Procedimientos almacenados

Vimos que SQL Server ofrece dos alternativas para asegurar la integridad de datos, la integridad:

1) DECLARATIVA, mediante el uso de restricciones (constraints), valores predeterminados (defaults) y reglas (rules) y

2) PROCEDIMENTAL, mediante la implementación de procedimientos almacenados y desencadenadores (triggers).

Nos detendremos ahora en procedimientos almacenados.

Un procedimiento almacenado es un conjunto de instrucciones a las que se les da un nombre, que se almacena en el servidor. Permiten encapsular tareas repetitivas.

SQL Server permite los siguientes tipos de procedimientos almacenados:

1) del sistema: están almacenados en la base de datos "master" y llevan el prefijo "sp_"; permiten recuperar información de las tablas del sistema y pueden ejecutarse en cualquier base de datos.

2) locales: los crea el usuario (próximo tema).

3) temporales: pueden ser locales, cuyos nombres comienzan con un signo numeral (#), o globales, cuyos nombres comienzan con 2 signos numeral (##). Los procedimientos almacenados temporales locales están disponibles en la sesión de un solo usuario y se eliminan automáticamente al finalizar la sesión; los globales están disponibles en las sesiones de todos los usuarios.

4) extendidos: se implementan como bibliotecas de vínculos dinámicos (DLL, Dynamic-Link Libraries), se ejecutan fuera del entorno de SQL Server. Generalmente llevan el prefijo "xp_". No los estudiaremos.

Al crear un procedimiento almacenado, las instrucciones que contiene se analizan para verificar si son correctas sintácticamente. Si no se detectan errores, SQL Server guarda el nombre del procedimiento almacenado en la tabla del sistema "sysobjects" y su contenido en la tabla del sistema "syscomments" en la base de datos activa. Si se encuentra algún error, no se crea.

Un procedimiento almacenados puede hacer referencia a objetos que no existen al momento de crearlo. Los objetos deben existir cuando se ejecute el procedimiento almacenado.

Ventajas:

- comparten la lógica de la aplicación con las otras aplicaciones, con lo cual el acceso y las modificaciones de los datos se hacen en un solo sitio.

- permiten realizar todas las operaciones que los usuarios necesitan evitando que tengan acceso directo a las tablas.

- reducen el tráfico de red; en vez de enviar muchas instrucciones, los usuarios realizan operaciones enviando una única instrucción, lo cual disminuye el número de solicitudes entre el cliente y el servidor.

Procedimientos almacenados (crear - ejecutar)

Los procedimientos almacenados se crean en la base de datos seleccionada, excepto los procedimientos almacenados temporales, que se crean en la base de datos "tempdb".

En primer lugar se deben tipear y probar las instrucciones que se incluyen en el procedimiento almacenado, luego, si se obtiene el resultado esperado, se crea el procedimiento.

Los procedimientos almacenados pueden hacer referencia a tablas, vistas, a funciones definidas por el usuario, a otros procedimientos almacenados y a tablas temporales.

Un procedimiento almacenado pueden incluir cualquier cantidad y tipo de instrucciones, excepto:

create default, create procedure, create rule, create trigger y create view.

Se pueden crear otros objetos (por ejemplo índices, tablas), en tal caso deben especificar el nombre del propietario; se pueden realizar inserciones, actualizaciones, eliminaciones, etc.

Si un procedimiento almacenado crea una tabla temporal, dicha tabla sólo existe dentro del procedimiento y desaparece al finalizar el mismo. Lo mismo sucede con las variables.

Hemos empleado varias veces procedimientos almacenados del sistema ("sp_help", "sp_helpconstraint", etc.), ahora aprenderemos a crear nuestros propios procedimientos almacenados.

Para crear un procedimiento almacenado empleamos la instrucción "create procedure".

La sintaxis básica parcial es:

```
create procedure NOMBREPROCEDIMIENTO
 as INSTRUCCIONES;
```

Para diferenciar los procedimientos almacenados del sistema de los procedimientos almacenados locales use un prefijo diferente a "sp_" cuando les de el nombre.

Con las siguientes instrucciones creamos un procedimiento almacenado llamado "pa_libros_limite_stock" que muestra todos los libros de los cuales hay menos de 10 disponibles:

```
create proc pa_libros_limite_stock
 as
  select *from libros
  where cantidad <=10;
```

Entonces, creamos un procedimiento almacenado colocando "create procedure" (o "create proc", que es la forma abreviada), luego el nombre del procedimiento y seguido de "as" las sentencias que definen el procedimiento.

"create procedure" debe ser la primera sentencia de un lote.

Para ejecutar el procedimiento almacenado creado anteriormente tipeamos:

```
exec pa_libros_limite_stock;
```

Entonces, para ejecutar un procedimiento almacenado colocamos "execute" (o "exec") seguido del nombre del procedimiento.

Cuando realizamos un ejercicio nuevo, siempre realizamos las mismas tareas: eliminamos la tabla si existe, la creamos y luego ingresamos algunos registros. Podemos crear un procedimiento almacenado que contenga todas estas instrucciones:

```
create procedure pa_crear_libros
as
if object_id('libros')is not null
 drop table libros;
create table libros(
 codigo int identity,
 titulo varchar(40),
 autor varchar(30),
 editorial varchar(20),
 precio decimal(5,2),
 primary key(codigo)
);

insert into libros values('Uno','Richard Bach','Planeta',15);
insert into libros values('Ilusiones','Richard Bach','Planeta',18);
insert into libros values('El aleph','Borges','Emece',25);
insert into libros values('Aprenda PHP','Mario Molina','Nuevo siglo',45);
```

```
insert into libros values('Matematica estas ahi','Paenza','Nuevo siglo',12);
insert into libros values('Java en 10 minutos','Mario Molina','Paidos',35);
```

Y luego lo ejecutamos cada vez que comenzamos un nuevo ejercicio y así evitamos tipear tantas sentencias:

```
exec pa_crear_libros;
```

Procedimientos almacenados (eliminar)

Los procedimientos almacenados se eliminan con "drop procedure". Sintaxis:

```
drop procedure NOMBREPROCEDIMIENTO;
```

Eliminamos el procedimiento almacenado llamado "pa_libros_autor":

```
drop procedure pa_libros_autor;
```

Si el procedimiento que queremos eliminar no existe, aparece un mensaje de error, para evitarlo, podemos emplear esta sintaxis:

```
if object_id('NOMBREPROCEDIMIENTO') is not null
 drop procedure NOMBREPROCEDIMIENTO;
```

Eliminamos, si existe, el procedimiento "pa_libros_autor", si no existe, mostramos un mensaje:

```
if object_id('pa_libros_autor') is not null
 drop procedure pa_libros_autor
else
 select 'No existe el procedimiento "pa_libros_autor"';
```

"drop procedure" puede abreviarse con "drop proc".

Se recomienda ejecutar el procedimiento almacenado del sistema "sp_depends" para ver si algún objeto depende del procedimiento que deseamos eliminar.

Podemos eliminar una tabla de la cual dependa un procedimiento, SQL Server lo permite, pero luego, al ejecutar el procedimiento, aparecerá un mensaje de error porque la tabla referenciada no existe.

Procedimientos almacenados (parámetros de entrada)

Los procedimientos almacenados pueden recibir y devolver información; para ello se emplean parámetros, de entrada y salida, respectivamente.

Veamos los primeros. 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 =VALORPORDEFECTO
 as SENTENCIAS;
```

Los parámetros se definen luego del nombre del procedimiento, comenzando el nombre con un signo arroba (@). Los parámetros son locales al procedimiento, es decir, existen solamente dentro del mismo. Pueden declararse varios parámetros por procedimiento, se separan por comas.

Cuando el procedimiento es ejecutado, deben explicitarse valores para cada uno de los parámetros (en el orden que fueron definidos), a menos que se haya definido un valor por defecto, en tal caso, pueden omitirse. Pueden ser de cualquier tipo de dato (excepto cursor).

Luego de definir un parámetro y su tipo, opcionalmente, se puede especificar un valor por defecto; tal valor es el que asume el procedimiento al ser ejecutado si no recibe parámetros. Si no se coloca valor por defecto, un procedimiento definido con parámetros no puede ejecutarse sin valores para ellos. El valor por defecto puede ser "null" o una constante, también puede incluir comodines si el procedimiento emplea "like".

Creamos un procedimiento que recibe el nombre de un autor como parámetro para mostrar todos los libros del autor solicitado:

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

El procedimiento se ejecuta colocando "execute" (o "exec") seguido del nombre del procedimiento y un valor para el parámetro:

```
exec pa_libros_autor 'Borges';
```

Creamos un procedimiento que recibe 2 parámetros, el nombre de un autor y el de una editorial:

```
create procedure pa_libros_autor_editorial
 @autor varchar(30),
 @editorial varchar(20)
as
 select titulo, precio
  from libros
  where autor= @autor and
  editorial=@editorial;
```

El procedimiento se ejecuta colocando "execute" (o "exec") seguido del nombre del procedimiento y los valores para los parámetros separados por comas:

```
exec pa_libros_autor_editorial 'Richard Bach','Planeta';
```

Los valores de un parámetro pueden pasarse al procedimiento mediante el nombre del parámetro o por su posición. La sintaxis anterior ejecuta el procedimiento pasando valores a los parámetros por posición. También podemos emplear la otra sintaxis en la cual pasamos valores a los parámetros por su nombre:

```
exec pa_libros_autor_editorial @editorial='Planeta', @autor='Richard Bach';
```

Cuando pasamos valores con el nombre del parámetro, el orden en que se colocan puede alterarse.

No podríamos ejecutar el procedimiento anterior sin valores para los parámetros. Si queremos ejecutar un procedimiento que permita omitir los valores para los parámetros debemos, al crear el procedimiento, definir valores por defecto para cada parámetro:

```
create procedure pa_libros_autor_editorial2
 @autor varchar(30)='Richard Bach',
 @editorial varchar(20)='Planeta'
as
 select titulo, autor,editorial,precio
  from libros
  where autor= @autor and
  editorial=@editorial;
```

Podemos ejecutar el procedimiento anterior sin enviarle valores, usará los predeterminados.

Si enviamos un solo parámetro a un procedimiento que tiene definido más de un parámetro sin especificar a qué parámetro corresponde (valor por posición), asume que es el primero. Es decir, SQL Server asume que los valores se dan en el orden que fueron definidos, no se puede interrumpir la secuencia.

Si queremos especificar solamente el segundo parámetro, debemos emplear la sintaxis de paso de valores a parámetros por nombre:

```
exec pa_libros_autor_editorial2 @editorial='Paidos';
```

Podemos emplear patrones de búsqueda en la consulta que define el procedimiento almacenado y utilizar comodines como valores por defecto:

```
create proc pa_libros_autor_editorial3
 @autor varchar(30) = '%',
 @editorial varchar(30) = '%'
```

```
as
 select titulo,autor,editorial,precio
  from libros
  where autor like @autor and
  editorial like @editorial;
```

La sentencia siguiente ejecuta el procedimiento almacenado "pa_libros_autor_editorial3" enviando un valor por posición, se asume que es el primero.

```
exec pa_libros_autor_editorial3 'P%';
```

La sentencia siguiente ejecuta el procedimiento almacenado "pa_libros_autor_editorial3" enviando un valor para el segundo parámetro, para el primer parámetro toma el valor por defecto:

```
exec pa_libros_autor_editorial3 @editorial='P%';
```

También podríamos haber tipeado:

```
exec pa_libros_autor_editorial3 default, 'P%';
```

Procedimientos almacenados (parámetros de salida)

Dijimos que 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;
```

Los parámetros de salida pueden ser de cualquier tipo de datos, excepto text, ntext e image.

Creamos un procedimiento almacenado al cual le enviamos 2 números y retorna el promedio:

```
create procedure pa_promedio
 @n1 decimal(4,2),
 @n2 decimal(4,2),
 @resultado decimal(4,2) output
 as
  select @resultado=(@n1+@n2)/2;
```

Al ejecutarlo también debe emplearse "output":

```
 declare @variable decimal(4,2)
 execute pa_promedio 5,6, @variable output
 select @variable;
```

Declaramos una variable para guardar el valor devuelto por el procedimiento; ejecutamos el procedimiento enviándole 2 valores y mostramos el resultado.

La instrucción que realiza la llamada al procedimiento debe contener un nombre de variable para almacenar el valor retornado.

Creamos un procedimiento almacenado que muestre los títulos, editorial y precio de los libros de un determinado autor (enviado como parámetro de entrada) y nos retorne la suma y el promedio de los precios de todos los libros del autor enviado:

```
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;
```

Ejecutamos el procedimiento y vemos el contenido de las variables en las que almacenamos los parámetros de salida del procedimiento:

```
 declare @s decimal(6,2), @p decimal(6,2)
 execute pa_autor_sumaypromedio 'Richard Bach', @s output, @p output
 select @s as total, @p as promedio;
```

Procedimientos almacenados (return)

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

Creamos un procedimiento que muestre todos los libros de un autor determinado que se ingresa como parámetro. Si no se ingresa un valor, o se ingresa "null", se muestra un mensaje y se sale del procedimiento:

```
 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;
```

Si al ejecutar el procedimiento enviamos el valor "null" o no pasamos valor, con lo cual toma el valor por defecto "null", se muestra un mensaje y se sale; en caso contrario, ejecuta la consulta luego del "else".

"return" puede retornar un valor entero.

Un procedimiento puede retornar un valor de estado para indicar si se ha ejecutado correctamente o no.

Creamos un procedimiento almacenado que ingresa registros en la tabla "libros". Los parámetros correspondientes al título y autor DEBEN ingresarse con un valor distinto de "null", los demás son opcionales. El procedimiento retorna "1" si la inserción se realiza, es decir, si se ingresan valores para título y autor y "0", en caso que título o autor sean nulos:

```
create procedure pa_libros_ingreso
 @titulo varchar(40)=null,
 @autor varchar(30)=null,
 @editorial varchar(20)=null,
 @precio decimal(5,2)=null
as
if (@titulo is null) or (@autor is null)
 return 0
else
begin
 insert into libros values (@titulo,@autor,@editorial,@precio)
 return 1
```

```
end;
```

Para ver el resultado, debemos declarar una variable en la cual se almacene el valor devuelto por el procedimiento; luego, ejecutar el procedimiento asignándole el valor devuelto a la variable, finalmente mostramos el contenido de la variable:

```
declare @retorno int
exec @retorno=pa_libros_ingreso 'Alicia en el pais...','Lewis Carroll'
select 'Ingreso realizado=1' = @retorno
exec @retorno=pa_libros_ingreso
select 'Ingreso realizado=1' = @retorno;
```

También podríamos emplear un "if" para controlar el valor de la variable de retorno:

```
declare @retorno int;
exec @retorno=pa_libros_ingreso 'El gato con botas','Anónimo'
if @retorno=1 print 'Registro ingresado'
else select 'Registro no ingresado porque faltan datos';
```

Procedimientos almacenados (modificar)

Los procedimientos almacenados pueden modificarse, por necesidad de los usuarios o por cambios en la estructura de las tablas que referencia.

Un procedimiento almacenado existente puede modificarse con "alter procedure". Sintaxis:

```
alter procedure NOMBREPROCEDIMIENTO
 @PARAMETRO TIPO = VALORPREDETERMINADO
 as SENTENCIAS;
```

Modificamos el procedimiento almacenado "pa_libros_autor" para que muestre, además del título, la editorial y precio:

```
alter procedure pa_libros_autor
```

```
 @autor varchar(30)=null
as
if @autor is null
begin
 select 'Debe indicar un autor'
 return
end
else
 select titulo,editorial,precio
  from  libros
  where autor = @autor;
```

Si quiere modificar un procedimiento que se creó con la opción "with encryption" y quiere conservarla, debe incluirla al alterarlo.