SQL SERVER 2008
Intermedio

Temario

Clase	Temas
1	Introducción a curso (Conocimientos básicos) Integridad de la base de datos Creación de Scripts SQL (1ra Parte)
2	Creación de Scripts SQL (2da Parte) Creación de Funciones de Renglón múltiple
3	Procedimientos Almacenados
4	Creación, uso y eliminación de vistas
5	Triggers (Disparadores)
6	Restricciones en Tablas
7	Modificación y eliminación de tablas con datos

Clase Anterior

- Repaso consultas

- Scripts
  - Declaración de variables (simples y tipo tabla)
  - Asignación de valores a variables (set/select)
  - Control de Flujo (If…else)

Creación de Scripts

Ejercicio

- Uso de condición en scripts

Creación de Scripts (TRANSACT-SQL)

- ✓ Lenguaje de Control de flujo
  - ❑ IF...ELSE
  - ❑ CASE (Transact-SQL)
  - ❑ WHILE
  - ❑ BREAK
  - ❑ RETURN
  - ❑ CONTINUE
  - ❑ GOTO label
  - ❑ WAITFOR

Creación de Scripts (TRANSACT-SQL)

- ✓ Lenguaje de Control de flujo

  - ❑ **IF...ELSE**

    IF expresión_booleana { sentencia sql | bloque de sentencias }
    [ELSE {sentencia sql | bloque de sentencias }]

  Ejemplo

```
DECLARE @compareprice money, @cost money

SET @cost = SELECT….

IF @cost <= @compareprice
BEGIN
PRINT ‘El precio del producto es menor’
END
ELSE
PRINT ‘El precio del producto excede la comparación’
```

Creación de Scripts (TRANSACT-SQL)

- ✓ Lenguaje de Control de flujo

  - ❑ CASE (Transact-SQL)

  - ❑ Expresión CASE simple:

```
CASE expresión_de_entrada
    WHEN expresión_cumplir THEN resultado_de_expresión[ ...n ]
    [ ELSE resultado_de_expresión]
END
```

  - ❑ Expresión CASE de búsqueda:

```
CASE
    WHEN expresión_booleana THEN resultado_de_expresión [ ...n ]
    [ ELSE resultado_de_expresión ]
END
```

Creación de Scripts (TRANSACT-SQL)

- ✓ CASE (Transact-SQL)

  - ❑ Expresión CASE simple:

```
SELECT ProdcutoId, Categoría =
                CASE LineaProducto
                        WHEN 'R' THEN ‘Ciudad'
                        WHEN 'M' THEN 'Montaña‘
                        ELSE ‘No a la venta'
                END, Name
FROM Producto
```

  - ❑ Expresión CASE de búsqueda:

```
UPDATE Empleado
SET HorasVacaciones = (
      CASE
                WHEN (HorasVacaciones -10  < 0) THEN VacationHours + 40
                ELSE (HorasVacaciones + 20.00)
      END )
```

Creación de Scripts (TRANSACT-SQL)

- **CURSOR**

```
-- Declaración del cursor
DECLARE <nombre_cursor> CURSOR
FOR

<sentencia_sql>

-- apertura del cursor
OPEN <nombre_cursor>

-- Lectura de la primera fila del cursor
FETCH NEXT FROM <nombre_cursor> INTO <lista_variables>

WHILE (@@FETCH_STATUS = 0)
BEGIN
        -- Lectura de la siguiente fila de un cursor
        FETCH NEXT FROM <nombre_cursor> INTO <lista_variables>
END -- Fin del bucle WHILE

-- Cierra el cursor
CLOSE <nombre_cursor>

-- Libera los recursos del cursor
DEALLOCATE <nombre_cursor>
```

Creación de Scripts (TRANSACT-SQL)

- **WHILE**

```
WHILE Boolean_expression
  BEGIN
        sql_statement | statement_block | BREAK | CONTINUE
  END
```

Ejemplo

```
DECLARE @inNumVuelos INT = 0

WHILE (@inNumVuelos  < 5)
BEGIN

    INSERT INTO Vuelo VALUES (234320, 1000, DATEADD(dd,7,GETDATE()))
    SELECT @inNumVuelos = count(*)
    FROM vuelo WHERE matriculaAvion = 234320
    SELECT 'Se insertó el vuelo ' + @@Identity

END
```

Creación de Scripts (TRANSACT-SQL)

- **GOTO**

label:

GOTO label

Ejemplo

```
DECLARE @inContador int = 1

WHILE @inContador < 10
BEGIN
   SELECT @inContador
   SET @inContador += 1
   IF @inContador = 4 GOTO Etiqueta_uno
   IF @inContador = 5 GOTO Etiqueta_dos
END
Etiqueta_uno:
   SELECT 'Etiqueta uno.'
   GOTO Etiqueta_tres
Etiqueta_dos:
   SELECT 'Etiqueta dos.'
Etiqueta_tres:
   SELECT 'Etiqueta tres.'
```

Creación de Scripts (TRANSACT-SQL)

- **WAITFOR**

```
WAITFOR
   DELAY ’tiempo a esperar'
  | TIME ’hora de ejecución'
```

Ejemplo

```
WAITFOR DELAY '00:01';
SELECT * from Persona

WAITFOR TIME '03:05';
SELECT * from Persona
```

Funciones Definidas por el Usuario

Funciones Definidas por el Usuario

- **Funciones Escalares**

```
CREATE FUNCTION
(
        -- Agregar parametros de entrada
        @Param1 [tipo de dato], …
)
RETURNS [tipo de dato]
AS
BEGIN
        -- Declarar variable de retorno
        DECLARE @ResultVar [tipo de dato]

        -- Sentencia(s) T-SQL

        -- Retornar el valor de la función
        RETURN @ResultVar
END
```

Funciones Definidas por el Usuario

- **Funciones Escalares (ejemplo)**

```
CREATE FUNCTION ObtenerNombreCompleto (@IdPersona INT)
RETURNS VARCHAR(100)
AS
BEGIN
        DECLARE @vcNombreCompleto VARCHAR(100)

        SELECT @vcNombreCompleto = nombre + ' ' + paterno + ' ' + ISNULL(materno,'')
        FROM persona
        WHERE IdPersona = @IdPersona

        RETURN @vcNombreCompleto
END
```

Funciones Definidas por el Usuario

- **Funciones de Tabla "en línea"**

```
CREATE FUNCTION
(
        -- Lista de parámetros
        @param1 [tipo de dato],...
)
RETURNS TABLE
AS
RETURN
(
                -- Sentencia T-SQL
)
```

Funciones Definidas por el Usuario

- **Funciones de Tabla "en línea"**

```
ALTER FUNCTION datosCompletosAvion(@vcMatricula VARCHAR(6) = NULL)
RETURNS TABLE
AS
RETURN
(
        Select a.*, b.tipoAvion, b.fabricante, b.numAsientos
        from avion                              a
            inner join tipoAvion                b
                on a.idTipoAvion = b.idTipoAvion
        where a.matricula = @vcMatricula OR @vcMatricula IS NULL
)
```

Funciones Definidas por el Usuario

- **Funciones de Tabla "sentencias multiples"**

```
CREATE FUNCTION nombre_funcion
(
      --Lista de parámetros
      @parametro [ tipo de dato ] ,...n
)
RETURNS @tabla_retorno TABLE <definición de la tabla>    [ AS ]
   BEGIN
      Sentencias T-SQL que llenen la tabla @tabla_retorno
      RETURN
   END
```

Funciones Definidas por el Usuario

- **Funciones de Tabla "sentencias multiples"**

```
CREATE FUNCTION ObtenerDireccionEmpleado (@IdEmpleado INT)
RETURNS @tbEmpleadoDireccion TABLE(
        IdEmpleado                      INT
        , nombreCompleto        VARCHAR(100)
        , direccionCompleta     VARCHAR(100)
)
AS
BEGIN
        INSERT INTO @tbEmpleadoDireccion
        SELECT a.idEmpleado,
                nombre + ' ' + paterno + ' ' + ISNULL(materno,''),
                calle + ' ' + numero + ' ' + colonia + ' ' + codigoPostal
        FROM Empleado                                   a
        INNER JOIN persona                              b
        on a.idPersona = b.IdPersona
        INNER JOIN Direccion                            c
        on b.direccionId = c.direccionId
        WHERE a.idJefe <> null
        and a.idEmpleado = @IdEmpleado

        RETURN
END
```