UNIVERSIDAD AUTÓNOMA GABRIEL RENÉ MORENO

FACULTAD DE INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN Y TELECOMUNICACIONES

Práctico #5

Apellidos y nombres: Soliz Supayabe Jose Daniel

Nro. Registro: 218075881

MATERIA: BASE DE DATOS 2 - SD

FECHA: 24/05/2021

SANTA CRUZ – BOLIVIA

Desarrolle las siguientes actividades:

1. **Cual es la diferencia entre el Lenguaje SQL y el Lenguaje Transact SQL**

   SQL.- Es un lenguaje de consultas para los sistemas de base de datos relacionales, no poseen la potencia de los lenguaje de programación.

   Transact SQL.- Es el lenguaje de programación que proporciona al SQL los elementos característicos de los lenguajes de programación :variables, sentencias de control, bucles

2. **Que se puede programar con Transact SQL**

   Se puede programar las unidades de programas dentro del SGBD, como ser

   - Scripts
   - Funciones
   - Procedimientos almacenados
   - Triggers

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

   Tipos Numericos
   BIT

   SmallInt
   Int
   Decimal(p,s)
   Float
   Real

   Tipos de Datos de Carácter
   Char(n)
   Varchar(n)

   Tipos de datos de fecha
   Datetime
   SmallDatetime

   Tipos de datos Binarios
   Binary
   Varbinary

   Tipos de datos XML
   XML

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

   Para declarar variables en Transact SQL utilizamos la palabra clave **declare** seguido del **@+identificador** y **tipo de dato** de la variable

```
declare @nombre char(40)

declare @numero int, @ubicacion char(40)
```

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

   Se puede asigar de distintas formas:

   - A través de la instrucción **SET**

```
SET @nombre='JUAN PABLO'
SET @nombre=(SELECT nomb from prov where cprv=1)
```

   - Utilizando la sentencia **select**

```
declare @codigo int, @ciud char(2)
SELECT  @codigo = calm,
        @ciudad = ciud from alma where calm=1
```

     Ojo.-teniendo encuenta que si en la consulta devuelve mas de un registro, la variables quedaran asignadas con los valores de la ultima fila

   - REALIZANDO un FETCH de un cursor

```
DECLARE @nombre char(40),
           @color varchar(15)
DECLARE CDATOS CURSOR
FOR SELECT nomp, colo FROM prod
OPEN CDATOS
FETCH CDATOS INTO @nombre, @color    -- Lectura de la primera
fila del cursor
WHILE (@@FETCH_STATUS = 0)
  BEGIN
      PRINT @nombre + @color
      FETCH CDATOS INTO @nombre, @color
  END
CLOSE CDATOS
DEALLOCATE CDATOS
```

6. **Comos asignar valores obtenidas de una consulta a una variable en Transact SQL, cite ejemplos**

   Utilizando la sentencia **select**

```
 SELECT @codigo= calm, @nombre=noma, @ciudad= ciud from
alma where calm=1
```

7. **Que es un CURSOR en Transact SQL**

   Es la herramienta mas importante dentro de Transact SQL, La cual es una variable que nos permite reocorrer con un conjunto de resultados obtenido atraves de una sentencia SELECT fila a fila

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

   Sirve para leer una lista de datos y hacer ciertos cálculos sobre la lista obtenida

9. **Como se declare un CURSOR en Transact SQL**

```
--DECLARACION DEL CURSOR
DECLARE <nombre_cursor> CURSOR
FOR <sentencia_SQL>
```

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

```
OPEN <nombre_cursor>
```

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

    LEE una fila determina de un cursor

12. **Para que sirve la variable @@FETC_STATUS, que valores devuelve**

    Si al leer el **fetch** encontró una fila devolverá cero = 0, caso contrario devolverá un valor distinto de cero <>0

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

```
CLOSE <nombre_cursor>
```

14. **Cite un ejemplo de como leer todas las filas de una tabla en Transact SQL**

```
DECLARE @nombre char(40),
             @color varchar(15)
```

```
DECLARE CDATOS CURSOR
FOR SELECT nomp, colo FROM prod
OPEN CDATOS
FETCH CDATOS INTO @nombre, @color
WHILE (@@FETCH_STATUS = 0)
  BEGIN
       PRINT @nombre + @color
       FETCH CDATOS INTO @nombre, @color
  END
CLOSE CDATOS
DEALLOCATE CDATOS
```

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

Operadores en Transact SQL

La siguiente tabla ilustra los operadores de **Transact SQL** .

Tipo de operador	Operadores
Operador de asignación	=
Operadores aritméticos	+ (suma) - (resta) * (multiplicación) / (división) ** (exponente) % (modulo)
Operadores relacionales o de comparación	= (igual a) <> (distinto de) != (distinto de) < (menor que) > (mayor que) >= (mayor o igual a) <= (menor o igual a) !> (no mayor a) !< (no menor a)
Operadores lógicos	AND (y lógico) NOT (negacion) OR (o lógico) & (AND a nivel de bit) | (OR a nivel de bit) ^ (OR exclusivo a nivel de bit)
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)

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

Permite expresar una expresión booleana(resultado VERDADERO-FALSO), y ejecuta las instrucciones contenidas dentro del formado BEGIN END.

Sintaxis:

```
if(<expresion>)
  begin

  end
ELSE begin

End
```

EJEMPLO: **Verificar si el provedor con código 1 ha suministrado algún producto**

```
IF(select count(*) from sumi where cprv=1) > 0
      PRINT 'El provedor 1 a sumistrado producto'
ELSE
      PRINT 'El provedor 1 NO a sumistrado producto'
```

17. Como funciona la estructura de control CASE, cite ejemplo

Permite evaluar una expresión y devolver un valor u otro, SINTAXIS :

```
CASE <expresion>
  WHEN <valor_expresion> THEN <valor devuelto>
  WHEN <valor_expresion> THEN <valor devuelto>
  WHEN <valor_expresion> THEN <valor devuelto>
  ELSE <valor devuelto>
END
```

EJEMPLO: Programa para clasificar a un provedor en base al importe de todos sus productos suministrado

```
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
Minoritas'
              WHEN (@impt>51 and @impt<=200) THEN 'Proveedor
Intermedio'
              WHEN (@impt>200) THEN 'Proveedor Mayoritas'
              ELSE 'Proveedor sin Clasificar'
            END)
PRINT @clase + 'Iporte' + cast(@impt as char(10))
```

18. Como funciona la estructura de control WHILE, cite ejemplo

El bucle while se repite mientras la expresión se evalue como verdadero, es el único bucle dentro de Transact SQL. SINTAXIS:

```
WHILE <expresion>
```

```
BEGIN
      ...
END
```

EJEMPLO:Listar los porductos suministrados por el proovedor 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
```

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

A través de las instrucciones TRY y CATCH se proporciona el control de errores.

La sintaxis es la siguiente:

```
BEGIN TRY
      ...
END TRY
BEGIN CATCH
      ...
END CATCH
```

Ejemplo: Programa que detecta que hay división por 0

```
BEGIN TRY
DECLARE @divisor int, @dividiendo int, @resultado int
SET dividiendo = 100
SET divisor = 0
--esta linea provoca un error de division entre 0
SET @resultado = @dividiendo/@divisor
PRINT 'no hay error'
END TRY

BEGIN CATCH
PRINT 'Se ha producido un error'
END CATCH
```

20. Para que sirve la variable @@ERROR y que valores devuelve

Almacena el número de error producido por la ultima sentencia Transact SQL ejecutada

```
DECLARE @divisor int, @dividiendo int, @resultado int
SET dividiendo = 100
SET divisor = 0
--esta linea provoca un error de division entre 0
SET @resultado = @dividiendo/@divisor
IF(@@ERROR = 0)
	PRINT 'no hay error'
ELSE
	PRINT 'Se ha producido un error'
```