**Cuál es la diferencia entre el Lenguaje SQL y el Lenguaje Transact SQL**

-SQL es un lenguaje de consultas para los sistemas de bases de datos relacionales.

-TRANSACT SQL es el lenguaje de programación que proporciona al SQL los elementos característicos de un

lenguaje de programación.

**Que se puede programar con Transact SQL**

Se puede programar las unidades de programa dentro del sistema de gestión de base de datos como ser:

-Scripts.

-Funciones o procedimientos almacenados.

-Triggers.

**Liste los tipos de datos que maneja Transact SQL**

Tipos de datos numéricos: Bit, Smallint, Int, decimal, float y real.

Tipos de datos de carácter: char, varchar.

Tipos de datos de fecha: Datetime, SmallDatetime.

Tipos de datos binarios: Binary, Varbinary.

Tipos de datos XML: XML.

**Como se declara una variable en Transact SQL, cite ejemplos**

El nombre de una variable debe comenzar por el carácter **@** y se utiliza la palabra clave **DECLARE** seguido

Del nombre y tipo de dato de la variable.

EJ) **DECLARE** **@**nombre char(40)

EJ) **DECLARE @**código int;

**Como se asigna valor a una variable en Transact SQL, cite ejemplos**

Se asigna valor con la palabra **SET**.

EJ) **SET** @nombre = ‘Base de datos II’

EJ) **SET** @código = 3;

**Cómo asignar valores obtenidas de una consulta a una variable en Transact SQL, cite ejemplos**

Se puede utilizar 2 instrucciones: Set y Select.

Utilizando la instrucción SET tenemos:

**SET** @nombre = (**SELECT** nomb **FROM** prov **WHERE** cprv=1)

**SET** @codigo = (**SELECT** calma **FROM** alma **WHERE** calma=1)

Utilizando la instrucción SELECT tenemos:

**SELECT** @nombre=noma, @codigo=calma **FROM** alma **WHERE** calma=1, cuando se ocupa **SELECT** no olvidar la cláusula WHERE.

**Que es un CURSOR en Transact SQL**

Es una variable que nos permite recorrer con un conjunto de resultados obtenido a través de una sentencia **SELECT** fila a fila.

**Para que sirve el uso de CURSOR en Transact SQL**

Se utiliza para leer una lista de datos y hacer cálculos con dichas listas.

**Como se declara un CURSOR en Transact SQL**

Se utiliza la clausula **DECLARE** seguida del nombre de cursor y luego se agrega la palabra **CURSOR.**

EJ) **DECLARE** Cdatos **CURSOR**

**Como se abre un CURSOR en Transact SQL**

Se utiliza la clausula **OPEN** seguido del nombre de cursor

EJ) **OPEN** Cdatos

**Para que sirve la instrucción FETCH en Transact SQL**

Sirve para leer los datos del cursor buscando la primera fila y luego se pasa los valores a una variable con la cláusula **into**.

**Para que sirve la variable @@FETCH\_STATUS, que valores devuelve**

Sirve para recorrer cada una de las filas y retorna el valor de cero si hay valores y sino retorna diferente de cero.

**Como se cierra un CURSOR en Transact SQL**

Utilizando la instrucción **CLOSE** seguida del nombre de cursor.

EJ) **CLOSE** Cdatos

**Cite un ejemplo de cómo leer todas las filas de una tabla en Transact SQL**

Utilizando la sentencia **WHILE**:

**WHILE** (**@@FETCH\_STATUS** = 0)

BEGIN

END

**Liste los principales operadores utilizados en Transact SQL**

**Operador de asignación**: =

**Operadores aritméticos**: suma’+’, resta‘-‘, multiplicación’\*’, división’/’, exponente’\*\*’, modulo’%’

**Operadores relacionales o de comparación**: igual a ‘=’, Mayor que ‘>’, Menor que ‘<’, Mayor o igual que ‘>=’,

Menor o igual que ‘<=’, Distinto de ‘<>’, Distinto de ‘!=’, No mayor a ‘!>’, No menor a ‘!<’

**Operador de concatenación** ‘+’

**Operadores Lógicos**: And, Not, Or, (&, |,) a nivel de bit.

**Otros Operadores**: All, Any, Between, Exists, In, Like, Not, Some

**Como funciona la estructura de control IF THEN, cite ejemplo**

Permite evaluar una expresión booleana dando como resultado verdadero o falso y ejecuta las operaciones dentro del **BEGIN END**.

EJ) **IF** (**SELECT** count(\*) **FROM** sumi **WHERE** cprv=1)>1

**PRINT** ’El proveedor 1 ha suministrado producto’

**ELSE**

**PRINT** ‘El proveedor 1 no ha suministrado producto’

**Como funciona la estructura de control CASE, cite ejemplo**

Permite evaluar una expresión y devolver un valor u otro.

EJ) Programa para clasificar a un proveedor en base al importe de todos sus productos suministrados.

**DECLARE** @clase char(40), @impt float,

**SELECT** @impt=isnull(sum(impt),0) **FROM** sumi **WHERE** cprv=3

**SET** @clase = (**CASE**

**WHEN** (@impt>0 and @impt<=50) **THEN** ‘Proveedor minorista’

**WHEN** (@impt>51 and @impt<=200) **THEN** ‘Proveedor intermedio’

**WHEN** (@impt>200) **THEN** ‘Proveedor mayorista’

ELSE ‘Proveedor sin clasificar’

**END**)

**PRINT** @clase + ‘Importe’ +**cast**(@impt as char(10))

**Como funciona la estructura de control WHILE, cite ejemplo**

El bucle **WHILE** se repite mientras la expresión sea verdadera, es el único tipo de bucle que dispone transact sql.

EJ) Listar los productos suministrados por el proveedor 3, en la lista se debe mostrar: el nombre del producto, color,

Cantidad y fecha de suministro.

**DECLARE** @nombre char(40), @color varchar(15), @cant float, @ftra date

**DECLARE** Cdatos **CURSOR**

**FOR SELECT** nomp, colo, cant, ftra **FROM** sumi, prod **WHERE** sumi.cprd=prod.cprd

**OPEN** Cdatos

**FETCH** Cdatos **INTO** @nombre, @color, @cant, @ftra

**WHILE** (@@FETCH\_STATUS = 0)

**BEGIN**

**PRINT** @nombre + @color + **cast**(@cant as char(5)) + **cast**(@ftra as char(12))

**FETCH** Cdatos **INTO** @nombre, @color, @cant, @ftra

**END**

**CLOSE** Cdatos

**DEALLOCATE** Cdatos

**Como se hace el control de errores en Transact SQL, cite ejemplo**

El control de errores se hace con el TRY CATCH.

Ej) Programa que detecta una división entre cero.

**BEGIN TRY**

**DECLARE** @divisor int, @dividendo int, @resultado int,

**SET** @dividendo=100

**SET** @divisor=0

**SET** @resultado = @dividendo / @divisor

**PRINT** ‘No hay error’

**END TRY**

**BEGIN CATCH**

**PRINT** ’Se ha producido un error’

**END CATCH**

**Para que sirve la variable @@ERROR y que valores devuelve**

Es una variable global de sistema que almacena el numero de error producido por la ultima sentencia transact sql ejecutada y retorna el valor de cero sino hay error caso contrario retorna valor distinto de cero.

EJ) **DECLARE** @divisor int, @dividendo int, @resultado int,

**SET** @dividendo=100

**SET** @divisor=0

**SET** @resultado = @dividendo / @divisor

**IF** @@ERROR = 0

**PRINT** ‘No hay error’

**ELSE**

**PRINT** ‘Se ha producido un error’