SQL Programming

Módulo 5 - Variables

Variables Locales y Globales

DECLARE

Las variables se declaran en el cuerpo de un proceso por lotes o un procedimiento con la instrucción DECLARE, y se les asignan valores con una instrucción SET o SELECT.

Las variables de cursor pueden declararse con esta instrucción y utilizarse con otras instrucciones relacionadas con los cursores.

Luego de la declaración, todas las variables se inicializan como NULL, a menos que se proporcione un valor como parte de la declaración.

Nota
Los nombres de algunas funciones del sistema Transact-SQL comienzan por dos arrobas (@@). A pesar de que en versiones anteriores de SQL Server se hacía referencia a @@functions como variables globales, no son variables y no tienen el mismo comportamiento que las variables. Las funciones @@functions son funciones del sistema y el uso de su sintaxis sigue las reglas de las funciones.

Declarar una Variable Escalar

La instrucción DECLARE inicializa una variable de Transact-SQL al:

Asignar un nombre. El nombre debe tener un único @ como primer carácter.

Asignar un tipo de datos suministrado por el sistema o definido por el usuario y una longitud. Para las variables numéricas, se asignan también una precisión y una escala.

Por ejemplo, la siguiente instrucción DECLARE crea una variable local llamada @mycounter con un tipo de datos int.

Sintaxis

```
DECLARE @Contador int;
```

Se puede declarar varias variables en una misma sentencia.

Sintaxis

```
DECLARE @Contador int, @Edad int;
```

Inicializar una variable

Declarar y Asignar un valor.

El siguiente ejemplo declara una variable @Contador y la inicializa en cero.

Sintaxis

```
DECLARE @Contador int=0;
```

Asignar un valor luego de la declaración de una variable.

El siguiente ejemplo declara una variable @Mensaje y la inicializa utilizando set.

Sintaxis

```
DECLARE @Mensaje varchar(50);

SET @Mensaje='Hola Mundo';
```

Asignar un valor luego de la declaración de una variable.

El siguiente ejemplo declara una variable @Mensaje y la inicializa utilizando select.

Sintaxis

```
DECLARE @Mensaje varchar(50);

SELECT @Mensaje='Hola Mundo';
```

Declarar una Variable de tipo Tabla

El ejemplo siguiente crea una variable de tipo tabla.

Sintaxis

```
DECLARE @VariableTabla table(
ID int NOT NULL,
Apellido varchar(50),
Nombre varchar(50)
);
```

Declarar una Variable de tipo Tabla

Ventajas que encontraremos al usar variables de tipo tabla:

- Tienen un ámbito bien definido. El procedimiento almacenado, la función o el batch en el que se declaran.
- Las variables de tipo tabla producen menos recopilaciones de los procedimientos almacenados en los que se encuentran que si utilizamos tablas temporales.
- Las variables de tabla no necesitan de bloqueos ni de tantos recursos como las tablas temporales.

Limitaciones que encontraremos al usar variables de tipo tabla:

- No podemos cambiar la definición de la tabla una vez declarada.
- No podemos utilizar índices que no sean agrupados.
- No se pueden utilizar en SELECT INTO.
- No podemos utilizar funciones en las restricciones.

Declarar una Variable de tipo Tabla

Si ponemos en una balanza las ventajas y los inconvenientes vemos que en general es mejor utilizar las variables de tipo tabla que las tablas temporales. Solo en el caso de tener gran cantidad de datos en una tabla temporal y si la vamos a usar varias veces es preferible la opción de tablas temporales porque en ellas podemos definir índices.

Variables Globales de Sistema

Como habíamos visto las variables SQL se declaran con la cláusula DECLARE y el nombre de la variable debe preceder el símbolo @, SQL proporciona variables del sistema predefinidas Los nombres de estas variables comienzan con el doble símbolo @@. No pueden asignarse valores. Solamente consultar.

La siguiente variable de sistema devuelve el identificador de la conexión actual.

```
@@rowcount

@@identity

@@servername

@@spid
```

Control de Flujo: Bloque BEGIN – END

Encierra un conjunto de instrucciones Transact-SQL de forma que se pueda ejecutar un grupo de instrucciones Transact-SQL. BEGIN y END son palabras clave del lenguaje de control de flujo.

Se utilizan para controles de flujos como IF, ELSE, WHILE.

Control de Flujo: Bloque IF … ELSE

Impone condiciones en la ejecución de una instrucción Transact-SQL.

La instrucción Transact-SQL (sql_statement) que sigue a Boolean_expression se ejecuta si Boolean_expression se evalúa como TRUE.

La palabra clave opcional ELSE es una instrucción Transact-SQL alternativa que se ejecuta cuando Boolean_expression se evalúa como FALSE o NULL.

El siguiente ejemplo declara una variable y la inicializa, luego realiza una evaluación.

Sintaxis

```
DECLARE @valor int

SET @valor=55

IF @valor<10
      PRINT 'EL PRECIO ES MUY BAJO'
ELSE
      PRINT 'EL PRECIO ES MUY CARO'
```

Control de Flujo: Bloque IF ... ELSE

Construcción WHILE, BREAK y CONTINUE

Establece una condición para la ejecución repetida de una instrucción o bloque de instrucciones SQL.

Las instrucciones se ejecutan repetidamente siempre que la condición especificada sea verdadera. Se puede controlar la ejecución de instrucciones en el bucle WHILE con las palabras clave BREAK y CONTINUE.

BREAK: Produce la salida del bucle WHILE más interno. Se ejecutan las instrucciones que aparecen después de la palabra clave END, que marca el final del bucle.

CONTINUE: Hace que se reinicie el bucle WHILE y omite las instrucciones que haya después de la palabra clave CONTINUE.

Control de Flujo: Bloque IF ... ELSE

GOTO

La instrucción GOTO provoca que, en la ejecución de un lote de Transact-SQL, se salte a una etiqueta.

Ninguna de las instrucciones situadas entre la instrucción GOTO y la etiqueta se ejecutarán. El nombre de la etiqueta se define con la sintaxis:

La etiqueta que constituye el objetivo de un GOTO sólo identifica el destino del salto.

WAITFOR

La instrucción WAITFOR suspende la ejecución de un lote, un procedimiento almacenado o una transacción hasta que:

- Haya pasado un intervalo de tiempo especificado.
- Se haya alcanzado una hora del día especificada.

Control de Flujo: Bloque IF ... ELSE

Sintaxis

```
DECLARE @indice INT;
SET  @indice=0 ;

WHILE (@indice<=10)
BEGIN
            IF @indice=7
            BEGIN
                  SET @indice=@indice+1;

                  WAITFOR DELAY '00:00:20'

                  CONTINUE;
            END

            -- Dos formas de salir forzando el while
            IF @indice=8 GOTO mensaje;
            IF @indice=9 BREAK;
            --


            SET @indice=@indice+1;
END

mensaje:
            PRINT 'SALI DEL WHILE POR EL GOTO'
```

¡Muchas gracias!

¡Sigamos trabajando!