SQL

Parte III

Objetos

- *Database*
- *Table*
- *View*
- *Stored Procedure*
- *Function*
- *Trigger*

Stored Procedure

Es un conjunto de sentencias SQL que pueden ejecutarse, con tan solo invocar su nombre. Los Stored procedures son similares a los procedemientos que se generan en otros lenguajes de programación. En ellos se podrá:

- *Incluir 1 ó n sentencias SQL, ya sea de DML ó DDL.*
- *Aceptar parámetros de entrada y podrá emitir parámetros de salida.*
- *Se podrá llamar a otro procedure.*
- *Podrá devolver el estado de ejecución del procedure en su nombre.*
- *Utilizar estrategias de programación, tales como variables, ciclos, condicionales.*

Stored Procedure: Crear

- **Sintaxis Simple**:

```
CREATE [OR ALTER] { PROC | PROCEDURE } NombreProcedure
    ([@parametro tipodedatos [ = default ] [ OUT | OUTPUT ]] [,...n ])
[ WITH [RECOMPILE|ENCRYPTION] ]
AS
[BEGIN]
        sql_statement
[END]
```

- **Ejemplo Simple**:

```
CREATE PROCEDURE p_borrarclientes
AS
    DELETE FROM CLIENTE
```

```
CREATE PROCEDURE p_borrarclientesID (@IDDESDE int, @IDHASTA int)
AS
BEGIN
    DELETE FROM CLIENTE where idcliente between @IDDESDE and @IDHASTA
    INSERT INTO LOG VALUES (GETDATE(), 'BORRA CLIENTES')
END
```

Stored Procedure: Crear

- **Ejemplo Simple 2**:

```
CREATE PROCEDURE p_borrarclientesID (@IDDESDE int, @IDHASTA int)
AS
BEGIN

    DELETE FROM VENTA where idcliente between @IDDESDE and @IDHASTA

    DELETE FROM CLIENTE where idcliente between @IDDESDE and @IDHASTA


END


CREATE PROCEDURE p_listarclientesID (@IDDESDE int, @IDHASTA int)
AS
BEGIN

    SELECT idcliente, nombre + ' ' + apellido, fechaalta, cuit
    FROM CLIENTE
    where idcliente between @IDDESDE and @IDHASTA
    Order by idcliente

END
```

Stored Procedure: Borrar

- **Sintaxis Simple**:

```
DROP {PROC|PROCEDURE} [IF EXISTS] NombreProcedure
```

- **Ejemplo Simple**:

```
DROP PROCEDURE p_borrarclientes
```

```
DROP PROCEDURE IF EXISTS p_borrarclientesID
```

Stored Procedure: Cambiar

- **Sintaxis Simple**:

```
ALTER { PROC | PROCEDURE } NombreProcedure
    ([@parameter tipodedatos [ = default ] [ OUT | OUTPUT ]] [,...n ])
[ WITH [RECOMPILE|ENCRYPTION] ]
AS
[BEGIN]
        sql_statement
[END]
```

- **Ejemplo Simple**:

```
ALTER PROCEDURE p_borrarclientes
AS
    DELETE FROM CLIENTE
```

```
ALTER PROCEDURE p_borrarclientesID (@IDDESDE int, @IDHASTA int)
AS
BEGIN
    DELETE FROM CLIENTE where idcliente between @IDDESDE and @IDHASTA
END
```

Stored Procedure: Ejecutar

- **Sintaxis Simple**:

```
{EXEC|EXECUTE} {PROC|PROCEDURE} NombreProcedure [@Param1 [OUTPUT], ...n]
```

- **Ejemplo Simple**:

```
EXEC PROCEDURE p_borrarclientes
```

```
EXECUTE PROCEDURE p_borrarclientesID @pcliid1,@pcliid2
```

Variables

- **Declaración: Sintaxis**

```
DECLARE @nombrevariable [AS] tipodedatos [=valordefecto] [,...n]
```

- **Declaración: Ejemplo**

```
DECLARE @vapellido varchar(100)

DECLARE @id int, @vnombre varchar(100)
```

- **Asignación: Sintaxis**

```
SET @nombrevariable = <valor>
```

- **Asignación: Ejemplo**

```
SET @id = 1

SET @vapellido='Perez'
```

Estructuras: while

- **While: Sintaxis**

```
WHILE <condicion>
[BEGIN]
        <SeteneciasSQL>

        [BREAK | CONTINUE]
[END]
```

- **While: Ejemplo**

```
DECLARE @cant int
SET @cant = (select count(*) from producto)

While @cant>0
Begin

    UPDATE Producto set precio=precio*1.10 where id=@cant

    Set @cant=@cant-1

End
```

Estructuras: Condicionales

- **IF: Sintaxis**

```
IF <condicion>
[BEGIN]
        <SeteneciasSQL>
[END]
[ELSE]
[BEGIN]
        <SeteneciasSQL>
[END]
```

- **IF: Ejemplo**

```
DECLARE @sueldo numeric(10,2)
SET @Sueldo = (Select sueldo from empleado where legajo=@legajo)

If @Sueldo > 20000
    UPDATE empleado SET sueldo=sueldo*1.20 where legajo=@legajo
Else
    UPDATE empleado SET sueldo=sueldo*1.30 where legajo=@legajo
```

Stored Procedure: Ejemplos

- **Ejemplo**:

```
CREATE PROCEDURE p_nuevocliente (@razonsocial varchar(100), @cuit bigint,
@domicilio varchar(200), @mail varchar(200))
AS
BEGIN

    Declare @id int
    Set @id = (Select max(idcliente) from cliente)

    If @razonsocial is null or @razonsocial='' or @cuit=0
        Select 'No se podrá insertar el cliente. Verificar datos'
    Else
    Begin
        If len(@mail)=0
            Set @mail='NO ASIGNADO'

        INSERT INTO Cliente (id,razonsocial,cuit,domicilio,mail)
        VALUES(@id+1, @razonsocial, @cuit, @domilicio, @mail)

    end

END
```

Stored Procedure: Ejemplos

- **Ejemplo**:

```
CREATE PROCEDURE p_nuevocliente (@razonsocial varchar(100), @cuit bigint,
@domicilio varchar(200), @mail varchar(200), @newid INT output)
AS
BEGIN
    Set @newid = (Select max(idcliente) from cliente)+1

    INSERT INTO Cliente (id,razonsocial,cuit,domicilio,mail)
    VALUES(@newid, @razonsocial, @cuit, @domilicio, @mail)
END
```

- **Ejecución**:

```
declare @rz varchar(200) = 'Perez S.A.'
declare @cuit bigint = 30258761234
declare @dom varchar(200) = 'Salta 123'
declare @new int

EXEC p_nuevocliente @rz,@cuit,@dom,@new OUTPUT

print @new
```

Objetos

- *Database*
- *Table*
- *View*
- *Stored Procedure*
- *Function*
- *Trigger*

Function

Es un conjunto de sentencias SQL que pueden ejecutarse, con tan solo invocar su nombre y en su nombre devolverá una respuesta. Las funciones son similares a las funciones que se generan en otros lenguajes de programación. En ellas se podrá:

- *Incluir 1 ó n sentencias SQL DML.*
- *Aceptar parámetros de entrada y podrá emitir parámetros de salida.*
- *Se podrá invocar a otra función.*
- *Utilizar estrategias de programación, tales como variables, ciclos, condicionales.*
- *Devolver un valor en su nombre. El valor que devuelve será un valor escalar o también podrá devolver un valor de tipo table.*

Scalar Function: Crear

- **Sintaxis Simple**:

```
CREATE [ OR ALTER ] FUNCTION NombreFuncion
([@parametro tipodedatos [ = default ]]   [ ,...n ])
RETURNS return_tipodedatos
[AS]
BEGIN
      Sentencias SQL

      RETURN valor
END
```

- **Ejemplo Simple**:

```
CREATE FUNCTION f_ProximoCliente ()

RETURNS int
AS
BEGIN
    declare @ult int
    Set @ult=(select coalesce(max(idcliente),0) from Cliente)

    return @ult + 1
END
```

Table Function: Crear

- **Sintaxis Simple**:

```
CREATE [ OR ALTER ] FUNCTION NombreFuncion
([@parametro tipodedatos [ = default ]]   [ ,...n ])
RETURNS table
[AS]
      RETURN (sentencia_Select)

CREATE [ OR ALTER ] FUNCTION NombreFuncion
([@parametro tipodedatos [ = default ]]   [ ,...n ])
RETURNS @vartable table (<definición>)
[AS]
BEGIN
        <sentencia_SQL>
        INSERT INTO @vartable …
        RETURN
END
```

- **Ejemplo Simple**:

```
CREATE FUNCTION f_Clientes ()
RETURNS table
AS
    return (select idcliente,razonsocial from Cliente)
```

Table Function: Invocar

- **Ejemplo Simple**:

```
Select * from f_clientes()
```

```
CREATE FUNCTION f_Clientes ()
RETURNS table
AS
    return (select idcliente,razonsocial from Cliente)
```

```
Select f_proximocliente()

Insert into cliente values (f_proximocliente(),'Juan')
```

```
CREATE FUNCTION f_ProximoCliente ()

RETURNS int
AS
BEGIN
    declare @ult int
    Set @ult=(select coalesce(max(idcliente),1) from Cliente)

    return @ult + 1
END
```

Table Function: Borrar

- **Sintaxis Simple**:

```
DROP FUNCTION [ IF EXISTS ] NombreFunction
```

- **Ejemplo Simple**:

```
DROP FUNCTION f_Clientes

DROP FUNCTION IF EXISTS f_Clientes
```

Function: Cambiar

- **Sintaxis Simple**:

```
ALTER FUNCTION NombreFuncion
([@parametro tipodedatos [ = default ]]   [ ,...n ])
RETURNS table
[AS]
      RETURN (sentencia_Select)

ALTER FUNCTION NombreFuncion
([@parametro tipodedatos [ = default ]]   [ ,...n ])
RETURNS @vartable table (<definición>)
[AS]
BEGIN
        <sentencia_SQL>
        INSERT INTO @vartable …
        RETURN
END
```

- **Ejemplo Simple**:

```
ALTER FUNCTION f_Clientes ()
RETURNS table
AS
    return (select idcliente as idX,razonsocial from Cliente)
…
```

INVOCAR: SELECT * FROM F_CLIENTES() WHERE idX=10

Objetos

- *Database*
- *Table*
- *View*
- *Stored Procedure*
- *Function*
- *Trigger*

Trigger

Es un conjunto de sentencias SQL que sólo se disparan cuando se produce un evento. Los eventos de DML pueden ser de Update, Delete ó Insert. En ellos se podrá:

- *Incluir 1 ó n sentencias SQL, ya sea de DML ó DDL.*
- *Utilizar estrategias de programación, tales como variables, ciclos, condicionales.*
- *Utilizar el mismo trigger para distintos eventos de DML.*
- *Utilizar sólo para una única tabla o vista.*
- *Utilizar las tablas temporales deleted, inserted para verificar lo que está tratando de cambiarse en el trigger.*

Trigger: Crear

- **Sintaxis Simple**:

```
CREATE [ OR ALTER ] TRIGGER NombreTrigger
ON { table|view }
{ FOR | AFTER | INSTEAD OF}
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
AS
    Sentencias_SQL
```

- **Ejemplo Simple**:

```
CREATE TRIGGER tg_borrarclientes ON cliente INSTEAD OF DELETE
AS
    If not exists(Select 1 from factura f
                Where f.nrocliente in (select nro from deleted))
        delete from cliente
        where nro in(select nro from deleted)

CREATE TRIGGER tg_clientes ON cliente AFTER INSERT,UPDATE
AS
    UPDATE cliente
    Set fechamodificacion=getdate()
    Where nro in (Select nro from inserted)
```

Trigger: Borrar

- **Sintaxis Simple**:

```
DROP TRIGGER [IF EXISTS] NombreTrigger
```

- **Ejemplo Simple**:

```
DROP TRIGGER tg_clientes
```

```
DROP TRIGGER IF EXISTS tg_clientes
```

Trigger: Cambiar

- **Sintaxis Simple**:

```
ALTER TRIGGER NombreTrigger
ON { table|view }
{ FOR | AFTER | INSTEAD OF} {[INSERT] [,] [UPDATE] [,] [DELETE]}
AS
BEGIN
    Sentencias_SQL
END
```

- **Ejemplo Simple**:

```
ALTER TRIGGER tg_borrarclientes ON cliente INSTEAD OF DELETE
AS
BEGIN
    If not exists(Select 1 from factura f
                  Where f.nrocliente in (select nro from deleted))
        delete from cliente
        where nro in(select nro from deleted)
END

ALTER TRIGGER tg_clientes ON cliente AFTER INSERT,UPDATE
AS
BEGIN
    UPDATE cliente
    Set fechamodificacion=getdate()
    Where nro in (Select nro from inserted)
END
```

Trigger: Habilitar / Deshabilitar

- **Sintaxis Simple**:

```
ENABLE TRIGGER [NombreTrigger|ALL] ON [table|view]

DISABLE TRIGGER [NombreTrigger|ALL] ON [table|view]
```

- **Ejemplo Simple**:

```
ENABLE TRIGGER tg_clientes  on Cliente

DISABLE TRIGGER ALL on Cliente
```

Revisión SQL III

- CREATE PROCEDURE
- DROP PROCEDURE
- ALTER PROCEDURE

- CREATE FUNCTION
- DROP FUNCTION
- ALTER FUNCTION

- CREATE TRIGGER
- DROP TRIGGER
- ALTER TRIGGER
- ENABLE TRIGGER
- DISABLE TRIGGER

Consultas

Gracias.