DuocUC

SQL SERVER APLICADO (SSA010)

Ariel Alexis Fierro Sáez
afierrosaez@gmail.com

Introducción Transact-SQL

- Transact SQL es el lenguaje de programación que proporciona Microsoft SQL Server para extender el SQL estándar con los elementos característicos de los lenguajes de programación
  - Variables
  - Estructuras de control de flujo
  - Gestión de Excepciones
  - Bucles

Introducción Transact-SQL

- Script (Bloque anónimo)
- Procedimiento Almacenado (Stored procedure)
- Funciones (Function)
- Disparadores (Trigger)

Tipo de Datos

- **Numéricos**

  - **Bit.** Una columna o variable de tipo bit puede almacenar el rango de valores de 1 a 0.
  - **Tinyint**. Una columna o variable de tipo tinyint puede almacenar el rango de valores de 0 a 255.
  - **SmallInt**. Una columna o variable de tipo smallint puede almacenar el rango de valores -32768 a 32767.
  - **Int**. Una columna o variable de tipo int puede almacenar el rango de valores -2^{31} a $2^{31}-1$.
  - **BigInt**. Una columna o variable de tipo bigint puede almacenar el rango de valores -2^{63} a $2^{63}-1$.
  - **Decimal(p,s).** Una columna de tipo decimal puede almacenar datos numéricos decimales sin redondear. Donde p es la precisión (número total del dígitos) y s la escala (número de valores decimales)
  - **Float**. Una columna de datos float puede almacenar el rango de valores $-1{,}79x\text{-}10^{308}$ a $1{,}79x\text{-}10^{308,}$, si la definimos con el valor máximo de precisión. La precisión puede variar entre 1 y 53.
  - **Real**. Sinónimo de float(24). Puede almacenar el rango de valores $-3{,}4x\text{-}10^{38}$ a $3{,}4x\text{-}10^{38,}$
  - **Money**. Almacena valores numéricos monetarios de -2^{63} a $2^{63}-1$, con una precisión de hasta diexz milesimas de la unidad monetaria.
  - **SmallMoney**. Almacena valores numéricos monetarios de -214.748,3647 a 214.748,3647, con una precisión de hasta diez milésimas de la unidad monetaria.
  - **Numeric(p, s).** Es funcionalmente equivalente de Decimal.

Tipo de Datos

- **Caracteres**

  - **Char(n).** Almacena n caracteres en formato ASCII, un byte por cada letra. Cuando almacenamos datos en el tipo char, siempre se utilizan los n caracteres indicados, incluso si la entrada de datos es inferior. Por ejemplo, si en un char(5), guardamos el valor 'A', se almacena 'A   ', ocupando los cinco bytes.
  - **Varchar(n).**Almacena n caracteres en formato ASCII, un byte por cada letra. Cuando almacenamos datos en el tipo varchar, únicamente se utilizan los caracteres necesarios, por ejemplo, si en un varchar(255), guardamos el valor 'A', se almacena 'A', ocupando solo un byte bytes.
  - **Varchar(max).** Igual que varchar, pero al declararse como max puede almacenar 2^{31}-1 bytes.
  - **Nchar(n).**Almacena n caracteres en formato UNICODE, dos bytes por cada letra. Es recomendable utilizar este tipo de datos cuando los valores que vayamos a almacenar puedan pertenecer a diferente idiomas.
  - **Nvarchar(n).**Almacena n caracteres en formato UNICODE, dos bytes por cada letra. Es recomendable utilizar este tipo de datos cuando los valores que vayamos a almacenar puedan pertenecer a diferente idiomas.
  - **Nvarchar(max).**Igual que varchar, pero al declararse como max puede almacenar 2^{31}-1 bytes.

Tipo de Datos

- **Fecha y tiempo**

  - **Datetime.** Almacena fechas con una precisión de milisegundo. Debe usarse para fechas muy especificas.

  - **SmallDatetime.** Almacena fechas con una precisión de minuto, por lo que ocupa la mitad de espacio de que el tipo datetime, para tablas que puedan llegar a tener muchos datos es un factor a tener muy en cuenta.

  - **TimeStamp.** Se utiliza para marcar un registro con la fecha de inserción - actualización. El tipo timestamp se actualiza automáticamente cada vez que insertamos o modificamos los datos.

Variables

- Una variable es un valor identificado por un nombre sobre el cual podemos realizar modificaciones.

- En transactSQL los identificadores de las variables deben comenzar con el carácter @, precedida de la sentencia DECLARE

Variables

- Declaración de variables

```
DECLARE
@nombre varchar(20),
@apellido varchar(20)
```

- Asignar valor a una variable
  - Instrucción SET
  - Sentencia SELECT
  - Sentencia FETCH de un cursor

Variables

- Asignar valor usando SET

```
set @nombre = (select nombre from cliente where rut='9999999-9')
set @apellido= 'Marin'
```

- Asignar valor usando SELECT

```
select @nombre=nombre, @apellido=apellido from cliente where
rut='9999999-9'
```

Operadores

Operador de asignación	=
Operadores aritméticos	+ (suma) - (resta) * (multiplicación) / (división) ** (exponente) % (modulo)
Operadores de comparación	= (igual a) <> (distinto de) < (menor que) > (mayor que) >= (mayor o igual a) <= (menor o igual a)
Operadores lógicos	AND (y lógico) NOT (negacion) OR (o lógico)
Operador de concatenación	+
Otros	ALL (Devuelve TRUE si el conjunto completo de comparaciones es TRUE) ANY(Devuelve TRUE si cualquier elemento del conjunto de comparaciones es TRUE) BETWEEN (Devuelve TRUE si el operando está dentro del intervalo) EXISTS (TRUE si una subconsulta contiene filas) IN (TRUE si el operando está en la lista) LIKE (TRUE si el operando coincide con un patron) NOT (Invierte el valor de cualquier operador booleano) SOME(Devuelve TRUE si alguna de las comparaciones de un conjunto es TRUE)

Estructuras de control de flujo

- BEGIN...END

Encierra un conjunto de instrucciones Transact-SQL de forma que se pueda ejecutar un grupo

```
BEGIN
    sql_statement
    ...
END
```

Estructuras de control de flujo

- Estructura IF...ELSE

```
IF boolean_expresion
    BEGIN
    sql_statement
    ...
    END
ELSE
    BEGIN
    sql_statement
    ...
    END
```

```
IF boolean_expresion
    BEGIN
    sql_statement
    ...
    END
ELSE IF
    BEGIN
    sql_statement
    ...
    END
ELSE
    BEGIN
    sql_statement
    ...
    END
```

- Estructura CASE

```
--Simple CASE expression:
CASE input_expression
    WHEN when_expression THEN result_expression
    WHEN when_expression THEN result_expression
    .....
    [ ELSE else_result_expression ]
    END
```

```
--Searched CASE expression:
CASE
    WHEN Boolean_expression THEN result_expression
    WHEN Boolean_expression THEN result_expression
    ....
    [ ELSE else_result_expression ]
    END
```

Ejemplo Script (o bloque anónimo)

```
DECLARE

@nombre nvarchar(20),
@apellido nvarchar(20),
@edad int


IF @edad <18
    BEGIN
    print 'Cliente debe ser mayor de edad'
    END
ELSE
    BEGIN
    INSERT INTO cliente VALUES(@nombre, @apellido, @edad);
    END
```

Ejercicios

- Determine la cantidad de peso y volumen que lleva cada camión, indicando si cumple con los restricción de 25 toneladas o 35 m3 como máximo en cada camión.

- Crear un script que permita determinar entre dos camiones si es posible trasladar los cartones siempre y cuando cumplan con los requisitos anteriores

- Determinar el porcentaje de camiones que no cumplen con el requisito descrito.

Ejercicios

- Crear una columna que describa el estado del cartón según la siguiente tabla

Estado	Descripción
00	Creado
10	Programado
40	Cargado
45	Pendiente
50	Camino
99	Sin Stock

- Crear un script que permita determina el porcentaje de carga de los cartones para un determinado camión.