Transact-SQL y Transacciones

TCL (Transaction Control Language)

El **TCL (Transaction Control Language)** en SQL Server se utiliza para gestionar las transacciones dentro de una base de datos. Las transacciones permiten garantizar la integridad y consistencia de los datos mediante comandos como COMMIT, ROLLBACK y SAVEPOINT.

- **COMMIT**: Guarda permanentemente los cambios realizados en una transacción.

```
COMMIT;
```

- **ROLLBACK**: Revierte todos los cambios realizados en la transacción desde el último COMMIT o SAVEPOINT.

```
ROLLBACK;
```

- **SAVEPOINT**: Crea un punto de restauración dentro de una transacción.

```
SAVEPOINT nombre_punto;
```

Elementos Auxiliares

Comando USE

El comando USE se utiliza para seleccionar la base de datos en la que se desea trabajar.

- **Ejemplo**:

```
USE MiBaseDeDatos;
```

Variables

En Transact-SQL, se pueden usar variables para almacenar valores temporales. Estas variables pueden ser definidas por el usuario o ser variables del sistema.

- **Variables definidas por el usuario**: Se definen con DECLARE y se les asigna un valor con SET.

```
DECLARE @Edad INT;
SET @Edad = 30;
```

- **Variables del sistema**: Son proporcionadas por SQL Server y contienen información sobre el sistema.

```
SELECT @@VERSION;
```

GO

El comando GO no es una instrucción de Transact-SQL, sino un comando de **SSMS** (SQL Server Management Studio) que indica el final de un bloque de instrucciones. Permite ejecutar un grupo de comandos como una única unidad.

- **Ejemplo**:

```
SELECT * FROM Empleados;
GO
```

EXECUTE

El comando EXECUTE (o EXEC) se utiliza para ejecutar procedimientos almacenados, funciones o comandos dinámicos.

- **Ejemplo**:

```
EXECUTE MiProcedimiento;
```

PRINT

El comando PRINT se utiliza para mostrar mensajes en la ventana de salida de SSMS.

- **Ejemplo**:

```
PRINT 'Transacción realizada con éxito';
```

Estructuras de Control

IF...ELSE

La estructura IF...ELSE permite ejecutar una acción u otra dependiendo de una condición.

- **Ejemplo**:

```
IF @Edad > 18
    PRINT 'Mayor de edad';
ELSE
    PRINT 'Menor de edad';
```

CASE

La expresión CASE es una forma de realizar múltiples comparaciones dentro de una consulta.

- **Ejemplo**:

```
SELECT Nombre,
       CASE
          WHEN Edad >= 18 THEN 'Adulto'
          ELSE 'Menor'
       END AS Categoria
FROM Empleados;
```

WHILE

El bucle WHILE ejecuta un bloque de código mientras una condición sea verdadera.

- **Ejemplo**:

```
DECLARE @Contador INT = 1;
WHILE @Contador <= 5
BEGIN
    PRINT @Contador;
    SET @Contador = @Contador + 1;
END
```

GOTO

El comando GOTO se utiliza para saltar a una etiqueta dentro de un bloque de código.

- **Ejemplo**:

```
GOTO etiqueta;

etiqueta:
PRINT 'Esto se imprimirá después del GOTO';
```

Funciones y Procedimientos Almacenados

Funciones definidas en el sistema

SQL Server proporciona una serie de funciones del sistema que devuelven información sobre el entorno o los objetos de la base de datos.

- **Ejemplo**: GETDATE() devuelve la fecha y hora actual del sistema.

```
SELECT GETDATE();
```

Procedimientos definidos en el sistema

SQL Server incluye procedimientos almacenados del sistema que realizan diversas tareas de administración y gestión.

- **Ejemplo**: sp_help muestra información sobre un objeto de la base de datos.

```
EXEC sp_help 'Empleados';
```

Cursores de Transact-SQL

Un **cursor** es una herramienta en Transact-SQL que permite recuperar y manipular fila por fila los resultados de una consulta.

Funciones de Cursores

- **FIRST**: Recupera el primer valor de un conjunto de resultados.
- **LAST**: Recupera el último valor de un conjunto de resultados.

Tipos de Cursores

Existen varios tipos de cursores que definen cómo se manejan los resultados y cómo se mueven entre las filas. Los más comunes son los cursores **estáticos** y **dinámicos**.

- **Ejemplo**:

```
DECLARE cursorEjemplo CURSOR FOR
SELECT Nombre FROM Empleados;
OPEN cursorEjemplo;
FETCH NEXT FROM cursorEjemplo INTO @Nombre;
```

Try/Catch

La estructura TRY...CATCH se utiliza para capturar y manejar errores en Transact-SQL.

- **Ejemplo**:

```
BEGIN TRY
    -- Intentar ejecutar el código
    DECLARE @Valor INT = 1 / 0;
END TRY
BEGIN CATCH
    -- Captura el error
    PRINT 'Se ha producido un error: ' + ERROR_MESSAGE();
END CATCH;
```

Triggers (Desencadenadores)

Un **trigger** (o desencadenador) es un tipo especial de procedimiento almacenado que se ejecuta automáticamente en respuesta a eventos como **DML** (Data Manipulation Language), **DDL** (Data Definition Language) o **Logon**.

DML Trigger

Un **trigger DML** se activa cuando se realizan operaciones como INSERT, UPDATE o DELETE en una tabla.

- **Ejemplo**:

```
CREATE TRIGGER trg_AfterInsert
ON Empleados
AFTER INSERT
AS
BEGIN
    PRINT 'Nuevo empleado insertado';
END;
```

DDL Trigger

Un **trigger DDL** se activa en respuesta a cambios en la definición de la base de datos, como la creación o eliminación de objetos.

- **Ejemplo**:

```
CREATE TRIGGER trg_AfterCreate
ON DATABASE
FOR CREATE_TABLE
AS
BEGIN
    PRINT 'Nueva tabla creada';
END;
```

Logon Trigger

Un **trigger de Logon** se activa cuando un usuario se conecta a la base de datos.

- **Ejemplo**:

```
CREATE TRIGGER trg_Logon
ON ALL SERVER
FOR LOGON
AS
BEGIN
    PRINT 'Usuario conectado';
END;
```