Instituto Politécnico Nacional

Escuela Superior de Cómputo

Bases de Datos

Práctica no. 6:

Procedures

Profesor: Euler Hernández Contreras

Alumno: Calva Hernández José Manuel

Grupo: 2CM12

## Índice

[Índice 2](#_Toc496810722)

[Marco Teórico 3](#_Toc496810723)

[Instrucciones 5](#_Toc496810724)

[Screenshots 9](#_Toc496810725)

[Conclusiones 15](#_Toc496810726)

[Referencias 15](#_Toc496810727)

## Marco Teórico

La mayoría de los sistemas de gestión de bases de datos incluyen una extensión del mismo SQL, llamada módulos de almacenamiento permanente (PSM), para permitir a los usuarios escribir procedimientos almacenados, llamados rutinas internas, dentro del espacio de proceso de la base de datos, en lugar de externamente. Es lo que históricamente se conoce como procedimientos almacenados de bases de datos, aunque pueden ser funciones o procedimientos. El término utilizado en el estándar SQL para los procedimientos almacenados es módulos almacenados persistentes, porque el DBMS almacena persistentemente estos programas, algo parecido a los datos persistentes almacenados por el DBMS.

Los procedimientos almacenados son útiles en las siguientes circunstancias:

* Si varias aplicaciones necesitan un mismo programa de bases de base de datos, este último se puede almacenar en el servidor e invocarlo desde esas aplicaciones. Esto reduce la duplicidad del esfuerzo y mejora la modularidad del software.
* La ejecución de un programa en el servidor puede reducir el coste derivado de la transferencia y la comunicación de datos entre el cliente y el servidor en ciertas situaciones.
* Estos procedimientos pueden mejorar la potencia de modelado de las vistas al permitir que los usuarios de bases de datos cuenten con tipos más complejos de datos derivados. Además, se pueden utilizar esos tipos para comprobar restricciones más complejas que quedan fuera de la especificación de aserciones y triggers.

En general, muchos DBMSs comerciales permiten escribir procedimientos y funciones almacenados en un lenguaje de programación de propósito general. De forma alternativa, un procedimiento almacenado puede estar compuesto por comandos SQL sencillos, como recuperaciones y actualizaciones.

En PL/SQL, un procedimiento es un bloque con nombre que tiene un conjunto opcional de parámetros. Cada parámetro contiene un nombre de parámetro, un uso (IN, OUT, IN OUT) y un tipo de dato. Un parámetro de entrada (IN) no debe cambiar dentro de un procedimiento. A un parámetro de salida (OUT) se le da un valor dentro de un procedimiento. Un parámetro de entrada-salida (IN OUT) debe tener un valor proporcionado fuera del procedimiento, pero puede cambiar dentro de éste. La especificación del tipo de datos no debe incluir ninguna limitación, como longitud. Por ejemplo, para un parámetro de hilera debe usar el tipo de dato VARCHAR2. No tiene que proporcionar la longitud en la especificación del tipo de dato para un parámetro.

Procedure Structure:

CREATE [OR REPLACE] PROCEDURE ProcedureName

[ ( Parameter1, . . . , ParameterN ) ]

IS

[ sequence of declarations ]

BEGIN

sequence of statements

[ EXCEPTION

sequence of statements to respond to exceptions ]

END;

Como el DBMS almacena persistentemente los procedimientos y las funciones, debe ser posible llamarlos desde varias interfaces SQL y técnicas de programación. Se puede utilizar la sentencia CALL del estándar SQL para invocar un procedimiento almacenado (desde una interfaz interactiva o desde SQL incrustado o SQLJ). El formato de la sentencia es el siguiente:

CALL <nombre del procedimiento o función> «lista de argumentos» ;

Si esta sentencia es llamada desde JDBC, debe asignarse a un objeto de sentencia de tipo *CallableStatement*.

Lista de excepciones comunes predefinidas en PL/SQL

**Excepción** **Cuando** **surge**

Cursor\_Already\_Open Trata de abrir un cursor abierto anteriormente

Dup\_Val\_On\_Index Trata de guardar un valor duplicado en un índice único

Invalid\_Cursor Trata de realizar una operación inválida en un cursor, como cerrar un cursor que no se abrió anteriormente

No\_Data\_Found La sentencia SELECT INTO no regresa ninguna fila

Rowtype\_Mismatch Trata de asignar valores con tipos de datos incompatibles entre un cursor y una variable

Timeout\_On\_Resource Ocurre un tiempo fuera como cuando esperamos un candado exclusivo1

Too\_Many\_Rows La sentencia SELECT INTO regresa más de una fila

## Instrucciones

1.- Cargar script tt.sql

create database tt1;

use tt1;

source ...

2.- Creación de las siguientes vistas...

a) Nombre del tt y su número de aquellos tts donde ha sido sinodal fabiola ocampo.

create view v1 as

Select t.\*

from tt t, sinodalia s, profesor p

where t.nott=s.nott

and (s.s1=p.idprof or s.s2=p.idprof or s.s3=p.idprof)

and p.nombre like "Fabiola%"

and p.apPaterno like "Ocampo%"

order by t.titulo, t.nott;

b) Mostrar las calificaciones de los tts remediales.

create view v2 as

Select califRevisor, califSinodales, nott

from presentacion

where tipo like "TT R%"

order by nott;

c) Mostrar la fecha de presentación y el tipo de tt que han sido dirigidos por el profesor Arturo Garfias.

create view v3 as

Select x.nott, x.fecha, x.tipo

from presentacion x, tt t, dirige d, profesor p

where x.nott=t.nott

and t.nott=d.nott

and d.idProf=p.idProf

and p.nombre like "Flavi%"

and p.apPaterno like "S\_nch%"

order by 1;

d) Mostrar el nombre de los profesores que han hecho un doctorado.

create view v4 as

Select p.nombre, p.apPaterno, p.apMaterno

from profesor p, gradoProf gp, gradoEstudios ge

where p.idProf=gp.idProf

and gp.idGrado=ge.idGrado

and ge.descripcion like "doctor%"

order by 2,3;

e) Mostrar los tts que se han presentado en el año 2007.

create view v5 as

Select t.\*, p.fecha

from tt t, presentacion p

where t.nott=p.nott

and fecha between "2007-01-01" and "2007-12-31"

order by nott;

f) Mostrar el nombre de los tts que ha sido sinodal la profesora Martha Rosa Cordero.

create view v6 as

Select t.\*

from tt t, sinodalia s, profesor p

where t.nott=s.nott

and (s.s1=p.idprof or s.s2=p.idprof or s.s3=p.idprof)

and p.nombre like "Martha%"

and p.apPaterno like "Cordero%"

order by t.titulo, t.nott;

g) Cuántos tts se presentaron en el año 2008.

create view v7 as

Select count(\*)

from presentacion

where fecha between "2008-01-01" and "2008-12-31";

h) Mostrar el dictamen de los tts que han sido revisados por la profesora Idalia Maldonado

create view v8 as

Select x.nott, x.dictamen

from presentacion x, tt t, sinodalia s, profesor p

where x.nott=t.nott

and t.nott=s.nott

and s.revisor=p.idprof

and p.nombre like "Idal%"

and p.appaterno like "Maldon%"

order by 2,1;

i) Mostrar el dictamen de los tts que han sido revisados por el profesor Ulises Vélez

create view v9 as

Select x.nott, x.dictamen

from presentacion x, tt t, sinodalia s, profesor p

where x.nott=t.nott

and t.nott=s.nott

and s.revisor=p.idprof

and p.nombre like "Ulise%"

and p.appaterno like "V\_le%"

order by 1,2;

3.- Primera función (procedure)

a) Conocer el número de registros donde su profe de bd ha dirigido tts.

b) Conocer el nott y la fecha de presentación de la consulta anterior

delimiter &

create procedure p1()

begin

Select count(\*)

from dirige d, profesor p

where d.idProf=p.idProf

and p.nombre like "Euler%"

and p.appaterno like "Hern%";

Select x.nott, x.fecha

from presentacion x, tt t, dirige d, profesor p

where x.nott=t.nott

and t.nott=d.nott

and d.idProf=p.idProf

and p.nombre like "Euler%"

and p.appaterno like "Hern%"

order by 1, 2;

end &

4.- Segunda función (procedure)

a) Los tts que tienen en su título el nombre de redes de computadoras

b) Conocer el nombre de sus directores de la consulta anterior

delimiter &

create procedure p2()

begin

Select \*

from tt

where titulo like "%redes de computadoras%"

order by titulo, nott;

Select t.\*, p.nombre, p.apPaterno, p.apMaterno

from tt t, dirige d, profesor p

where t.nott=d.nott

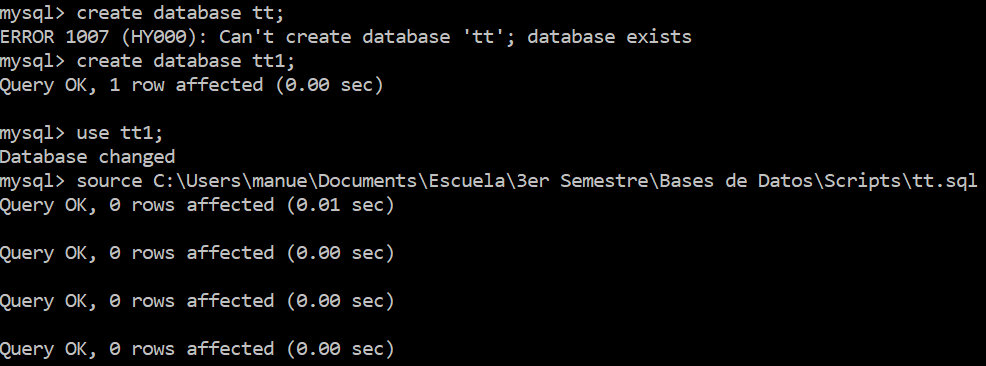
and d.idprof=p.idprof

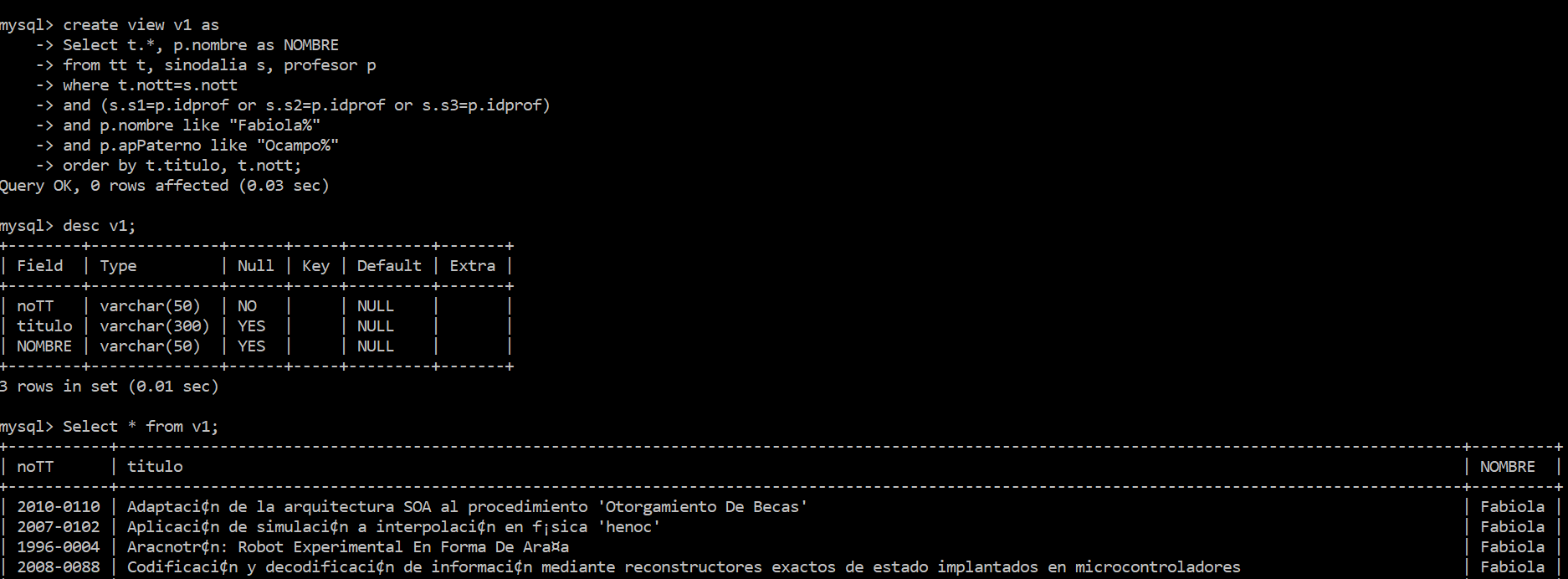
and t.titulo like "%redes de computadoras%"

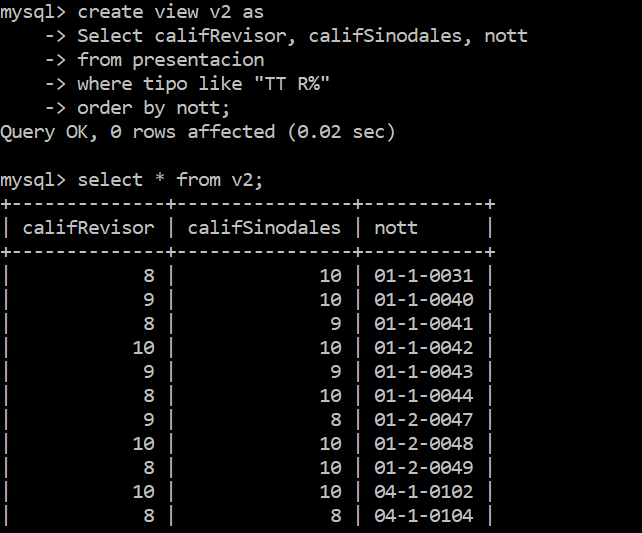
order by p.appaterno, p.apmaterno;

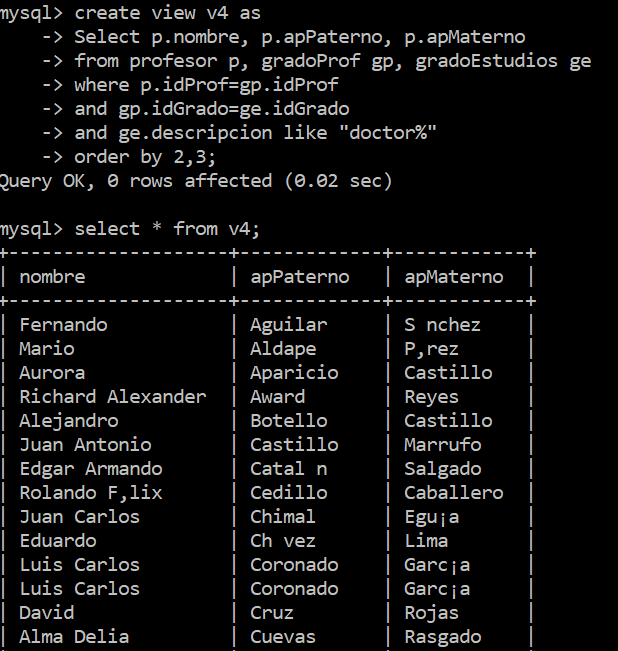
end &

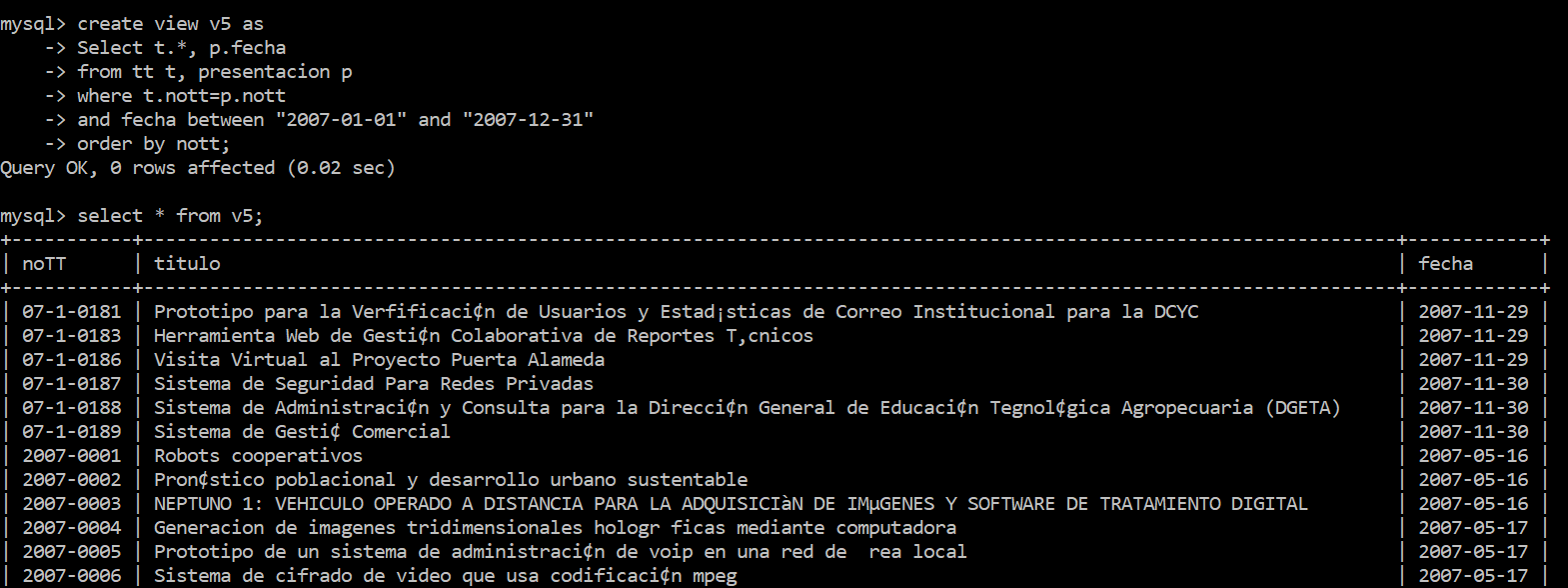
## Screenshots

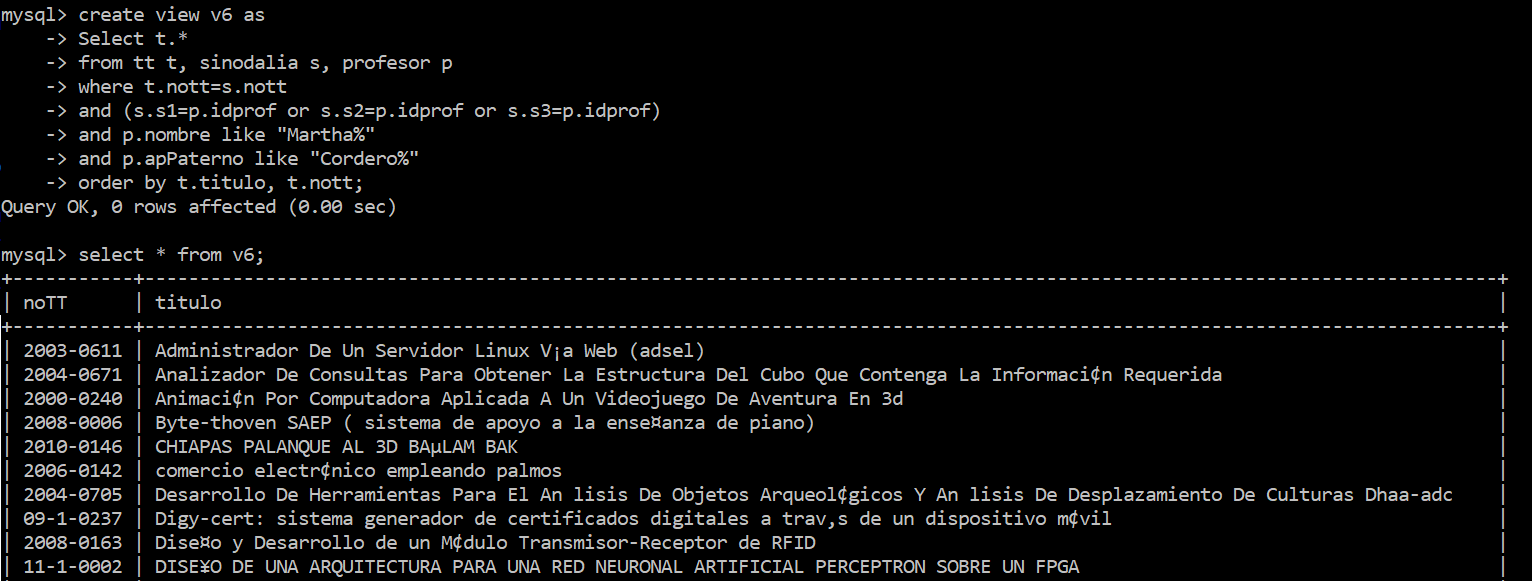


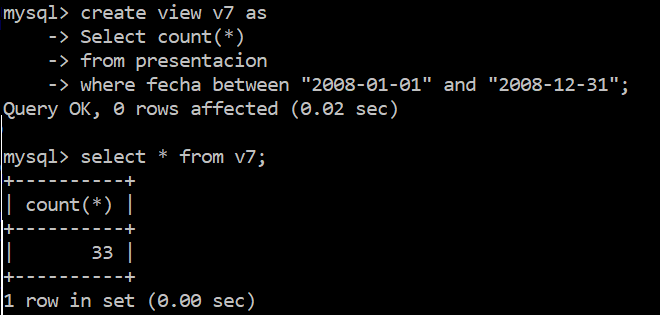


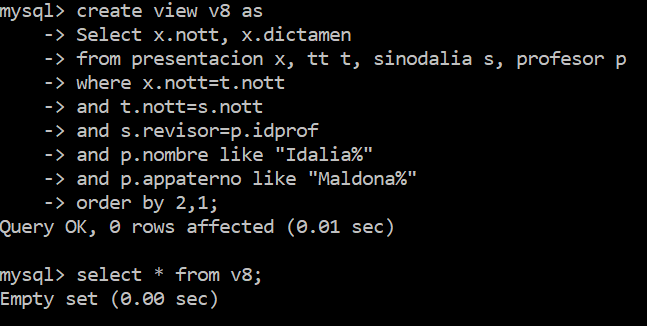


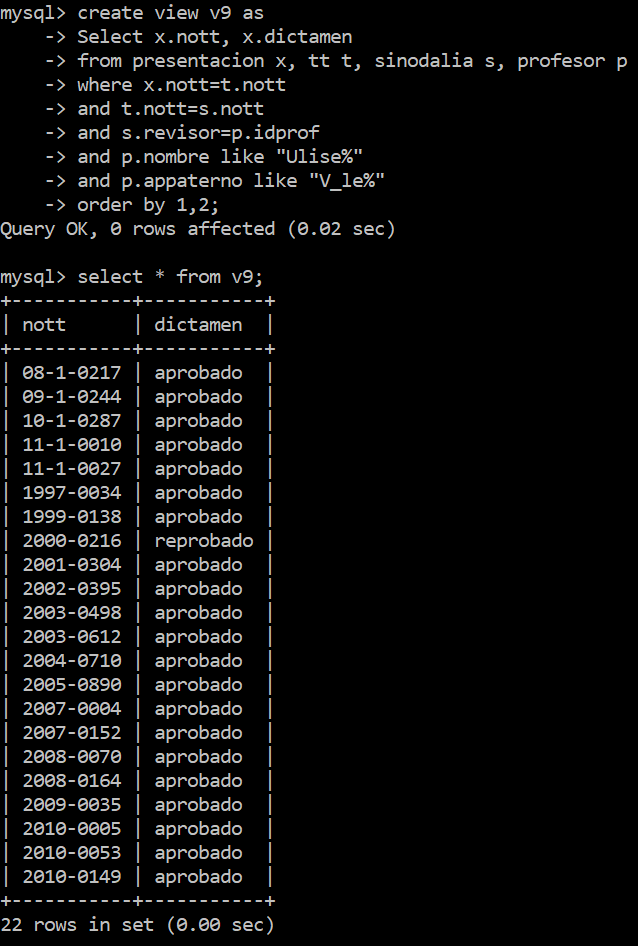


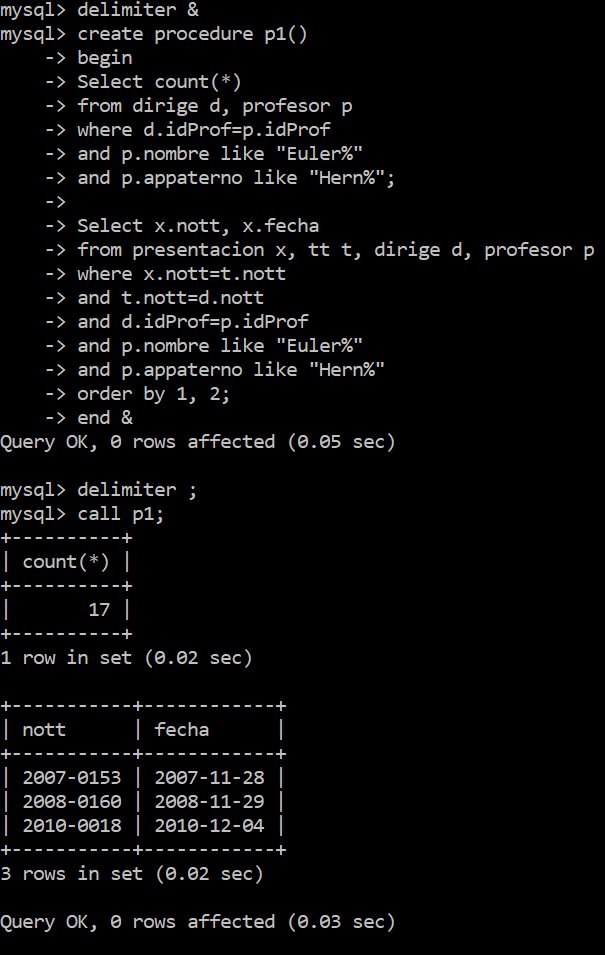


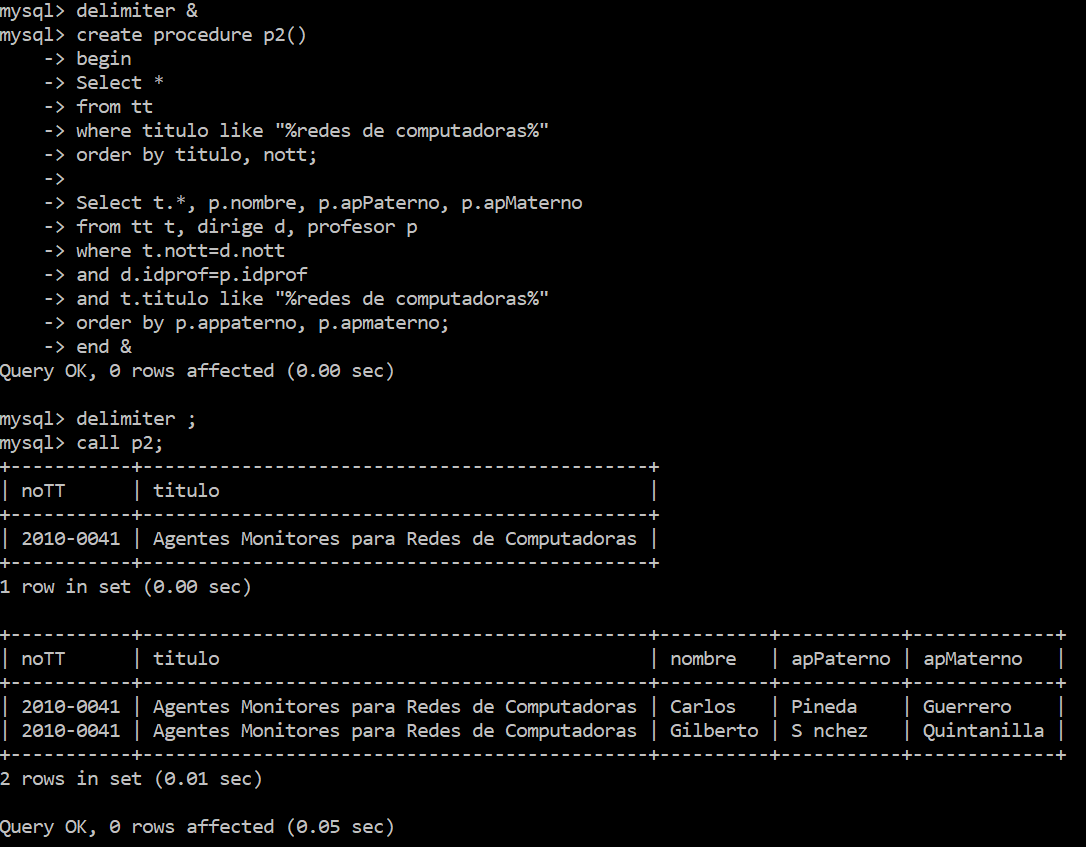












## Conclusiones

La práctica resultó como un modo de conclusión y refuerzo para la parte de vistas, ya que volvimos a aplicar vistas y hacer consultas sobre ellas, nuevamente, sin mayor complicación.

Por otra parte, iniciamos a definir procedures para nuestra base de datos, estos nos permiten realizar múltiples consultas al mismo tiempo, aunque de la teoría aprendí que es posible realizar muchas más tareas complejas sobre la base de datos, sin embargo, dado que esto fue únicamente una introducción, la mayor complicación fue aplicar correctamente la sintaxis del procedure para que nos marque un error al momento de ingresarlo en la terminar de MySQL. Esto se debe a que la sintaxis de las consultas ya la tenemos bien aprendida, por lo que lo único nuevo fue como incluir varias de ellas dentro del procedure.

## Referencias

Ramez, E., & Navathe, S. (2000). *Sistemas de Bases de Datos: Conceptos Fundamentales* (1st ed.). México: Pearson Educación.

Ricardo, C., Campos Olguín, V., & Enríquez Brito, J. (2010). *Bases de datos*. México: Mcgraw-Hill Interamericana.

Mannino, M. (2007). *Administración de Bases de Datos. Diseño y desarrollo de aplicaciones* (3rd ed.). México: McGraw-Hill Interamericana.