsuarioDos drop login UsuarioTres

select\*from dbo.CreEliMod



4.- Utilizando trigger DDL a nivel de base de datos auditar la creación, eliminación o modificación de tablas vistas, procedimientos almacenados, funciones y trigger. Grabar en una tabla:

* ID,
* Fecha del evento,
* inicio de sesión,
* usuario de base de datos,
* acción,
* objeto,
* sentencia y
* detalle del evento XML.

Realizar un comando para vistas, procedimientos almacenados, funciones y trigger.

--tabla que se llenara dentro del trigger DLL sobre base de datos create table CambiosDDL(

ID int identity primary key,

Fecha datetime not null constraint DF\_CambiosDDL\_FechaCambio default(getdate()), InicioSesion varchar(500) not null,

Usuario varchar(500) not null, Accion varchar(500) not null, Objeto varchar(500),

Sentencia varchar(max) not null, DetalleEvento xml not null);

go

--Trigger DLL sobre base de datos: auditoria sobre creacion, modificacion y eliminacion de tablas, vistas, procedimientos almacenados

-- funciones Y triggers

create trigger trDB\_AuditoriaCambiosA on database

for

CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE, CREATE\_VIEW, ALTER\_VIEW, DROP\_VIEW, CREATE\_PROCEDURE, ALTER\_PROCEDURE, DROP\_PROCEDURE, CREATE\_FUNCTION, ALTER\_FUNCTION, DROP\_FUNCTION,

CREATE\_TRIGGER, ALTER\_TRIGGER, DROP\_TRIGGER

as begin

set nocount on;

--Declaracion de variables de proceso

declare @InfoEvento xml, @Accion varchar(500), @Objeto varchar(500), @Sentencia varchar(max);

--1.-Captura de evento en variable principal set @InfoEvento = EVENTDATA();

--2.- Consumir informacion en variables secundarias usando XQuery

------Captura la accion DDL (si es CREATE, ALTER o DROP)

set @Accion = @InfoEvento.value('(/EVENT\_INSTANCE/EventType)[1]', 'varchar(500)');

------Captura el nombre objeto afectado, anteponiendo el nombre del esquema set @Objeto = @InfoEvento.value('(/EVENT\_INSTANCE/SchemaName)[1]',

'varchar(250)')

+'.'+@InfoEvento.value('(/EVENT\_INSTANCE/ObjectName)[1]',

'varchar(250)');

------Captura la secuencia ejecutada

set @Sentencia = @InfoEvento.value('(/EVENT\_INSTANCE/TSQLCommand)[1]', 'nvarchar(max)')

--3.- Almacenar detalles de la accion en la tabla log insert

dbo.CambiosDDL(Fecha,InicioSesion,Usuario,Accion,Objeto,Sentencia,DetalleEvento) values(getdate(), SUSER\_NAME(), CURRENT\_USER, @Accion, @Objeto, @Sentencia,

@InfoEvento); end;

go

create view vw\_AuditoriaUno as select orderid, orderdate, employeeid from orders

create proc sp\_AuditoriaUno as select\*from region

create function fn\_AuditoriaUno() returns table as

return

(select companyname from suppliers)

alter function fn\_AuditoriaUno() returns table as

return

(select ContactTitle from suppliers)

create trigger tr\_AuditoriaUno on suppliers for insert

as begin

print 'Prueba' end

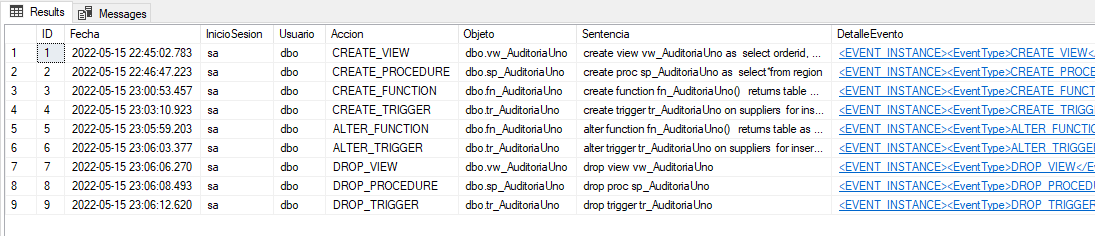
alter trigger tr\_AuditoriaUno on suppliers for insert

as begin

print 'Auditoria' end

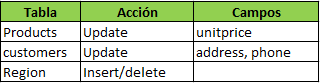
drop view vw\_AuditoriaUno drop proc sp\_AuditoriaUno

drop function fn\_AuditoriaUno() drop trigger tr\_AuditoriaUno select\*from dbo.CambiosDDL



5.- Auditar los cambios de las tablas de la base de datos Northwind, grabar en una tabla los campos:

* TipoTransaccion: insert, update, delete
* Nombre de la Tabla
* Campo de la llave primaria
* Campo Modificado
* Valor Original
* Valor Nuevo
* Fecha de la transacción
* Nombre de Usuario



create table TipoCambios ( TipoTran varchar(6), Tabla varchar(128),

PK varchar(1000), CampoModificado varchar(128), ValorOrginal Varchar(1000), ValorNuevo Varchar(1000),

FechaTran DateTime Not null constraint DF\_TipoCambios\_FechaTran DEFAULT (gETdATE()), Usuario varchar(128)

)

--Products, Accion = Update, Campos = Unitprice

create trigger tr\_ProductsCambios on products for update as

begin

declare @TipoTran varchar(6), @Tabla varchar(128), @PK varchar(1000), @CampoModificado varchar(128), @ValorOriginal varchar(1000), @ValorNuevo varchar(1000), @FechaTran datetime, @Usuario varchar(128)

if exists(select\*from inserted)

if exists(select\*from deleted) -- Si es un update select @TipoTran = 'Update'

else --Si es un insert

select @TipoTran = 'Insert'

else --Si es un delete

select @TipoTran = 'Delete' select @Tabla = 'Products'

select @Usuario = system\_user select @FechaTran = getdate()

--Validar si se cambia el campo UnitPrice if update(UnitPrice)

begin

select @ValorNuevo = convert(varchar(1000),UnitPrice) from inserted select @ValorOriginal = convert(varchar(1000),UnitPrice) from deleted end

if exists(select\*from inserted)

select @PK = 'Clave='+convert(varchar(1000),ProductID) from inserted else

select @PK = 'Clave='+convert(varchar(1000),ProductID) from deleted

insert TipoCambios (TipoTran,Tabla,PK,CampoModificado,ValorOrginal,ValorNuevo,FechaTran,Usuario) values (@TipoTran,@Tabla,@PK,'UnitPrice',@ValorOriginal,@ValorNuevo,@FechaTran,@Usuario)

end

select\*from products

update products set unitprice = 20 where productid = 1 select\*from tipoCambios

--Customers, Accion = update, Campos = Address, Phone create trigger tr\_CustomersCambios on Customers for update as

begin

declare @TipoTran varchar(6), @Tabla varchar(128), @PK varchar(1000), @CampoModificado varchar(128), @ValorOriginal varchar(1000), @ValorNuevo varchar(1000), @FechaTran datetime, @Usuario varchar(128)

if exists(select\*from inserted)

if exists(select\*from deleted) -- Si es un update select @TipoTran = 'Update'

else --Si es un insert

select @TipoTran = 'Insert'

else --Si es un delete

select @TipoTran = 'Delete'

select @Tabla = 'Customers' select @Usuario = system\_user select @FechaTran = getdate()

--Validar si se cambia el campo Address if update(Address)

begin

select @CampoModificado = 'Address'

select @ValorNuevo = convert(varchar(1000),Address) from inserted select @ValorOriginal = convert(varchar(1000),Address) from deleted end

--Validar si se cambia el campo Phone else if update(Phone)

begin

select @CampoModificado = 'Phone'

select @ValorNuevo = convert(varchar(1000),Phone) from inserted select @ValorOriginal = convert(varchar(1000),Phone) from deleted end

if exists(select\*from inserted)

select @PK = 'Clave='+convert(varchar(1000),CustomerID) from inserted else

select @PK = 'Clave='+convert(varchar(1000),CustomerID) from deleted

insert TipoCambios (TipoTran,Tabla,PK,CampoModificado,ValorOrginal,ValorNuevo,FechaTran,Usuario) values

(@TipoTran,@Tabla,@PK,@CampoModificado,@ValorOriginal,@ValorNuevo,@FechaTran,@Usuari o)

end

select\*from customers

update customers set address = 'Prueba 1' where CustomerID = 'ANATR' update customers set Phone = '(171) 5555-77788' where CustomerID = 'ALFKI'

select\*from TipoCambios

--Region, Accion = Insert, Delete

create trigger tr\_RegionCambios on Region for Insert, Delete as

begin

declare @TipoTran varchar(6), @Tabla varchar(128), @PK varchar(1000), @CampoModificado varchar(128), @ValorOriginal varchar(1000), @ValorNuevo varchar(1000), @FechaTran datetime, @Usuario varchar(128)

if exists(select\*from inserted)

if exists(select\*from deleted) -- Si es un update select @TipoTran = 'Update'

else --Si es un insert

select @TipoTran = 'Insert'

else --Si es un delete

select @TipoTran = 'Delete'

select @Tabla = 'Region' select @Usuario = system\_user select @FechaTran = getdate()

select @CampoModificado = ' '

select @ValorNuevo = convert(varchar(1000),RegionDescription) from inserted select @ValorOriginal = convert(varchar(1000),RegionDescription) from deleted

if exists(select\*from inserted)

select @PK = 'Clave='+convert(varchar(1000),RegionID) from inserted else

select @PK = 'Clave='+convert(varchar(1000),RegionID) from deleted

insert TipoCambios (TipoTran,Tabla,PK,CampoModificado,ValorOrginal,ValorNuevo,FechaTran,Usuario)

values (@TipoTran,@Tabla,@PK,@CampoModificado,@ValorOriginal,@ValorNuevo,@FechaTran,@Usuari o)

end

select\*from region

insert region values(5, 'Prueba') delete region where RegionID = 5

select\*from TipoCambios

