

Procedimientos almacenados

Procedimientos almacenados en Transact SQL

Un procedimiento es un programa dentro de la base de datos que ejecuta una acción o conjunto de acciones específicas.

Un procedimiento tiene un nombre, un conjunto de parámetros (opcional) y un bloque de código.

En **Transact SQL** los procedimientos almacenados pueden devolver valores (numérico entero) o conjuntos de resultados.

```
CREATE PROCEDURE <nombre_procedure> [@param1 <tipo>, ...]  
AS  
-- Sentencias del procedimiento  
GO
```

Se ejecuta con:

```
EXEC <nombre_procedure> [VALORES PARÁMETROS]
```

El siguiente ejemplo muestra un procedimiento almacenado, denominado spu_addCliente que inserta un registro en la tabla "CLIENTES".

```
CREATE PROCEDURE spu_addCliente  
@nombre varchar(100),  
@apellido1 varchar(100),  
@apellido2 varchar(100),  
@nifCif varchar(20),  
@fxNacimiento datetime  
AS  
INSERT INTO CLIENTES  
(nombre, apellido1, apellido2, nifcif, fxnacimiento) VALUES  
(@nombre, @apellido1, @apellido2, @nifCif, @fxNacimiento)  
GO
```

Variables

```
DECLARE @variable varchar(30);
```

```
SET @find = 'Man%';
```

Declarar dando
nombre y tipo,
siempre comienzan
con @

Asignación de valor con SET

```
DECLARE @variable varchar(30) = 'Man%';
```

También inicializando su valor

Se pueden usar en las cláusulas SELECT

```
SELECT @variable  
PRINT @variable
```

Para visualizarlo

Parámetros de entrada

Los parámetros de entrada posibilitan pasar información a un procedimiento.

Para que un procedimiento almacenado admita parámetros de entrada se deben declarar variables como parámetros al crearlo. La sintaxis es:

```
create proc NOMBREPROCEDIMIENTO  
@NOMBREPARAMETRO TIPO =VALORPORDEFEECTO  
as SENTENCIAS;
```

```
create procedure pa_libros_autor  
    @autor varchar(30)  
as  
    select titulo, editorial, precio  
    from libros  
    where autor= @autor;  
go
```

Parámetros de salida

Los procedimientos almacenados pueden devolver información, para ello se emplean parámetros de salida. El valor se retorna a quien realizó la llamada con parámetros de salida. Para que un procedimiento almacenado devuelva un valor se debe declarar una variable con la palabra clave "output" al crear el procedimiento:

```
create procedure NOMBREPROCEDIMIENTO
@PARAMETROENTRADA TIPO =VALORPORDEFECTO,
@PARAMETROSALIDA TIPO=VALORPORDEFECTO output
as
    SENTENCIAS
select @PARAMETROSALIDA=SENTENCIAS;
```

```
create procedure pa_autor_sumaypromedio
@autor varchar(30)='% ',
@suma decimal(6,2) output, @promedio decimal(6,2) output
as
select titulo,editorial,precio from libros where autor like @autor
select @suma=sum(precio) from libros where autor like @autor
select @promedio=avg(precio) from libros where autor like @autor;
go
```

```
CREATE PROCEDURE Dice_Hola
AS
PRINT 'Hola';
```

go

```
EXEC Dice_Hola;
```

```
CREATE PROCEDURE Dice_Palabra
@palabra CHAR(30)
```

```
AS
```

```
PRINT @palabra;
```

```
GO
```

```
EXEC Dice_Palabra 'Lo que quiera';
```

```
EXEC Dice_Palabra 'Otra cosa';
```

```
EXEC Dice_Palabra;
```

```
CREATE PROCEDURE Dice_Palabra2 @palabra CHAR(30)='No indicó nada'
```

```
AS
```

```
PRINT @palabra;
```

```
GO
```

```
EXEC Dice_Palabra2 'Lo que quiera';
```

```
EXEC Dice_Palabra2 'Otra cosa';
```

```
EXEC Dice_Palabra2;
```

```
create table prueba          insert into prueba
(dato1 integer,              values    (1, 'uno'), (2, 'dos'), (3, 'tres'),
dato2 varchar(30))          (4, 'cuatro'), (5, 'cinco'), (6, 'seis');
```

```
select dato1,dato2 from prueba;
```

```
CREATE PROCEDURE  VerTabla
AS
    select dato1,dato2 from prueba;
go

exec VerTabla;
```

```
CREATE PROCEDURE  VerTabla2 @limite integer
AS
    select dato1,dato2 from prueba
    where dato1>@limite;
go

exec VerTabla2 2;
```



```
CREATE PROCEDURE VerTabla3 @limite integer, @filtro varchar(30)
AS
    select dato1,dato2 from prueba
    where dato1>@limite and dato2 like @filtro
go

exec VerTabla3 1, '%e%';
exec VerTabla3 @filtro='c%',@limite=2;
```

```
CREATE PROCEDURE VerTabla4 @limite integer, @filtro varchar(30) = '%'
AS
    select dato1,dato2 from prueba
    where dato1>@limite and dato2 like @filtro
go

exec VerTabla4 @limite=2;
```

```

CREATE PROCEDURE ContarRegistros
    @limite integer,
    @filtro varchar(30) = '%',
    @resultado integer output
AS
    select @resultado=(select count(*) from prueba
    where dato1>@limite and dato2 like @filtro)
go

DECLARE @valor AS integer;
EXEC ContarRegistros 1, '%o%', @valor OUTPUT;
PRINT @valor;
go

```

También podemos usar:
 Set @resultado=...
 y select @resultado=count(*)...

IF

```
-----  
--  
--procedimiento almacenado con IF  
if object_id('ver_texto','P') is not null  
    drop procedure ver_texto;  
go  
  
create procedure ver_texto  
@texto varchar(20)  
as  
IF @texto like 'A%'  
    BEGIN  
        PRINT @texto + ' comienza con A';  
    END  
ELSE  
    BEGIN  
        PRINT @texto + ' no comienza con A';  
    END  
GO  
exec ver_texto 'hola';  
exec ver_texto 'antes';
```

```
-----  
--procedimiento detectar pares  
if object_id('detecta_pares','P') is not null  
drop procedure detecta_pares;  
go  
create procedure detecta_pares  
    @numero int  
as  
IF (@numero % 2)=0  
    BEGIN  
        PRINT cast(@numero AS varchar) + ' es par';  
    END  
ELSE  
    BEGIN  
        PRINT cast(@numero AS varchar) + ' es impar';  
    END  
GO  
exec detecta_pares 44;  
exec detecta_pares 33;  
exec detecta_pares -1;
```

-- procedimiento almacenado IF con subconsulta

if object_id('libros','U') is not null

drop table libros;

go

create table libros(
 titulo varchar(40),
 autor varchar(30),
 editorial varchar(15),
 precio float,
 cantidad integer

Cargamos datos de
ejemplo

);

go

insert into libros (titulo,autor,editorial,precio,cantidad)

values

('El aleph','Borges','Emece',25.50,100),

('Alicia en el pais de las maravillas',

'Lewis Carroll','Atlantida',10,200),

('Crimen y castigo','Fiódor Dostoievski','Ed. uno',10.25,2),

('Diez negritos','Agatha Christie','E. uno', 6.50,3),

('Cien años de soledad','Gabriel García Márquez',

'Ed. siempre',10.20,7),

('Los Pilares de la Tierra','Ken Follett',

'Ed. siempre',16.40,2),

('El viejo y el mar','Ernest Hemingway',

'Ed. Santillana',6.50,4);


go


```

if object_id('actualizar_autor','P') is not null
drop procedure actualizar_autor;
go
create procedure actualizar_autor
    @autoranterior varchar(30),@autornuevo varchar(30)
as
IF EXISTS(SELECT titulo,autor,editorial,precio,cantidad FROM
libros
        WHERE autor = @autoranterior)
    BEGIN
        update libros
        set autor = @autornuevo
        where autor=@autoranterior;
    END
else
    print 'El autor indicado no está';
go

SELECT titulo,autor,editorial,precio,cantidad
FROM libros;
exec actualizar_autor 'Borges','Jorge Luis Borges';
SELECT titulo,autor,editorial,precio,cantidad
FROM libros;
exec actualizar_autor 'Cervantes','Miguel de Cervantes';

```

 Aquí lo ejecutamos



While

```
-----  
-- ejemplo de while  
if object_id('proc_contador', 'P') is not null  
drop procedure proc_contador;  
go  
  
create procedure proc_contador  
as  
DECLARE @contador int;  
SET @contador = 1;  
WHILE (@contador <= 20)  
    BEGIN  
        PRINT 'Iteracion del bucle '  
            + cast(@contador AS varchar);  
        SET @contador = @contador + 1;  
    END  
print 'Final';  
go  
exec proc_contador;
```

set nocount y @@rowcount

Set nocount on no muestra el nº de filas afectadas, al contrario que colocando off.

@@rowcount contiene el nº de registros afectados por la sentencia anterior.

-- uso de set nocount y @@rowcount

```
if object_id('ver_autores','P') is not null
drop procedure ver_autores;
go
create procedure ver_autores
as
set nocount on;
select autor
from libros;
go
exec ver_autores;
```

Partimos de la misma
tabla de libros anterior

```
if object_id('ver_libros','P') is not null
drop procedure ver_libros;
go
create procedure ver_libros
@autor varchar(30)
as
set nocount on --off
select titulo from libros
where autor like @autor;
print cast(@@rowcount as varchar) + ' registros'
go

exec ver_libros 'Cervantes'
exec ver_libros 'Borges%'
go
```

Return

La instrucción "return" sale de una consulta o procedimiento y todas las instrucciones posteriores no son ejecutadas.

```
-- uso de return
if object_id('pa_libros_autor','P') is not null
    drop procedure pa_libros_autor;
go
create procedure pa_libros_autor
    @autor varchar(30)=null
as
if @autor is null
    begin
        select 'Debe indicar un autor';
        return
    end;
select titulo from libros where autor = @autor;
go
```

Case

La función "case" compara 2 o más valores y devuelve un resultado.

La sintaxis es la siguiente:

```
select campo, campo,  
case VALORACOMPARAR  
  when VALOR1 then RESULTADO1  
  when VALOR2 then RESULTADO2  
  ...  
  else RESULTADO3  
end as nombre  
from ...
```

```
select campo,campo, nombre=  
case  
  when condición1 then valor1  
  when condición2 then valor2  
  when condición3 then valor3  
  else valor4  
end  
from ...
```

-- case

```
SELECT titulo,autor,editorial,precio,cantidad  
FROM libros;
```

```
SELECT titulo,autor,editorial,precio,cantidad,  
case  
    when precio<8 then 'Barato'  
    when precio between 8 and 12 then 'Medio'  
    else 'Caro'  
end as nivelprecio  
FROM libros;  
go
```

Concatenar cadenas de caracteres

Con las tablas de ejemplos anteriores...

El select funciona como un mientras, que recorre los registros de la consulta.

`-- concatenar resultados`

```
if object_id('concatenar_autores','P') is not null
drop procedure concatenar_autores;
go
create procedure concatenar_autores
as
declare @delimitador varchar, @autores varchar(max);
set @delimitador = ',';
set @autores = '';
select @autores=@autores +@delimitador+autor
from libros;
-- visualizamos sin la primera coma
select substring(@autores,2,len(@autores)-1);
go

exec concatenar_autores;
select autor from libros;
```

Cargar datos en Tablas temporales

Con el procedimiento anterior

```
-----  
-- ver resultado de procedimiento en tabla  
temporal  
create table #t1  
(  
autores varchar(100)  
);  
go  
insert #t1 exec concatenar_autores;  
select autores from #t1;  
-- con ## son tablas temporales globales
```

#tabla son tablas temporales que duran mientras dure la conexión actual.
##tabla son tablas temporales globales, que se mantienen mientras alguna conexión hace uso de ella.



Control de errores

```
-----  
--try catch  
if object_id('libros','U') is not null  
drop table libros;  
go  
create table libros(  
    idlibro int primary key,  
    titulo varchar(40),  
    autor varchar(30),  
    editorial varchar(15),  
    precio float,  
    cantidad integer  
);  
go
```

```

if object_id('insertar_libros') is not null
drop procedure insertar_libros;
go
create procedure insertar_libros
    @idlibro integer,
    @idtítulo varchar(40),
    @autor varchar(30),
    @editorial varchar(15),
    @precio float,
    @cantidad integer
as
begin try
    set nocount on
    insert into libros (idlibro,título,autor,editorial,precio,cantidad)
    values (@idlibro,@idtítulo,@autor,
            @editorial,@precio,@cantidad);
    select 'Registro insertado correctamente';
end try
begin catch
    SELECT ERROR_NUMBER()AS 'Nº de error',
    ERROR_MESSAGE() as 'Mensaje de error';
end catch;
go
exec insertar_libros 2,'El Quijote','Cervantes','Siglo XXI',18.8,200;
exec insertar_libros 1,'Alicia en el país de las maravillas',
'Lewis Carroll','Atlantida',10,200;
exec insertar_libros 2,'El aleph','Borges','Emece',25.50,100;
go

```

- hacer select campo de tabla especificada como parámetro

```
if object_id('proc_vertabla','P') is not null  
drop procedure proc_vertabla;  
go
```

Sp_executesql

Procedimiento para ejecutar una cadena de caracteres que contiene una sentencia SQL

```
create procedure proc_vertabla  
@tabla nvarchar(100), @campo nvarchar(100)  
as  
set nocount on  
if object_id(@tabla) is null  
begin  
print N'La tabla ' + @tabla + N' no existe'  
end  
else  
begin  
declare @sentencia nvarchar(1000);  
set @sentencia=N'select ' + @campo + N' from ' +  
@tabla;  
exec sp_executesql @sentencia;
```

go

```
exec proc_vertabla N'libros',N'autor'  
exec proc_vertabla N'libros',N'titulo'  
exec proc_vertabla N'otra',N'campo'  
go
```

Igual sin nvarchar
mediante
exec(@sentencia)

```
-----  
-- otra forma  
if object_id('proc_vertabla2','P') is not null  
drop procedure proc_vertabla2;  
go  
  
create procedure proc_vertabla2  
@tabla varchar(100), @campo varchar(100)  
as  
set nocount on  
if object_id(@tabla) is null  
begin  
    print 'La tabla ' + @tabla + ' no existe'  
end  
else  
begin  
    declare @sentencia varchar(1000);  
    set @sentencia='select '+@campo+' from ' +  
        @tabla;  
    exec(@sentencia);  
end  
go  
  
exec proc_vertabla2 'libros','autor';  
exec proc_vertabla2 'libros','titulo';  
exec proc_vertabla2 'otra','campo';  
go
```

