use AdventureWorks

--Traigo esto de la clase anterior

--para poder usar la tabla Productos

drop table if exists Productos

go

select ProductID Producto, ListPrice Precio into Productos from Production.Product

go

select \* from Productos

go

--

--Clase 07--

--Transacciones--

--Operación que produce una modificación en la DB

--(por lo general en una tabla) (insert, update, delete)

--(transacciones implícitas/atómicas)

drop table if exists productos2

select \* into productos2 from Productos

select \* from productos2

--Transacciones atómicas/implicitas

update productos2 set Precio = 500

insert into productos2 values(3000),(2000),(1000)

delete from productos2 where Producto < 996

--Explícita

--(necesito decirle al servidor que estoy dentro

--de una transacción explícita, y la misma tendra

--dentro una o varias trans. implícitas)

--(lo guarda en un espacio intermedio de la memoria

--del servidor, cuando las consulto se lee el valor

--pero la transacción sigue abierta)

--(Mientras la transacción no se confirme, los datos

--no pasan de la tabla intermedia a la tabla física

--de la DB)

begin transaction

update productos2 set Precio = 500

select \* from productos2

--commit (cierra la transacción, confirmando que todas

--las trans. implícitas que tengo dentro de la explícita

--confirmen los cambios hechos)

--rollback (en cambio roolback, cierra también la transacción

--pero deshace todas las trans. implícitas que están dentro

--de la transacción explícita)(sirve como mecanismo de seguridad)

--TransaciónVenta

/\*

-verificar stock

-si no existe el cliente, lo inserto

-si existe, agrego la venta al historial del cliente

-genero factura

-genero remito

-genero registro nuevo en tabla de centas

-actualizo stock

-si stock < minimo recomendado >> alerta a traves

de trigger que envie mail a deposito

\*/

drop table if exists articulos

go

drop table if exists articulos\_seg

go

create table articulos(

codigo int,

nombre varchar(50),

precio money)

go

create table articulos\_seg(

codigo\_producto int,

usuario varchar(50))

go

select \* from articulos

select \* from articulos\_seg

--Puntos de restauración en una transacción

begin transaction

insert into articulos values (1, 'Tv Led 43"', 0)

save transaction alta\_articulo --punto restauración

insert articulos\_seg values (1, SUSER\_NAME())

rollback transaction alta\_articulo

update articulos set precio = 40000 where codigo = 1

commit

--

--Laboratorio: AdventureWorks--

--Transacciones--

--1. Borrar todos los productos que no se hayan

--vendido y luego revertir la operación.

--Tablas: Production.Product, Sales.SalesOrderDetail

--Campos: ProductID

begin transaction

delete p from Production.Product p where not exists

(select \* from Sales.SalesOrderDetail sod where p.ProductID = sod.ProductID)

rollback

select ProductID from Production.Product where ProductID not in

(select ProductID from Sales.SalesOrderDetail)

--2. Incrementar el precio a 200 para todos los productos cuyo

--precio sea igual a cero y confirmar la transacción.

--Tablas: Production.Product

--Campos: ListPrice

begin transaction

update Production.Product set ListPrice = 200 where ListPrice = 0

rollback--pruebas

commit--uso commit

select ListPrice, ProductID from Production.Product

--3. Obtener el promedio del listado de precios y guardarlo en una variable

--llamada @Promedio. Incrementar todos los productos un 15% pero si

--el precio mínimo no supera el promedio revertir toda la operación.

--Tablas: Production.Product

--Campos: ListPrice

begin transaction

declare @promedio money

select @promedio = avg(ListPrice) from Production.Product

update Production.Product set ListPrice = ListPrice \* 1.15

select ListPrice, @promedio [Promedio] from Production.Product

if (select min(ListPrice) from Production.Product) < @promedio

rollback

else

commit

select ListPrice from Production.Product

go

--Errores--

--Try/Catch

begin try

select 15/0 error

end try

begin catch

select ERROR\_NUMBER() numero\_error,

ERROR\_STATE() estado,

ERROR\_SEVERITY() gravedad,

ERROR\_PROCEDURE() procedimiento,

ERROR\_LINE() linea,

ERROR\_MESSAGE() mensaje

end catch

drop table if exists errores

go

create table errores(

id int identity,

usuario varchar(50),

numero\_error int,

estado int,

gravedad int,

linea int,

procedimiento varchar(max),

mensaje varchar(max),

fecha datetime

)

go

drop table if exists ventas

go

create table ventas(

id int identity,

producto int not null,

vendedor int not null,

cantidad int not null check(cantidad > 0)

)

go

alter table productos2 add primary key(producto)

go

alter table ventas add foreign key (producto)

references productos2(producto) on update cascade

go

alter table ventas add foreign key (vendedor)

references Sales.SalesPerson(BusinessEntityID) on update cascade

go

select \* from ventas

drop procedure if exists agregarVenta

go

create procedure agregarVenta @cod\_producto int, @cod\_vendedor int, @p\_cantidad int

as

begin try

insert ventas values(@cod\_producto, @cod\_vendedor, @p\_cantidad)

end try

begin catch

insert errores values(

SUSER\_NAME(),

ERROR\_NUMBER(),

ERROR\_STATE(),

ERROR\_SEVERITY(),

ERROR\_LINE(),

ERROR\_PROCEDURE(),

ERROR\_MESSAGE(),

GETDATE()

)

end catch

go

select top 5 \* from productos2

select top 5 \* from Sales.SalesPerson

exec dbo.agregarVenta 2, 274, 100

exec dbo.agregarVenta 1000, 274, 100

exec dbo.agregarVenta 3, 1274, 100

exec dbo.agregarVenta 3, 274, 0

select \* from ventas

select \* from errores

select \* from sys.messages order by message\_id

--Cursores--

select \* from Sales.SalesOrderDetail

where ModifiedDate between '2005-07-15' and '2005-07-31'

go

---

--Cursor

declare @id uniqueidentifier

declare cursor\_ventas cursor for

select rowguid

from Sales.SalesOrderDetail

where ModifiedDate between '2005-07-15' and '2005-07-31'

open cursor\_ventas

fetch next from cursor\_ventas into @id

while @@FETCH\_STATUS = 0

begin

select \* from Sales.SalesOrderDetail

where rowguid = @id

fetch next from cursor\_ventas into @id

end

close cursor\_ventas

deallocate cursor\_ventas

go