TRANSACCIONES

Si todas las consutas se ejecutan correctamente se hace el cambio pero si una de las consutas es errónea, se hace un rollback revirtiendo todos los cambios que se habían hecho. Quiere decir que los datos van a estar como estaban antes de ejecutar la query.

Transacciones

- Una transacción es un conjunto de instrucciones que, por su naturaleza, deben ejecutarse como una unidad.
- Lo anteriormente mencionado responde a la propiedad de **Atomicidad** de las transacciones. Es decir, se ejecutan todas las consultas correctamente ó no se ejecuta ninguna.
- Tiene mecanismos para indicar que la transacción debe convalidarse (*commit)* ó, tras un error, debe ser revertida (*rollback).*
- Al revertir una transacción, todos los datos vuelven al estado previo a la ejecución de la transacción.

Ejemplo de atomicidad

Sin transacción	Con transacción
	Inicio de transacción
INSERT en tabla A ✔	INSERT en tabla A ✔
UPDATE en tabla B ✔	UPDATE en tabla B ✔
DELETE en tabla C ✘	DELETE en tabla C ✘
	Reversión de la transacción
Los cambios en tabla A y B ya fueron registrados. El registro de la tabla C no fue eliminado	Los cambios en tabla A y B no se registrarán. El registro de la tabla C no fue eliminado

```
ALTER PROCEDURE SP_AGREGAR_VENTA(
    @DNI VARCHAR(10),
    @IDARTICULO BIGINT,
    @CANTIDAD INT
)
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION



    END TRY
    BEGIN CATCH

    END CATCH
END

SELECT * FROM ARTICULOS
```

```
    BEGIN TRANSACTION

    DECLARE @PU MONEY
    SELECT @PU = PRECIO FROM ARTICULOS WHERE IDARTICULO = @IDARTICULO

    INSERT INTO VENTAS (DNI, IDARTICULO, CANTIDAD, FECHA, IMPORTE)
    VALUES (@DNI, @IDARTICULO, @CANTIDAD, GETDATE(), @PU*@CANTIDAD)

    UPDATE ARTICULOS SET STOCK = STOCK - @CANTIDAD WHERE IDARTICULO = @IDARTICULO

    COMMIT TRANSACTION Finaliza la transaccion, si no hay errores --> gemeramos el commi

END TRY
BEGIN CATCH
    ROLLBACK TRANSACTION Por tener 1 error viene por el catch, realiza el rollback y muesta el msj RAISERRO
    RAISERROR('NO SE PUDO AGREGAR LA VENTA', 16, 1)
END CATCH
```

Ejecutamos un Store Procedure que tiene una transacción y si ejecuta correctamente esa transacción ejecuta otra transacción en la misma operación:

```
BEGIN TRANSACTION

DECLARE @PU MONEY
SELECT @PU = PRECIO FROM ARTICULOS WHERE IDARTICULO = @IDARTICULO

INSERT INTO VENTAS (DNI, IDARTICULO, CANTIDAD, FECHA, IMPORTE)
VALUES (@DNI, @IDARTICULO, @CANTIDAD, GETDATE(), @PU*@CANTIDAD)

--UPDATE ARTICULOS SET STOCK = STOCK - @CANTIDAD WHERE IDARTICULO = @IDARTICULO
EXEC SP_DESCONTAR_STOCK @IDARTICULO, @CANTIDAD

COMMIT TRANSACTION
```

Reemplazamos UPDATE por EXEC para modificar el Stock

Continuación:

```
CREATE PROCEDURE SP_DESCONTAR_STOCK(
    @IDARTICULO BIGINT,
    @CANTIDAD INT
)
AS
BEGIN
    BEGIN TRY
    END TRY
    BEGIN CATCH
    END CATCH
END
```

```
CREATE PROCEDURE SP_DESCONTAR_STOCK(
    @IDARTICULO BIGINT,
    @CANTIDAD INT
)
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION

        COMMIT TRANSACTION
    END TRY
    BEGIN CATCH
        ROLLBACK TRANSACTION
        RAISERROR('NO SE PUDO DESCONTAR EL STOCK', 16, 1)
    END CATCH
```

```
CREATE PROCEDURE SP_DESCONTAR_STOCK(
    @IDARTICULO BIGINT,
    @CANTIDAD INT
)
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION

            UPDATE ARTICULOS SET STOCK = STOCK - @CANTIDAD WHERE IDARTICULO = @IDARTICULO

        COMMIT TRANSACTION
    END TRY
    BEGIN CATCH
        ROLLBACK TRANSACTION
```

Ejecutamos la query pasándole esos parámetros:

```
EXEC SP_AGREGAR_VENTA 10, 13, 1
```