

Capítulo X

Introducción a la Programación con Transact SQL

En este capítulo veremos cuáles son los elementos que debemos conocer para programar en Transact-SQL, el lenguaje SQL de Microsoft basado en el estándar ANSI SQL.

Transact-SQL es la implementación Microsoft del estándar ANSI SQL que define elementos del lenguaje SQL que pueden ejecutarse desde cualquier aplicación frontal. Además de los elementos del estándar ANSI SQL, Transact-SQL tiene elementos del lenguaje que son exclusivos de él conocidos como **extensiones Transact-SQL**. Estos, son mejoras a algunas instrucciones del estándar ó añadidos que proporcionan nuevas funcionalidades.

Al escribir y ejecutar sentencias Transact-SQL usará:

- Declaraciones del **Data Manipulation Language (DML)** que se usan para consultar y modificar los datos.
- Declaraciones del **Data Definition Language (DDL)** que se usan para crear los objetos en la base de datos.
- Declaraciones del **Data Control Language (DCL)** que se utilizan para determinar quién ve o modifica los datos.
- Instrucciones de transacción que se utilizan para el control de las transacciones.
- Elementos adicionales del language como: variables, operadores, funciones, sentencias de control de flujo, y comentarios.

1. DECLARACIONES DEL DATA MANIPULATION LANGUAGE - DML

Las declaraciones DML están constituidas por las instrucciones que permiten trabajar con los datos e incluyen a las siguientes:

Instrucción	Descripción
SELECT	Selecciona filas y columnas de una ó más tablas de la base de datos.
INSERT	Añade una nueva fila a una tabla.
UPDATE	Modifica los datos existentes en una tabla.
DELETE	Elimina filas de una tabla.
MERGE	Inserta, actualiza o elimina en una tabla destino en base al resultado de la combinación con una tabla origen.
BULK INSERT	Importa un archivo de datos en una tabla o vista.
TRUNCATE TABLE	Elimina todas las filas de una tabla sin registrar la operación en el registro de transacciones.

2. DECLARACIONES DEL DATA DEFINITION LANGUAGE - DDL

Las declaraciones del DDL permiten crear bases de datos, tablas, tipos de datos definidos por usuarios, y otros objetos de la base de datos. También se usan para manejar los objetos de la base de datos.

Las siguientes son declaraciones del DDL:

Instrucción	Descripción
CREATE <i>tipo_objeto</i>	Crea un objeto de la base de datos.
ALTER <i>tipo_objeto</i>	Modifica la definición de un objeto de la base de datos.
DROP <i>tipo_objeto</i>	Elimina un objeto de la base de datos.
ENABLE TRIGGER	Habilita un desencadenante.
DISABLE TRIGGER	Deshabilita un desencadenante.
UPDATE STATISTICS	Actualiza las estadísticas de optimización de consultas para una tabla o vista indexada.

3. DECLARACIONES DEL DATA CONTROL LANGUAGE - DCL

Las declaraciones del DCL se usan para cambiar los permisos o roles asociados con un usuario de la base de datos. La tabla siguiente describe las declaraciones de DCL.

Instrucción	Descripción
GRANT	Crea una entrada en la seguridad del sistema que le permite a un usuario trabajar con los datos o ejecutar ciertas sentencias Transact-SQL.
DENY	Crea una entrada en la seguridad del sistema negando un permiso de una cuenta de seguridad y evita que el usuario, grupo, o rol herede el permiso a través de su grupo y rol.
REVOKE	Quita un permiso previamente concedido o negado.

4. INSTRUCCIONES DE TRANSACCIÓN

Una transacción es un conjunto de operaciones de modificaciones de datos que debe ejecutarse como una unidad para que no se generen inconsistencias en los datos de la base de datos. Las instrucciones de transacción permiten controlar la culminación exitosa o con errores de una transacción.

Las siguientes son las instrucciones de transacción de Transact-SQL:

Instrucción	Descripción
BEGIN TRANSACTION	Marca el punto de inicio de una transacción explícita.
COMMIT TRANSACTION ó COMMIT WORK	Marca el final sin errores de una transacción.
ROLLBACK TRANSACTION ó ROLLBACK WORK	Revierte una transacción hasta el inicio ó hasta un punto de retorno dentro de la transacción.
SAVE TRANSACTION	Establece un punto de retorno dentro de una transacción.
BEGIN DISTRIBUTED TRANSACTION	Marca el punto de inicio de una transacción distribuida.

5. VARIABLES

Una variable es un elemento del lenguaje que se define en la memoria del sistema, y al que se puede asignar un valor que se utilizará posteriormente. Una variable puede ser local o global.

5.1. Variables locales – Las instrucciones DECLARE y SET

Una variable local es creada por el usuario (el programador) y se define en una sentencia DECLARE, se le asigna un valor inicial en una instrucción SET, y se utiliza dentro de la declaración, batch, o procedimiento en la que fue declarada. Una variable local se identifica con un símbolo @ que precede a su nombre.

Sintaxis

```
DECLARE @nombre_variable tipo_dato , ...
```

```
SET @nombre_variable = expresión
```

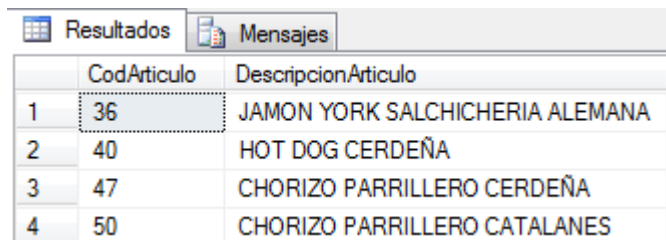
Ejercicio 10.1: Uso de una variable local

Este ejercicio declara una variable local **@proveedor**, le asigna un valor de tipo cadena usando el comodín %, y consulta la base de datos para buscar los artículos del proveedor cuyo **nomProveedor** contiene la cadena **El Gordito**.

```
USE QhatuPERU
go

DECLARE @proveedor varchar(40)
SET @proveedor = '%El Gordito%'

SELECT ARTICULO.CodArticulo,
       ARTICULO.DescripcionArticulo
FROM ARTICULO INNER JOIN PROVEEDOR
  ON ARTICULO.CodProveedor =
     PROVEEDOR.CodProveedor
WHERE PROVEEDOR.NomProveedor LIKE @proveedor
go
```



The screenshot shows a SQL Server interface with two tabs: 'Resultados' (Results) and 'Mensajes' (Messages). The 'Resultados' tab is active, displaying a table with 4 rows and 2 columns: 'CodArticulo' and 'DescripcionArticulo'. The first row is highlighted with a dashed border.

	CodArticulo	DescripcionArticulo
1	36	JAMON YORK SALCHICHERIA ALEMANA
2	40	HOT DOG CERDEÑA
3	47	CHORIZO PARRILLERO CERDEÑA
4	50	CHORIZO PARRILLERO CATALANES

Ejercicio 10.2: Uso de SELECT para entregar un valor a una variable local

```
USE QhatuPERU
go

-- asignar valor a variable usando SELECT
DECLARE @articulo varchar(40)
SELECT @articulo = 'COREDISE'
SELECT @articulo AS Nombre
go
```

Resultados Mensajes	
Nombre	
1	COREDISE

Si bien es posible asignar un valor a una variable con la instrucción SELECT, se recomienda utilizar para ello la instrucción SET. Utilice SELECT cuando necesite entregar a una variable un valor retornado por una consulta.

En el siguiente ejemplo se entrega a una variable el resultado de ejecutar una consulta.

```
-- asignar valor a variable
-- con el resultado de una consulta
DECLARE @articulo varchar(40)
SELECT @articulo = DescripcionArticulo
                FROM ARTICULO
SELECT @articulo AS Nombre
go
```

Resultados Mensajes	
Nombre	
1	YOGURT YOLEIT FRESA

La consulta lee la columna **DescripcionArticulo** de la tabla ARTICULO. A la variable se le entrega el último valor leído.

Si la consulta no recupera datos, a la variable se le asigna el valor NULL.

```
DECLARE @articulo varchar(40)
SELECT @articulo = (SELECT DescripcionArticulo
                FROM ARTICULO
                WHERE CodProveedor = 999)
SELECT @articulo AS Nombre
go
-- la consulta no encuentra datos.
-- se le asigna el valor NULL a la variable
```

Resultados Mensajes	
Nombre	
1	NULL

5.2. Variables globales

Las variables globales son predefinidas y mantenidas por SQL Server. El usuario no puede asignar o cambiar directamente los valores de las variables globales. Muchas de las variables globales reportan al sistema la actividad que ha tenido lugar desde la última vez que SQL Server fue iniciado, otras reportan información sobre una conexión. Una variable global es identificada con dos símbolos @ precediendo su nombre.

Las variables globales son útiles para verificar la seguridad o condiciones del entorno actual de SQL Server. Por ejemplo, si quiere verificar el número de versión de SQL Server, puede realizar la consulta a la variable global @@version. La consulta devuelve el número de la versión del SQL Server que está ejecutando.

Ejercicio 10.3: Uso de una variable global

Este ejercicio utiliza la variable global @@ROWCOUNT para averiguar el número de filas recuperadas por la instrucción SELECT.

La variable global @@ROWCOUNT devuelve el número de filas afectadas por la última sentencia SQL ejecutada.

```
USE QhatuPERU
go
```

```
SELECT NomProveedor, Direccion, Ciudad
FROM PROVEEDOR
WHERE Departamento <> 'Lima'
go
```

```
SELECT @@ROWCOUNT
go
```

Resultados		Mensajes	
	NomProveedor	Direccion	Ciudad
1	LACTEOS DEL CENTRO	LIBERTAD 345 URB. EL PINO	HUANCAYO
2	EMBOTELLADORA LA PREFERIDA	HEROES DEL CENEP 342 CERCADO	TRUJILLO
3	DROGUERIA MAHAN	AV. CIRCUNVALACION 625 ZONA INDUSTRIAL	AREQUIPA
4	QUIMICA DEL NORTE	JOSE CARLOS MARIATEGUI 473 CERCADO	CHICLAYO
5	GOLOSINAS Y ANTOJOS	AV. CIRCUNVALACION 755 ZONA INDUSTRIAL	AREQUIPA
	(Sin nombre de columna)		
1	5		

6. INSTRUCCIONES DE CONTROL DE FLUJO

Las instrucciones de control de flujo permiten establecer el orden de ejecución de las sentencias SQL en un programa.

6.1. Bloque sentencias SQL

Sintaxis

```
BEGIN

    sentencias_SQL

END
```

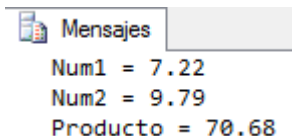
Es una estructura que contiene varias instrucciones SQL en secuencia y encerradas entre las palabras reservadas BEGIN y END.

Ejercicio 10.4: Bloque BEGIN ... END

```
BEGIN
    DECLARE @num1 decimal(7,2),
            @num2 decimal(7,2)
    DECLARE @producto decimal(12,2)

    SET @num1 = RAND()*10
    SET @num2 = RAND()*10
    SET @producto = @num1 * @num2

    PRINT CONCAT('Num1 = ', @num1)
    PRINT CONCAT('Num2 = ', @num2)
    PRINT CONCAT('Producto = ', @producto)
END
```



Mensajes

Num1 = 7.22
Num2 = 9.79
Producto = 70.68

6.2. Condicional IF ... ELSE

Sintaxis

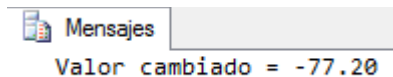
```
IF expresión_lógica
    sentencia_SQL | bloque_sentencias_SQL
[ ELSE
    sentencia_SQL | bloque_sentencias_SQL ]
```

Si **expresión_lógica** se evalúa como VERDADERA, se ejecuta la **sentencia_SQL** o **bloque_sentencias_SQL** que sigue a IF. De modo opcional se puede establecer una cláusula ELSE que establece la **sentencia_SQL** o **bloque_sentencias_SQL** a ejecutar si la **expresión_lógica** es FALSA.

Ejercicio 10.5: Condicional IF ... ELSE

```
DECLARE @num decimal(7,2)
SET @num = RAND()*100

IF @num > 50
    BEGIN
        SET @num = @num * (-1)
        PRINT CONCAT('Valor cambiado = ', @num)
    END
ELSE
    PRINT CONCAT('Valor sin cambio = ', @num)
```



6.3. Bucle WHILE

Sintaxis

```
WHILE expresión_lógica
    sentencia_SQL | bloque_sentencias_SQL
```

Ejecuta repetidamente la **sentencia_SQL** o **bloque_sentencias_SQL** mientras que la **expresión_lógica** se evalúe a VERDADERA.

Ejercicio 10.6: Bucle WHILE

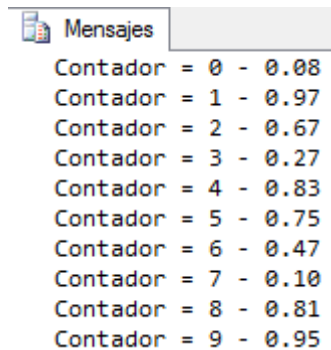
```
DECLARE @num1 int, @contador int,
        @num2 decimal(7,2)
```

```

SET @num1 = CAST(RAND()*10 AS int)
SET @contador = 0
SET @num2 = 0

WHILE @contador <= @num1
BEGIN
    SET @num2 = RAND()
    PRINT CONCAT('Contador = ', @contador,
        ' - ', @num2)
    SET @contador = @contador + 1
END

```



6.4. Instrucciones BREAK y CONTINUE

Estas instrucciones por lo general se ejecutan dentro de un condicional IF.

La instrucción BREAK hace que el control de flujo abandone el bucle WHILE y continúe después del END del WHILE.

La instrucción CONTINUE hace que el control de flujo siga ejecutando el bucle WHILE.

Ejercicio 10.7: Uso de BREAK y CONTINUE

```

DECLARE @num1 int, @contador int,
        @num2 decimal(7,2)
SET @num1 = CAST(RAND()*10 AS int)
SET @contador = 0
SET @num2 = 0

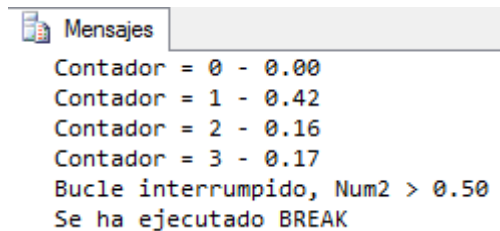
WHILE @contador <= @num1
BEGIN
    PRINT CONCAT('Contador = ', @contador,
        ' - ', @num2)

```

```

SET @contador = @contador + 1
SET @num2 = RAND()
IF @num2 > 0.5
    BEGIN
        PRINT 'Bucle interrumpido, Num2 > 0.50'
        BREAK
    END
ELSE
    CONTINUE
END
PRINT 'Se ha ejecutado BREAK'

```



6.5. Instrucción RETURN

La instrucción RETURN hace que el control de flujo abandone incondicionalmente un bloque de sentencias SQL, un procedimiento o un batch.

Ejercicio 10.8: Uso de RETURN

```

DECLARE @num1 int, @contador int,
        @num2 decimal(7,2)
SET @num1 = CAST(RAND()*10 AS int)
SET @contador = 0
SET @num2 = 0

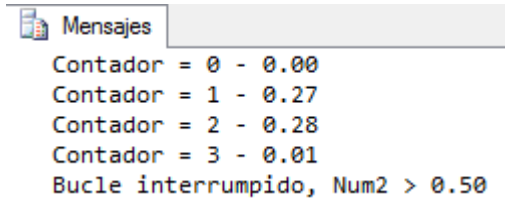
WHILE @contador <= @num1
BEGIN
    PRINT CONCAT('Contador = ', @contador,
                ' - ', @num2)
    SET @contador = @contador + 1
    SET @num2 = RAND()
    IF @num2 > 0.5
        BEGIN
            PRINT 'Bucle interrumpido, Num2 > 0.50'
            RETURN
        END
END

```

```

ELSE
    CONTINUE
END
PRINT 'Se ha ejecutado BREAK'

```



```

Mensajes
Contador = 0 - 0.00
Contador = 1 - 0.27
Contador = 2 - 0.28
Contador = 3 - 0.01
Bucle interrumpido, Num2 > 0.50

```

Observe que si se ejecuta el WHILE, ya no se ejecuta la sentencia PRINT al final.

6.6. Instrucción GOTO *etiqueta*

La instrucción GOTO *etiqueta* hace que el control de flujo se transfiera a la etiqueta especificada.

Ejercicio 10.9: Uso de GOTO

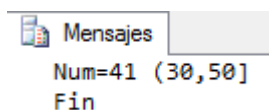
```

DECLARE @num int
SET @num = CAST(RAND()*100 AS int)

IF @num >= 0 AND @num <= 30 GOTO Marca1
IF @num > 30 AND @num <= 50 GOTO Marca2
IF @num > 50 AND @num <= 100 GOTO Marca3

Marca1:
    PRINT CONCAT('Num=', @num, ' [0,30]')
    GOTO Marca4
Marca2:
    PRINT CONCAT('Num=', @num, ' (30,50]')
    GOTO Marca4
Marca3:
    PRINT CONCAT('Num=', @num, ' (50,100]')
Marca4:
    PRINT 'Fin'

```



```

Mensajes
Num=41 (30,50]
Fin

```

7. EJERCICIOS PROPUESTOS

1. Escribir un programa que calcule el factorial de N.

$$N! = 1 \times 2 \times 3 \times 4 \times 5 \times \dots \times (N-1) \times N$$

2. Escribir un programa que encuentre el término en la posición 50 de la sucesión de Fibonacci:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \dots$$

La sucesión se inicia con los valores 0 y 1, y cada término es la suma de los dos anteriores.

3. Escribir un programa que utilizando la serie de Leibnitz

$$4 - 4/3 + 4/5 - 4/7 + 4/9 - \dots$$

calcule el valor de PI (3.14159...). Para que se acerque al valor de PI el programa deberá ejecutar por lo menos 20 millones de iteraciones. Tenga en cuenta que para que la operación de división genere un valor decimal tanto el numerador como el denominador deben ser de tipo decimal ó real.

4. Escribir un programa que usando la función RAND genere 3 números enteros de 1 dígito. Luego usando la instrucción PRINT debe imprimirlos ordenados de mayor a menor.
5. Escribir un programa que usando la función RAND genere un número entero de 3 dígitos, verifique si el número es primo, e imprima un mensaje que muestre el número indicando si es primo o no.
6. Escribir un programa que usando la función RAND genere un número entero de 3 dígitos, verifique si el número es par o impar, e imprima un mensaje que muestre el número indicando su paridad.
7. Escribir un programa que usando la función RAND genere 10 números enteros de 2 dígitos, que identifique al menor del grupo, y que lo imprima.