Procedimientos Almacenados(Stored Procedures)

Son un conjunto de sentencias SQL que son agrupadas y almacenadas con un nombre particular en una BD relacional para ser reusada y ejecutada cada vez que sea necesario.

Son útiles cuando hay que ejecutar frecuentemente código SQL.

Una de las ventajas de los procedimientos almacenados es que son precompilados, esto significa que el rendimiento va a ser mucho mejor y la velocidad de acceso a los datos va a ser mál alta. También el tráfico de la red va a verse afectado positivamente porque el procedimiento almacenado se ejecuta directamente en el motor de BD.

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

La sintaxis básica parcial es:

create procedure NOMBREPROCEDIMIENTO()

begin

INSTRUCCIONES;

end

Ejemplo.

Con las siguientes instrucciones creamos un procedimiento almacenado llamado "pa\_libros\_limite\_stock" que retorna todos los libros de los cuales hay menos de 10 disponibles:

create procedure pa\_libros\_limite\_stock()

begin

select \* from libros

where stock<=10;

end

Para llamar luego al procedimiento almacenado debemos utilizar la cláusula 'call' y seguidamente el nombre del procedimiento almacenado:

call pa\_libros\_limite\_stock();

DELIMITER

Como un procedimiento almacenado puede tener muchos comandos SQL entre las palabras claves begin y end debemos informar de alguna manera a MySQL que ejecute dichos comandos. Para ello utilizamos el comando 'delimiter' cambiando el caracter ';' como fin de instrucción. Luego debemos codificar el procedimiento almacenado cambiando el delimitador con la siguiente sintaxis:

delimiter //

create procedure pa\_libros\_limite\_stock()

begin

select \* from libros

where stock<=10;

end //

delimiter ;

call pa\_libros\_limite\_stock();

Utilizamos el deliminador '//' como podría ser cualquier otro, por ejemplo '$'

### Parámetros de entrada, salida y entrada/salida

En los procedimientos almacenados podemos tener tres tipos de parámetros:

**Entrada:** Se indican poniendo la palabra reservada IN delante del nombre del parámetro. Estos parámetros no pueden cambiar su valor dentro del procedimiento, es decir, cuando el procedimiento finalice estos parámetros tendrán el mismo valor que tenían cuando se hizo la llamada al procedimiento. En programación sería equivalente al paso por valor de un parámetro.

**Salida:** Se indican poniendo la palabra reservada OUT delante del nombre del parametro. Estos parámetros cambian su valor dentro del procedimiento. Cuando se hace la llamada al procedimiento empiezan con un valor inicial y cuando finaliza la ejecución del procedimiento pueden terminar con otro valor diferente. En programación sería equivalente al paso por referencia de un parámetro.

**Entrada/Salida:** Es una combinación de los tipos IN y OUT. Estos parámetros se indican poniendo la palabra reservada IN/OUT delante del nombre del parámetro.

Ejemplo de procedimiento almacenado con parámetro de entrada IN.

delimiter //

create procedure alumnos\_con\_letra(**IN** letra char)

begin

select \* from alumno where nombre like concat(%,letra,%);

end //

delimiter;

call alumnos\_con\_letra(‘j’);

El anterior procedimiento almacenado devuelve los registros que contengan en el campo “nombre” la letra ‘j’

Ejemplo de procedimiento almacenado con parámtero de entrada (IN) y salida (OUT).

Supongamos que queremos guardar la cantidad de registros que contienen la letra ‘j’, en una variable llamada ‘cantidad’.

delimiter //

create procedure alumnos\_con\_letra(**IN** letra char, **OUT** cantidad INT)

begin

select \* from alumno where nombre like concat(%,letra,%);

end //

delimiter;

call alumnos\_con\_letra(‘j’,@cantidad);

select @cantidad;

Otro de ejemplo de procedimiento almacenado con parámtero de entrada (IN) y salida (OUT).

En este ejemplo vamos a crear dos procedimientos sin sentencias SQL:



Ejercicio 1.

Una empresa almacena los datos de sus empleados en una tabla llamada "empleados".

1- Eliminamos la tabla, si existe y la creamos:

drop table if exists empleados;

create table empleados(

documento char(8),

nombre varchar(20),

apellido varchar(20),

sueldo decimal(6,2),

cantidadhijos int,

seccion varchar(20),

primary key(documento)

);

2- Ingrese algunos registros:

insert into empleados values('22222222','Juan','Perez',300,2,'Contaduria');

insert into empleados values('22333333','Luis','Lopez',300,0,'Contaduria');

insert into empleados values ('22444444','Marta','Perez',500,1,'Sistemas');

insert into empleados values('22555555','Susana','Garcia',400,2,'Secretaria');

insert into empleados values('22666666','Jose Maria','Morales',400,3,'Secretaria');

3- Elimine el procedimiento llamado "pa\_empleados\_sueldo" si existe:

drop procedure if exists pa\_empleados\_sueldo;

4- Cree un procedimiento almacenado llamado "pa\_empleados\_sueldo" que seleccione los nombres,

apellidos y sueldos de los empleados.

delimiter //

create procedure pa\_empleados\_sueldo()

begin

select nombre,apellido,sueldo

from empleados;

end //

delimiter ;

5- Ejecute el procedimiento creado anteriormente.

call pa\_empleados\_sueldo();

6- Elimine el procedimiento llamado "pa\_empleados\_hijos" si existe:

drop procedure if exists pa\_empleados\_hijos;

7- Cree un procedimiento almacenado llamado "pa\_empleados\_hijos" que seleccione los nombres,

apellidos y cantidad de hijos de los empleados con hijos.

delimiter //

create procedure pa\_empleados\_hijos()

begin

select nombre,apellido,cantidadhijos

from empleados

where cantidadhijos>0;

end //

delimiter ;

8- Ejecute el procedimiento creado anteriormente.

call pa\_empleados\_hijos();

9- Actualice la cantidad de hijos de algún empleado sin hijos y vuelva a ejecutar el procedimiento

para verificar que ahora si aparece en la lista.

update empleados set cantidadhijos=1 where documento='22333333';

call pa\_empleados\_hijos();

Ejercicio 2.

Una empresa almacena los datos de sus empleados en una tabla llamada "empleados".

1- Eliminamos la tabla, si existe y la creamos:

drop table if exists empleados;

create table empleados(

documento char(8),

nombre varchar(20),

apellido varchar(20),

sueldo decimal(6,2),

cantidadhijos int,

seccion varchar(20),

primary key(documento)

);

2- Ingrese algunos registros:

insert into empleados values('22222222','Juan','Perez',300,2,'Contaduria');

insert into empleados values('22333333','Luis','Lopez',300,0,'Contaduria');

insert into empleados values ('22444444','Marta','Perez',500,1,'Sistemas');

insert into empleados values('22555555','Susana','Garcia',400,2,'Secretaria');

insert into empleados values('22666666','Jose Maria','Morales',400,3,'Secretaria');

3- Elimine el procedimiento llamado "pa\_empleados\_sueldo" si existe:

drop procedure if exists pa\_empleados\_sueldo;

4- Cree un procedimiento almacenado llamado "pa\_empleados\_sueldo" que seleccione los nombres,

apellidos y sueldos de los empleados que tengan un sueldo superior o igual al enviado como

parámetro.

delimiter //

create procedure pa\_empleados\_sueldo(

in p\_sueldo decimal(6,2))

begin

select nombre,apellido,sueldo

from empleados

where sueldo>=p\_sueldo;

end //

delimiter ;

5- Ejecute el procedimiento creado anteriormente con distintos valores:

call pa\_empleados\_sueldo(400);

call pa\_empleados\_sueldo(500);

6- Intente ejecute el procedimiento almacenado "pa\_empleados\_sueldo" sin parámetros.

Muestra mensaje de error.

7- Elimine el procedimiento almacenado "pa\_empleados\_actualizar\_sueldo" si existe:

drop procedure pa\_empleados\_actualizar\_sueldo;

8- Cree un procedimiento almacenado llamado "pa\_empleados\_actualizar\_sueldo" que actualice los

sueldos iguales al enviado como primer parámetro con el valor enviado como segundo parámetro.

delimiter //

create procedure pa\_empleados\_actualizar\_sueldo

(in p\_sueldoanterior decimal(6,2),

in p\_sueldonuevo decimal(6,2))

begin

update empleados set sueldo=p\_sueldonuevo

where sueldo=p\_sueldoanterior;

end //

delimiter ;

9- Ejecute el procedimiento creado anteriormente y verifique si se ha ejecutado correctamente:

call pa\_empleados\_actualizar\_sueldo(300, 350);

select \* from empleados;

10- Ejecute el procedimiento "pa\_empleados\_actualizar\_sueldo" enviando un solo parámetro.

Error.

<https://www.youtube.com/watch?v=iDiFyCF7m5M>

<https://www.youtube.com/watch?v=ZLXhrXpnHVg>