

TEMA 6: T-SQL PROGRAMANDO CON SQL

BERNAT COSTA

BERNAT.COSTA@CESURFORMACION.COM

¿QUÉ ES T-SQL?

T-SQL es una extensión a SQL que nos dará herramientas para incluir :

programación procedimental (condicionales,bucles..)

Variables locales.

Manejo de strings (replace...),

procesamiento de fechas (DateAdd, day(),month(),getdate()...)

Operaciones Matemáticas .

¿QUÉ PODEMOS HACER CON T-SQL?

Guiones
(Scripts)

Métodos o
funciones

Triggers

¿ES ESTO LO MISMO QUE PL/SQL?

- En muchas ofertas de trabajo se pide PL/SQL. ¿Es esto lo mismo?
- NO. PL/SQL es el language que se usa en ORACLE
- Nosotros aprenderemos a usar T-SQL, que es el "PL/SQL de Microsoft"
- Es muy parecido, aunque con algunas diferencias.

T-SQL O EL TRANSACT-SQL PERMITE:

Tipos de datos

- Los que ya teníamos, int, datetime, varchar...

Variables

- Nombre que le damos a cierto valor

Estructuras de control

- Condicionales
- Bucles

Gestión de excepciones

- Que hacer si ocurre un error

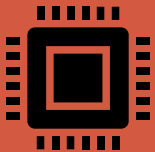
Funciones predefinidas

- Algunas nos las da SQL Server, otras las podremos "hacer" nosotr@s.

SIN EMBARGO NO PERMITE



Crear interfaces de usuario.



Crear aplicaciones ejecutables, serán aplicaciones para lanzar desde el motor de bbdd.

BLOQUES DE CÓDIGO

En T-SQL los bloques de código se traducen por:

BEGIN

...

END

Dentro podremos ponerle todas las instrucciones que queramos.

Son nuestras {} de Java.



DECLARACIÓN DE VARIABLES

- Las variables se tienen que declarar y setear.
- el nombre empieza SIEMPRE con @
- Se declaran con el comando

DECLARE @nombre as varchar(100)

- Las variables son volátiles. No se guardan en la BBDD. Se crean y destruyen al ejecutar nuestros scripts o métodos.

USAR UNA VARIABLE

Las variables se pueden setear con Select o con set.

```
DECLARE @fecha as DateTime
```

```
SET @fecha=getdate()
```

```
SELECT @fecha=getdate()
```

```
PRINT @fecha
```

SETEAR DESDE UNA CONSULTA

Podemos setear una variable desde una consulta.

```
DECLARE @idmax as int  
SET @idmax =  
(SELECT max(id)  
FROM menu)
```

```
DECLARE @idmax as int  
SELECT  
@idmax=max(id)  
FROM menu
```

¿CÓMO DEVOLVEMOS UN VALOR?

- Podemos guardarlo en una tabla con un update.
- Podemos "pintarlo" en la pantalla.

Si tenemos:

```
DECLARE @fecha as DATETIME
```

```
SET @fecha = getdate()
```

Podemos hacer:

```
PRINT @fecha  
Para devolver  
en formato texto
```

```
SELECT @fecha  
Para devolver  
una tabla
```

ESTRUCTURAS DE CONTROL



CONDICIONALES



BUCLES

CONDICIONALES

CASE
WHEN

Se puede poner en
medio de
una consulta

IF ELSE

Para definir dos
bloques
diferenciados con
BEGIN END

¿QUÉ ES UN CONDICIONAL?



Es una estructura de control que nos servirá para decidir entre 2 o más opciones según una condición.



Si se cumple esto, haz esto, sino, haz esto otro.



En programación también tenemos el SWITCH. Aquí no lo vamos a usar. Es prescindible.

CASE WHEN THEN ELSE END

```
CREATE TABLE Personas (nombre varchar(100),apellido varchar(100),edad int))  
INSERT INTO Personas SELECT 'Pedro','Rodriguez',41  
  
INSERT INTO Personas SELECT 'Adam','Rodriguez',10  
INSERT INTO Personas SELECT 'Pau','Rodriguez',7  
  
INSERT INTO Personas SELECT 'Laia','Rodriguez',4
```

SELECT nombre, apellido,

CASE WHEN edad>18 **THEN 'SI' ELSE 'NO' END** as mayorEdad

FROM Personas

Si dentro de la condición necesitamos escribir más de una línea,
lo encapsulamos con BEGIN END

IF ELSE

```
DECLARE @num as INT  
SET @num=5  
BEGIN
```

```
IF @num>4  
    PRINT 'es mayor que 4'  
ELSE  
    PRINT 'NO es mayor que 4'
```

```
DECLARE @num as INT  
SET @num=5  
BEGIN
```

```
IF @num>4  
    BEGIN  
        DECLARE @num2 as INT  
        SET @num2=@num  
        PRINT 'es mayor que 4'  
    END  
ELSE  
    BEGIN  
        Print 'NO es mayor que 4'  
    END
```


BUCLES

- Un bucle es un fragmento de código que se repite hasta que nosotros decidamos.
- En programación, hay muchos tipos de bucles
 - While, for, do while...
 - Nosotros solo vamos a ver el WHILE. Todo programa que se puede hacer con un bucle se puede hacer con un while.

WHILE

- El WHILE consta de dos partes:
 - Condición de salida
 - Lineas que se van a repetir hasta que se cumpla la condición de salida.
 - Estas lineas, deben contener una modificación de la variable que se comprueba para salir.

WHILE condicion

BEGIN

....

END

EJEMPLO: SCRIPT PARA CALCULAR EL FACTORIAL DE 6

```
DECLARE @resultado as int =1
```

```
DECLARE @num as INT
```

```
SET @num=6
```

```
BEGIN
```

```
WHILE @num<>0 --condición de salida
```

```
BEGIN
```

```
    SET @resultado = @resultado * @num
```

```
    SET @num = @num-1 -- linea para modificar la condición
```

```
END
```

```
PRINT @resultado
```

```
END
```

SQLCMD, SQL SERVER DESDE LA LINEA DE COMANDOS

- **sqlcmd** -S localhost -U alumnadobbdd -P 123456Ab\$ -i fichero.sql

```
docker exec -it sql-server-db /opt/mssql-tools/bin/sqlcmd -S localhost -U  
sa -P 12345Ab## -Q "USE AREPAZO; SELECT nombre,precioventa from menu "
```

- Nos servirá para lanzar scripts desde la consola de windows o linux contra SQL Server.
 - -S -> server name
 - -U -> user name
 - -P -> password
 - -i -> ruta fichero script.sql

FUNCIONES CON VARCHARS

Replace

Left y right

- le pasamos una cadena y un numero, nos devolverá la cadena recortada a ese numero de caracteres.

Trim, ltrim,rtrim...

- quita espacios en blanco

Len

- nos dará la longitud...

Concat

- concatena dos cadenas

Substring

- nos da una parte de un string indicandole la posición y el numero de caracteres

EJEMPLOS

- `REPLACE('hola mundo','hola','adios') = 'adios mundo'`
- `LEFT('hola mundo',4) = 'hola'`
- `RIGHT('hola mundo',5) = 'mundo'`
- `TRIM(' hola mundo ') = 'hola mundo'`
- `LEN('hola mundo') = 10`
- `CONCAT('hola',' mundo') = 'hola mundo' = 'hola ' + 'mundo'`
- `SUBSTRING ('hola mundo',2,1) = 'o'`

CONVERTIR TIPOS DE DATOS

- Cuando tenemos un INT y lo queremos "sumar" a un VARCHAR, tenemos que decirle antes, que nos convierta el INT a VARCHAR
- Hacemos un CAST de la variable. Le cambiamos el tipo.

```
Declare @num as int
```

```
SET @num = 5
```

```
DECLARE @cadena as varchar(10)
```

```
SET @cadena = cast(@num as varchar(10))
```

SUMA DE FECHAS

- Las fechas no se pueden sumar con el operador +
- Se suman con la función DATEADD que recibe tres parametros.
 - 1º Que le vamos a sumar
 - 2º Cuanto le vamos a sumar
 - 3ª a que le vamos a sumar
- DATEADD(year,2,getdate()) = sumará 2 años a la fecha actual.
- DATEADD(day,-15,getdate())= restará 15 días a la fecha actual.
- + info en <https://docs.microsoft.com/en-us/sql/t-sql/functions/dateadd-transact-sql?view=sql-server-ver15>

¿QUE SABEMOS HACER?

- Sabemos crear y usar variables
- Sabemos usar un condicional (case when o if else)
- Sabemos usar un bucle(WHILE)
- Sabemos mezclar todo esto con sentencias SQL.
- Guardar un script y ejecutarlo cuando queramos desde la consola o desde un cliente SQL.

¿QUE MÁS PODEMOS HACER?

- CREAR MÉTODOS.
- Un método es un conjunto de instrucciones al que le ponemos un nombre y se queda guardado para poderlo ejecutar más adelante.
- Además, podremos pasarle 1 o varios parámetros de entrada y podrá tener también parametros de salida.
- Son "cajas negras" para hacer algún cálculo, mostrar una tabla o modificar una bbdd.

2 FORMAS DE ENCAPSULAR CÓDIGO

- Procedimientos
- Funciones

PROCEDIMIENTOS

- Estos podrán tener parametros de entrada y salida.
- Serán muy parecido a lo que estamos haciendo, pero le daremos un nombre y lo guardaremos en nuestra BBDD.

```
CREATE PROCEDURE SelectAllCustomers  
AS  
  BEGIN  
    SELECT * FROM Customers  
  END
```

COMO SE DISPARAN LOS PROCEDURES?

EXEC nombredelprocedimiento

EXEC SelectAllCustomers

A decorative graphic on the left side of the slide, consisting of a network of light blue lines and small circles, resembling a circuit board or a neural network, extending from the top to the bottom of the frame.

EJEMPLO DE UN HELLO WORLD

```
CREATE PROCEDURE HELLOWORLD  
AS  
    BEGIN  
        PRINT 'HOLA MUNDO!';  
    END
```

PROCEDIMIENTOS CON PARAMETROS DE ENTRADA

```
CREATE PROCEDURE Dice_Palabra @palabra CHAR(30)
```

```
AS
```

```
BEGIN
```

```
    PRINT @palabra;
```

```
END
```

```
EXEC Dice_Palabra 'hola clase de cesur';
```

```
EXEC Dice_Palabra 'adios clase de cesur';
```

¿Y CON VARIOS PARAMETROS DE ENTRADA?

USE arepazo

```
CREATE PROCEDURE or ALTER vermenu @tipoalimento INT,@categoria INT
```

```
AS
```

```
BEGIN
```

```
IF @categoria is null
```

```
    SELECT * FROM menu WHERE tipo=@tipoalimento
```

```
ELSE
```

```
    SELECT * FROM menu WHERE tipo=@tipoalimento AND categoria = @categoria;
```

```
END
```

```
EXEC vermenu 1, 2 --> comida ,platos
```

```
EXEC vermenu 2,null --> postres
```


PARÁMETROS OPCIONALES

- Se definen dándole un valor por defecto al parametro

CREATE PROCEDURE

vermenu @tipoalimento INT, @categoria INT = NULL

(Si el valor por defecto está en un tercer parametro en medio, deberemos usar valor por defecto, RECOMENDACIÓN: el valor opcional, siempre el último.)

```
CREATE or ALTER PROCEDURE VerUsuariosPoblacion @ISOEstado  
CHAR(2),@pob CHAR(30)='Madrid',@pro CHAR(30)
```

```
AS
```

```
BEGIN
```

```
SELECT * FROM usuarios WHERE poblacion=@pob  
AND provincia = @pro and iso=@ISOEstado;
```

```
END
```

```
EXEC VerUsuariosPoblacion 'ES,@pro='Madrid'
```

```
-->falla
```

```
EXEC VerUsuariosPoblacion 'ES,DEFAULT,@pro='Madrid'
```

```
-->Funciona!
```

PARÁMETROS DE SALIDA

- Podemos definir procedimientos con 1 o varios parámetros de salida.

```
CREATE or ALTER PROCEDURE ultimo_contrato @ofi INT,
```

```
    @fecha DATETIME OUTPUT
```

```
AS
```

```
BEGIN
```

```
    SELECT @fecha=(SELECT MAX(fecha_contrato)
```

```
        FROM empleados WHERE oficina=@ofi)
```

```
END
```

¿Y COMO SE USA?

```
DECLARE @ULTIMA AS DATETIME;
```

```
EXEC ULTIMO_CONTRATO 12,@ULTIMA OUTPUT;
```

```
PRINT @ULTIMA;
```

MODIFICAR Y DESTRUIR PROCEDIMIENTOS

ALTER PROCEDURE
nombre....

DROP
PROCEDURE nombre

(Igual que una tabla)



That's all Folks!