

EJERCICIOS DE PROGRAMACIÓN DE PROCEDIMIENTOS Y FUNCIONES CON MySQL

NOTA: LOS SIGUIENTES EJERCICIOS SE HAN DESARROLLADO PARA LAS BD DE EJEMPLO QUE VIENEN INSTALADAS POR DEFECTO CON MYSQL, SAKILA Y WORLD.

Este documento se distribuye bajo la licencia
“ReconocimientoNoComercialCompartirIgual 3.0 España” de Creative Commons, para
más información dirigirse a <http://creativecommons.org/licenses/by-sa/3.0/es/deed.es>

© Carmen Julve Tiestos

1. Crea un procedimiento que visualice todas las películas cuyo costo de reemplazo sea superior a un valor que se pasará como parámetro de entrada. ¿Cuántas películas tienen un costo de reemplazo superior a 20€?

```
CREATE DEFINER='admin'@'%' PROCEDURE `costo_reemplazo` ( IN costo decimal(5,2) )  
BEGIN  
select * from film where replacement_cost >costo;  
END
```

2. Crea un procedimiento que visualice todas las películas cuyo costo de reemplazo esté comprendido entre dos cantidades que se pasarán como parámetros de entrada. ¿Cuántas películas tienen un costo de reemplazo superior a 20€ y 21,99€?

```
CREATE DEFINER='admin'@'%' PROCEDURE `costo_reemplazo1` ( IN costo decimal(5,2),  
IN costo1 decimal(5,2) )  
BEGIN  
select * from film where replacement_cost between costo and costo1;  
END
```

3. Crea un procedimiento que reciba como parámetros de entrada el continente y la lengua y obtenga todos los países de ese continente que hablen esa lengua. ¿qué países de Asia tienen como lengua entre otras el inglés? Nota, a pesar de que el campo continente es de tipo enum, podemos pasar el continente como tipo varchar porque es compatible.

```
CREATE DEFINER='admin'@'%' PROCEDURE `continente_lengua` (IN continente  
char(40), IN lengua char(30))  
BEGIN  
select name, region, population from country  
where continent=continente and code in (select countrycode from countrylanguage where  
language=lengua);  
END
```

4. Crear una función que calcule el volumen de una esfera cuyo radio de tipo FLOAT se pasará como parámetro. Realiza una consulta después para calcular el volumen de una esfera de radio 5

```
CREATE FUNCTION volumen(radius FLOAT)  
RETURNS FLOAT DETERMINISTIC  
BEGIN  
DECLARE volume FLOAT;  
SET volume = (4/3)*PI()*POW(radius,3);  
RETURN volume;  
END $$
```

5. Escribe un procedimiento que reciba una palabra y la devuelva escrita del revés. Utiliza para ello un único parámetro de entrada y salida.

```
CREATE DEFINER=`admin`@`%` PROCEDURE `revés`(INOUT cadena varchar(25))  
BEGIN  
declare cad varchar(25) ;  
declare myc char(1);  
declare i int default 1;  
declare lon int default CHAR_LENGTH(cadena);  
set cad=SUBSTRING(cadena,lon, 1);  
WHILE i <= lon DO  
SET myc = SUBSTRING(cadena,lon-i, 1);  
SET cad=concat (cad,myc);  
SET i = i + 1;  
END WHILE;  
set cadena=cad;
```

```
set @a='HOLA'  
call revés(@a);  
select @a
```

NOTA: un argumento OUT o INOUT debe ser una variable.

6. Crear un PA que cambie el mail de un cliente, tabla costumer, por otro que se pasará como parámetro, el PA recibirá dos parámetros, el identificador del cliente y el nuevo mail. Ejecutar el PA

```
CREATE DEFINER=`admin`@`%` PROCEDURE `cambio_mail`(identi smallint(5), mail  
varchar(50) )  
BEGIN  
update customer set email=mail where customer_id=identi;  
END  
call cambio_mail(2,'patricia@gmail.com')
```

7. Crear un PA que tenga como parámetros el nombre, apellidos y el nuevo mail de un cliente de la tabla *customer*, (NOTA: debes utilizar el procedimiento anterior para cambiar el mail.)

```
CREATE DEFINER='admin'@'%' PROCEDURE `cambio_mail2` (nombre varchar(45),
apellido varchar(45), mail varchar(50))
BEGIN
declare iden smallint(5);
select customer_id into iden from customer where first_name=nombre and
last_name=apellido;
call cambio_mail(iden, mail);
END
call cambio_mail2('MARY','SMITH','mary1@gmail.com')
```

8. Crear una función que retorne el número de actor de la tabla *actor*, pasando como parámetros el apellido y nombre.

```
CREATE DEFINER='admin'@'%' FUNCTION `actor_num` (apellido varchar(45), nombre
varchar(45)) RETURNS smallint(5)
BEGIN
declare val smallint(5);
select actor_id into val from actor where last_name=apellido and first_name=nombre;
return val;
END
```

```
select actor_num('CHASE', 'ED') as numero
```

9. Realizar una función que calcule el factorial de un número. Recuerda que 0! o 1! Es 1. No utilices recursividad. Trata de utilizar ahora la recursividad ¿Qué ocurre?

```
CREATE DEFINER='admin'@'%' FUNCTION `factorial` (n integer) RETURNS int(11)
BEGIN
declare fac, i integer default 1;
if (n=0 or n=1) then
return 1;
end if;
while i<=n do
set fac=fac*i;
set i=i+1;
end while;
RETURN fac;
END
```

```
CREATE DEFINER='admin'@'%' FUNCTION `factorial` (n integer) RETURNS int(11)
BEGIN
declare fac, i integer default 1;
set i=n;
if (n=0 or n=1) then
return 1;
end if;
while i>=1 do
```

```
set fac=i*factorial(i-1);
set i=i-1;
end while;
RETURN fac;
END
```

Da un error puesto que no está permitida la recursividad en las funciones.

10. Crea un procedimiento que visualice las películas cuya categoría (comedia, drama,... se pasa como parámetro). Llama después a este procedimiento para obtener todas las películas de la categoría drama y de la categoría comedia. ¿ qué ventaja le encuentras a realizar esta consulta de esta forma a realizarla de forma directa a través de sentencias SQL?

```
CREATE DEFINER=`admin`@`%` PROCEDURE `Buscar_por_categoria`(IN categoria
varchar(25) )
BEGIN
select title, description from film
where film_id in (select film_id from film_category where category_id =
(select category_id from category where name=categoria));
END
```

11. Crea un procedimiento que pase dos parámetros de entrada, identificador de categoría e identificador de actor y obtenga los datos de las películas sobre esa categoría en las que ha trabajado ese actor. Prueba el ejemplo con el actor 182 y la categoría 2.

```
CREATE PROCEDURE `sakila`.`categoria_actor` (IN actor smallint(5), IN categ
tinyint(3))
BEGIN
select * from film where film_id in
( select film_id from film_actor where actor_id=actor)
and film_id in
(select film_id from film_category where category_id=categ);
END
```

12. Modifica el procedimiento para que tenga 3 parámetros de entrada, nombre, apellidos del actor y nombre de la categoría y visualice para un determinado actor y una determinada categoría, las películas en las que ha trabajado.(Realiza este procedimiento primero sin utilizar el ejercicio anterior y luego utilizándolo) Utiliza cualquiera de los dos procedimientos que has creado en este ejercicio para encontrar todas las películas de animación en las que ha trabajado el actor DEBBIE AKROYD. Busca ahora todas las películas Documentales en las que ha trabajado el actor ED CHASE

```
CREATE DEFINER=`admin`@`%` PROCEDURE `categoria_actor1`(IN nombre
varchar(45), IN apellido varchar(45), IN categoria varchar(45))
BEGIN
select * from film where film_id in( select film_id from film_actor
where actor_id= (select actor_id from actor where first_name=nombre and
last_name=apellido))
and film_id in (select film_id from film_category where category_id=
(select category_id from category where name=categoria));
END
```

13. Crea un procedimiento que tenga un parámetro de entrada que será el nombre de la categoría y un parámetro de salida que contendrá el número de películas para esa categoría.

```
CREATE DEFINER=`admin`@`%` PROCEDURE `new_procedure`(IN categoria
varchar(45), OUT total INT)
BEGIN
select count(*) into total from film
where film_id in (select film_id from film_category where category_id in (select category_id
from category where
name= categoria));
END
```

```
Call ('Animation', @total)
Select @total as total_animacion;
```

14. Crea un procedimiento que reciba una cadena que puede contener letras y números y sustituya los números por *. Por ejemplo si hemos introducido la cadena 12abcd23rts, devolverá **abcd**rst
Consulta el manual de referencia sobre funciones

<http://dev.mysql.com/doc/refman/5.0/es/functions.html>

```
CREATE DEFINER=`admin`@`%` PROCEDURE `sin_numero`(INOUT str varchar(50))
BEGIN
DECLARE i INT DEFAULT 1;
DECLARE myc CHAR(1);
DECLARE outstr VARCHAR(50) DEFAULT str;
WHILE i <= CHAR_LENGTH(str) DO
SET myc = SUBSTRING(str, i, 1);
IF myc IN ('1', '2', '3', '4', '5', '6', '7', '8', '9', '0') THEN
SET outstr = INSERT(outstr,i,1,'*');
END IF;
SET i = i + 1;
END WHILE;
SET str = outstr;
END
```

```
Set @str='12holaue998kte';
Call (@str)
Select @str
```

15. Desarrollar una función que devuelva el número de años completos que hay entre dos fechas que se pasan como parámetros. Utiliza la función DATEDIFF . Para visualizar el formato de la fecha con la que trabaja mysql en la sesión que estás utilizando visualiza la fecha actual utilizando la función current_date() .

```
CREATE DEFINER=`admin`@`%` FUNCTION `años_completos`(fecha1 date, fecha2
date) RETURNS int(11)
BEGIN
declare dias, comple integer;
select datediff(fecha1,fecha2)into dias;
set dias=truncate(dias/365,0);

RETURN dias;
END
```

¿Cuántos años completos han transcurrido entre el 1 de enero de 1890 y el 23 de octubre de 2012?

```
select años_completos('2012-10-23', '1890-01-01')
```

16. Escribir una función que, haciendo uso de la función anterior, devuelva los trienios que hay entre dos fechas.

```
CREATE DEFINER='admin'@'%' FUNCTION `trienios`(fecha1 date, fecha2 date)  
RETURNS int(11)  
BEGIN  
declare trienios int;  
select truncate(años_completos(fecha1,fecha2)/3,0) into trienios;  
RETURN trienios;  
END
```

17. Codificar un procedimiento que reciba una lista de hasta tres números y visualice su suma.

Nota, PL-SQL permite utilizar parámetros opcionales, mysql no, debes pasar tantos parámetros a la función o procedimiento como hayas definido en la función. Si luego no quieres pasar un parámetro en mysql, utiliza null a la hora de llamar a la función o procedimiento.

```
CREATE DEFINER='admin'@'%' FUNCTION `suma`(n1 int, n2 int, n3 int) RETURNS  
int(11)  
BEGIN  
if n2 is null  
then  
    if n3 is null  
    then  
        return n1;  
    end if;  
return n1+n3;  
end if;  
if n3 is null then  
return n1+n2;  
else  
return n1+n2+n3;  
end if;  
  
END
```

18. Escribir un procedimiento que permita borrar un actor cuyo identificador se pasará como parámetro. Si el actor cuyo número se ha pasado como parámetro no existe, aparecerá un mensaje diciendo “Ese actor no existe”. Comprueba el funcionamiento del procedimiento. ¿Qué ocurre cuando tratas de borrar un actor que ya existe? ¿Porqué?

```
CREATE DEFINER='admin'@'%' PROCEDURE `borrar_actor`(num smallint(5))  
BEGIN  
declare nume smallint(5);  
select actor_id into nume from actor where actor_id=num;  
if nume is null then  
select concat('El actor no existe', num);  
else  
delete from actor where actor_id=num;
```

end if;**END****No permite borrar el actor porque existen registros relacionados en la tabla actor_film**

19. Escribir un procedimiento que añada una nueva entrada a la tabla film_category que guarda la categoría/s a la/s que pertenece cada películas. El procedimiento recibirá como parámetros el identificador de película y el nombre de la categoría. Deberán tenerse en cuenta las siguientes situaciones: NOTA: no utilices HANDLERS

- Si no existe el film correspondiente al número pasado como parámetro, se mostrará un mensaje diciendo “ El film con nº x no existe” en x debe aparecer el número de film pasado como parámetro y se abandonará el procedimiento.
- Si no existe la categoría pasada como parámetro, se mostrará un mensaje diciendo “la categoría x no existe” donde x será el nombre de la categoría pasada como parámetro y se abandonará el procedimiento.
- Si ya existe la entrada o fila que se pretende añadir a film_category, aparecerá un mensaje diciendo el film x ya pertenece a esa categoría. En caso contrario se procederá a dar de alta la fila en la tabla film_category.
- Comprueba finalmente el procedimiento anterior para ver el funcionamiento de las distintas situaciones.

```
CREATE PROCEDURE `sakila`.`añadir_categoria` (id smallint(5), categoria
varchar(25) )
cate: BEGIN
declare num, num1 smallint(5);
declare numcate tinyint(3);
select film_id into num from film where film_id=id;
if num is null then
select concat(concat('el film ', id),' no existe');
leave cate;
end if;

select category_id into numcate from category where name=categoria;
if numcate is null then
select concat(concat('La categoria ', categoria),' no existe');
leave cate;
end if;

select film_id into num1 from film_category where film_id=id and
category_id=numcate;

if num1 is null then
insert into film_category(film_id, category_id) values (id, numcate);
else
select concat(concat('La pelicula', id),'ya pertenece a esa categoria');
end if;

END
```


20. Crear un procedimiento en el que se introduzca el identificador de una película y nos devuelva en un parámetro de salida si es de “corta duración” (length<50), “media duración”((50<length<120) o “larga duración” (length>120). Comprueba que tipo de duración tienen las películas 1, 12 y 100

```
CREATE DEFINER='admin'@`%` PROCEDURE `tipo_duracion` ( iden smallint(5), out tipo  
varchar(25) )  
BEGIN  
declare duracion smallint(5);  
select length into duracion from film where film_id=iden;  
case  
    when duracion<50 then  
        set tipo='corta duración';  
    when duracion between 50 and 120 then  
        set tipo='duración media';  
    when duracion>120 then  
        set tipo='larga duración';  
end case;  
END
```

Call (1,@a);

Select @a;

21. En la tabla country de la BD World, actualizar el GNP de un país cuyo nombre se pasará como parámetro, de la siguiente forma, si es de África, se aumentará un 1%, si es de Antartida un 1,5%, si es de Asia o Sudamérica un 1,7%, si es de Europa o NorteAmérica un 1,9% y si es de Oceanía, un 1,6%.

```
CREATE DEFINER='admin'@`%` PROCEDURE `actualiza_GPN` ( cod char(3))  
BEGIN  
declare continente varchar(25);  
select continent into continente from country where code=cod;  
case continente  
when 'Africa' then  
    update country set GNP=GNP*1.1 where code=cod;  
when 'Asia' then  
    update country set GNP=GNP*1.17 where code=cod;  
when 'Antarctica' then  
    update country set GNP=GNP*1.15 where code=cod;  
when 'Europe' then  
    update country set GNP=GNP*1.19 where code=cod;  
when 'North America' then  
    update country set GNP=GNP*1.19 where code=cod;  
when 'Oceania' then  
    update country set GNP=GNP*1.16 where code=cod;  
when 'South America' then  
    update country set GNP=GNP*1.17 where code=cod;  
end case;  
END
```

22. Realiza un procedimiento en el que se pasen como parámetros el identificador, nombre y apellidos de un actor y lo inserte en la tabla actor. ¿qué ocurre si tratas de insertar un actor con un identificador que ya existe? Modifica el procedimiento anterior tratándose esta condición (clave duplicada). En ese caso el procedimiento deberá finalizar y mostrar el mensaje *el identificador de actor ya existe*. Probar el procedimiento primero con los datos (1,'luis','lopez') y luego con los datos (300,'luis','lopez')

```
CREATE DEFINER='admin'@'%` PROCEDURE `insertar_actor` (iden smallint(5), nombre  
varchar(45), apellido varchar(45))  
BEGIN  
declare clave_duplicada condition for 1062;  
declare EXIT handler for clave_duplicada select 'el identificador de actor ya existe'.;  
insert into actor(actor_id, first_name, last_name) values (iden, nombre, apellido);  
END
```

Call insertar_actor(1,'luis','lopez')

Call insertar_actor(300,'luis','lopez')

23. Crea un procedimiento con dos parámetros de entrada, nombre y apellidos, para borrar un actor de la tabla actor. Trata ahora de borrar un actor ¿qué ocurre? Modifica ahora el procedimiento realizando el tratamiento de los posibles errores con los correspondientes mensajes. (Ten en cuenta que pueden darse dos posibles errores, que el actor no exista, o que el actor esté en otra tabla relacionado).

Comprueba el funcionamiento del procedimiento con los valores

Actor: ED CHASE

Actor: ALVARO LOPEZ

Inserta en la tabla actor la entrada (800, Julio, Perez) y aplica luego a este actor el procedimiento ¿Lo ha borrado? ¿Porqué?

```
CREATE PROCEDURE `sakila`.`borrar_actor1` (nombre varchar(25), apellido varchar(25))  
BEGIN  
declare act int;  
declare exit handler for not found select 'el actor no existe';  
declare exit handler for 1451 select 'hay registros relacionados';  
select actor_id into act from actor where first_name=nombre and last_name=apellido;  
delete from actor where first_name=nombre and last_name=apellido;  
  
END
```

24. Crea un procedimiento para añadir un nuevo registro a la tabla city haciendo el tratamiento de errores y mostrando los mensajes pertinentes.

Aplica el procedimiento con los siguientes datos para comprobar el funcionamiento del procedimiento:

(1, 'Ciudad1',3)

(601, 'Ciudad1', 300)

(601, 'ciudad1', 4)

```
CREATE PROCEDURE `sakila`.`insertar_ciudad` (id_ciudad smallint(5), nombre  
varchar(50), id_pais smallint(5) )
```

```

BEGIN
declare pais smallint(5);
declare exit handler for not found select 'el pais no existe en la BD';
declare exit handler for 1062 select 'clave de ciudad duplicada';
select country_id into pais from country where country_id=id_pais;
insert into city(city_id, city, country_id) values(id_ciudad, nombre, id_pais);

END

```

25. Crea un procedimiento para añadir un nuevo registro a la tabla film_actor, recibirá como parámetros de entrada el título de la película, nombre y apellidos del actor realiza el tratamiento de todos los posibles errores que puedan producirse.

*Si el actor no existe, debe aparecer el mensaje *el actor xxxxx no existe* y salir del procedimiento.

*Si la película no existe, debe aparecer el mensaje *la película xxxx, no existe en la BD* y se saldrá del procedimiento.

*Si la entrada ya existe en la tabla film-actor, deberá aparecer un mensaje, “entrada duplicada”

Realiza las siguientes llamadas al procedimiento con estos datos y comprueba el funcionamiento.

Película: ALIEN CENTER Actor ED CHASE debe hacer una inserción correcta

Película: ALIEN CENTER3 Actor ED CHASE debe aparecer el mensaje *la película no existe en la BD*

Película: ALIEN CENTER Actor ED1 CHASE1 debe aparecer el mensaje *el actor no existe en la BD*

Película: ALONE TRIP Actor ED CHASE debe aparecer el mensaje, entrada duplicada en la tabla film-actor

```

CREATE DEFINER=`admin`@`%` PROCEDURE `insertar_film_actor`(pelicula
varchar(255), nombre varchar(45), apellido varchar(45) )
BEGIN
declare idact, idpel smallint(5);
declare exit handler for not found
    select concat(concat ('la pelicula ', pelicula), 'no existe');
begin
    declare exit handler for not found
        select concat (concat(concat ('el actor', nombre),apellido), 'no existe');
        select actor_id into idact from actor where first_name=nombre and
last_name=apellido;
    end;

    select film_id into idpel from film where title=pelicula;
    if idact is not null then
        begin
            declare exit handler for 1062 select 'entrada duplicada en film_actor';
            insert into film_actor(actor_id, film_id) values (idact, idpel);
        end;
    end if;
end;

```

26. Crea un procedimiento de modo que permita actualizar el campo `special_features` de la tabla `film` para una determinada película. Tendrá como parámetro el identificador del film y el parametro para `special_features`, defínelo como `varchar(25)`.

Prueba el procedimiento con los datos (1, 'Special') ¿Qué ocurre? ¿Porqué? Anota el número del error obtenido y realiza el tratamiento para este error.

Prueba ahora el procedimiento para los siguientes valores

(1, 'Trailers')

(1, 'Trailers222')

```
CREATE PROCEDURE `sakila`.`actualiza_pelicula` (id smallint(5), especial varchar(25))  
BEGIN  
declare valor_erroneo condition for 1265;  
declare exit handler for valor_erroneo select 'no es válida esa opcion de special_features';  
update film set special_features=especial where film_id=id;  
END
```

27. Crea un procedimiento para dar de alta un nuevo actor, de modo que si la clave para ese nuevo actor ya existe, se pondrá como clave el valor inmediatamente superior al valor mayor de las claves. Por ejemplo si la clave mayor es 300, se pondrá como clave 301. Realiza el procedimiento utilizando el handler para el error 1062 se produce cuando se está duplicando una clave primaria al hacer una inserción.

Prueba el procedimiento con los siguientes valores

(193, 'Maria', 'Arnes')

(305, 'Julio', 'Arranz')

```
CREATE PROCEDURE `sakila`.`insertar_actor1` (id smallint(5), nombre varchar(25),  
apellido varchar(25))  
BEGIN  
declare iden smallint(5);  
declare clave_duplicada condition for 1062;  
declare exit handler for clave_duplicada  
begin  
select max(actor_id) into iden from actor;  
insert into actor(actor_id, first_name, last_name) values (iden+1, nombre, apellido);  
end;  
insert into actor(actor_id, first_name, last_name) values (id, nombre, apellido);  
  
END
```

28. En la BD World, crear un procedimiento para actualizar la población de un determinado país. Se pasarán dos parámetros, la nueva población de tipo float y el nombre del país. Realiza el procedimiento primero sin hacer el tratamiento de errores y pruébalo con los siguientes valores (Angola, 1234567891234) ¿Qué ocurre y porqué?

No permite porque la población introducida supera el valor permitido que es int(11)

Realiza ahora el tratamiento de los errores de modo que si se introduce un valor para la población mayor del permitido, se actualizará la población de ese país aumentándola un 10%. Si se introduce un país que no existe, se acabará el procedimiento con un mensaje indicando que el país no existe.

```
CREATE DEFINER=`admin`@`%` PROCEDURE `actualizar_poblacion` ( pobla float,  
nombre char(52) )  
BEGIN  
declare nom char(52);
```

```

declare exit handler for 1264 update country set population=population*0.9 where
name=nombre;
declare exit handler for not found select 'el pais no existe';
select name into nom from country where name=nombre;
update country set population=pobla where name=nombre;
END

```

29. Crea un procedimiento que muestre para cada continente, los nombres y la población de los 5 países más poblados. Utiliza un cursor que recorra los distintos continentes que aparecen en la tabla country. Observa que para cada fetch, mysql muestra el resultado en una ventana nueva. Modifica el procedimiento para que muestre a modo de informe y en una sola ventana el resultado de los 5 continentes. (Nota: para ello deberás crear una tabla en el procedimiento que almacene el resultado)

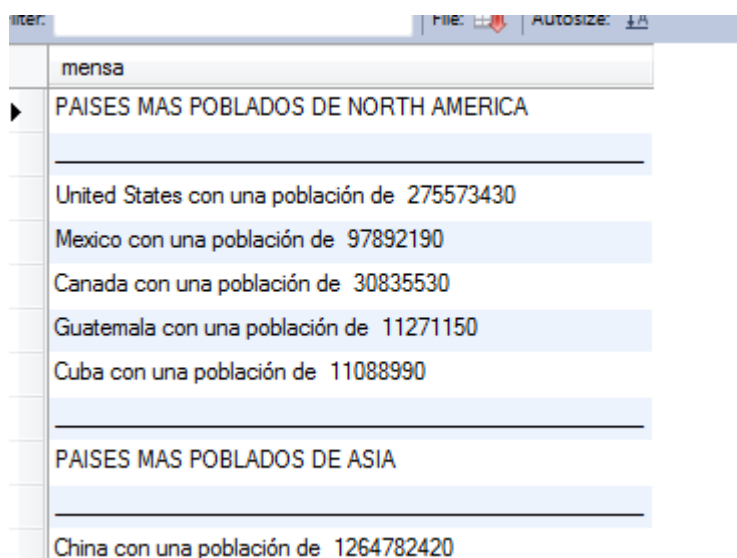
```

CREATE DEFINER=`admin`@`%` PROCEDURE `new_procedure1`()
BEGIN
declare done int default 0;
declare nombre varchar(50);
declare continente cursor for select distinct continent from country;
declare continue handler for not found set done=1;

open continente;
REPEAT
fetch continente into nombre;
if not done then
select name, population from country where continent=nombre order by population desc limit
5 ;
end if;
UNTIL done END REPEAT;
close continente;
END

```

Nota: el procedimiento anterior podría haberse realizado con
El formato de la salida, será el siguiente:



PAISES MAS POBLADOS DE NORTH AMERICA	
United States	con una población de 275573430
Mexico	con una población de 97892190
Canada	con una población de 30835530
Guatemala	con una población de 11271150
Cuba	con una población de 11088990
PAISES MAS POBLADOS DE ASIA	
China	con una población de 1264782420

```
CREATE PROCEDURE `world`.`paisemaspoplados` ()
```

```

BEGIN
declare done int default 0;
declare nombre varchar(50);
declare continente cursor for select distinct continent from country;
declare continue handler for not found set done=1;
create table if not exists paisemaspoplados (mensa varchar(255));
delete from paisemaspoplados;
open continente;
REPEAT
fetch continente into nombre;
insert into paisemaspoplados select concat('Países más poblados de ', nombre);
insert into paisemaspoplados select ' _____';
insert into paisemaspoplados select concat(concat (name, ' con una población de ', population
)) from country where continente=nombre order by population desc limit 5 ;
insert into paisemaspoplados select ' _____';
fetch continente into nombre;
UNTIL done END REPEAT;
close continente;
select * from paisemaspoplados;
drop table paisemaspoplados;

END

```

30. Crear un procedimiento que actualice el costo de reemplazo de las películas de una determinada categoría. Tendrá tres parámetros de entrada, el nombre de la categoría, un importe y un porcentaje. La subida será el porcentaje o el importe que se indica en el parámetro (el que sea superior.)

```

CREATE DEFINER=`admin`@`%` PROCEDURE `costo_actual` ( categoria varchar(50),
importe decimal(5,2), porcentaje float)
BEGIN
declare done int default 0;
declare peli smallint(5);
declare costo decimal(5,2);
declare peliculas cursor for select distinct film_id from film_category where category_id
= (select category_id from category where name=categoria);
declare continue handler for not found set done=1;

open peliculas;
repeat
fetch peliculas into peli;
select replacement_cost into costo from film where film_id=peli;
if (importe<costo*(1+porcentaje)) then
update film set replacement_cost=costo*(1+porcentaje) where film_id=peli;
else
update film set replacement_cost=importe where film_id=peli;
end if;
until done end repeat;
close peliculas;
END

```

31. Cursor que muestra un informe como el que se indica en la figura, debe mostrar para cada película de la tabla film, los actores que han trabajado en ella.

Actores que han trabajado en ACADEMY DINOSAUR	
lola hhh	
CHRISTIAN GABLE	
LUCILLE TRACY	
SANDRA PECK	
JOHNNY CAGE	
MENA TEMPLE	
WARREN NOLTE	
OPRAH KILMER	

```

CREATE DEFINER=`admin`@`%` PROCEDURE `actores`()
BEGIN

declare done int default 0;
declare iden smallint(5);
declare titulo varchar(255);
declare peliculas cursor for select film_id from film;
declare continue handler for not found set done=1;
create table if not exists actoresporpelicula (mensa varchar(255));
delete from actoresporpelicula;
open peliculas;
REPEAT
fetch peliculas into iden;

insert into actoresporpelicula select concat('Actores que han trabajado en ',title) from film
where film_id=iden;
insert into actoresporpelicula select '_____';
insert into actoresporpelicula select concat(concat(first_name,' ', last_name )) from actor where
actor_id in (select actor_id from film_actor
where film_id=iden);
insert into actoresporpelicula select '_____';

UNTIL done END REPEAT;
close peliculas;
select * from actoresporpelicula;
drop table actoresporpelicula;

END

```

32. Modificar el procedimiento anterior para que muestre los actores que han trabajado en cada película pero clasificadas por cada una de las categorías

```
CREATE DEFINER=`admin`@`%` PROCEDURE `ejercicio32`()
BEGIN
declare salir bool default false;
declare salir2 bool default false;
declare id_peli smallint(5);
declare titulo_peli varchar(255);
declare nombre varchar(45);
declare id_cat tinyint(3);
DECLARE cur2 CURSOR FOR select distinct category_id,name from category;
declare continue handler for not found set salir=true;

drop table if exists temp;
create table temp (
texto varchar(120)
);

OPEN cur2;
FETCH cur2 INTO id_cat,nombre;

REPEAT
insert into temp values(concat('Categoria: ',nombre));
begin
declare cur3 cursor for select film_id from film_category where category_id=id_cat;
declare continue handler for not found set salir2=true;
open cur3;
REPEAT
fetch cur3 into id_peli;
select title into titulo_peli from film where film_id=id_peli;
insert into temp values(concat('Actores que han trabajado en ',titulo_peli));
insert into temp values ('_____');

insert into temp SELECT concat(first_name,' ',last_name) from actor WHERE actor_id IN
(select actor_id from film_actor where film_id=id_peli);

insert into temp values ('_____');

UNTIL salir2=true
END REPEAT;
close cur3;
end;
FETCH cur2 into id_cat,nombre;
UNTIL salir=true
END REPEAT;
close cur2;
select * from temp;
drop table temp;

END
```


EJERCICIOS TRIGGERS CON MySQL

1. Visualiza los triggers de la BD sakila de dos formas diferentes.

SHOW TRIGGERS

```
select * from information_schema.triggers where trigger_Schema='sakila'
```

2. Visualiza los triggers de la BD world.

```
select * from information_schema.triggers where trigger_Schema='world'
```

3. Con la información que muestra la tabla TRIGGERS de information_schema, ¿Qué tipo de evento y cuando disparan los triggers customer_create_date, ins_film, upd_film, del_film?

sakila	customer_create_date	customer	0	NULL	SET NEW.create_date = NOW()	ROW	BEFORE
--------	----------------------	----------	---	------	-----------------------------	-----	--------

4. Visualiza los triggers de la tabla film. ¿Cuántos tiene?

```
select * from information_schema.triggers where trigger_Schema='sakila' and
event_object_table='film'
```

5. Construir un disparador de base de datos que permita auditar las operaciones de borrado de datos que se realicen en la tabla actor según las siguientes especificaciones:

En primer lugar se creará la tabla *auditaractor* con la columna *col1 VARCHAR2(200)*.

Cuando se produzca cualquier borrado de datos en esta tabla se insertará una fila en dicha tabla que contendrá:

Fecha y hora

Número de actor

Apellido de actor

Puede aparecer con el siguiente formato : El actor XX de apellido XXXX fue borrado el día XXXXX

Para probar el funcionamiento del trigger, realiza las siguientes operaciones:

*Inserta las siguientes filas en la tabla actor

(444, 'Sergio', 'Lopez')

(445, 'Sergio', 'Lopez')

(446, 'Sergio', 'Lopez')

*Borra primero la primera fila (delete from actor where actor_id=444). Comprueba los cambios en la tabla auditar_actor.

*Borra las dos siguientes filas (delete from actor where first_name='Sergio'). Comprueba los cambios en la tabla auditar_actor.

delimiter \$\$

CREATE trigger borrar_antes_actor before delete on actor

for each row

BEGIN

```
insert into tabla1 select concat (current_time, ' ', old.actor_id, ' ', old.last_name);
END;
$$
```

6. Construir un nuevo disparador de base de datos de forma que al borrar una actor introduzca en la tabla *auditaractor* el mensaje 'se ha borrado un actor'. ¿Qué ocurre? ¿Porqué?

No lo permite ya que solo se puede crear un único disparador para la misma tabla de tipo *before delete*

7. Escribir un trigger de base de datos un que permita auditar las modificaciones de nombre y apellidos en la tabla *actores* insertado en la tabla *auditaremples* los siguientes datos:

- Fecha y hora
- Número de actor
- La operación de actualización: *MODIFICACIÓN*.
- El valor anterior y el valor nuevo del nombre y el apellido

Realiza una operación de actualización sobre la tabla *actor* y comprueba los cambios en la tabla *tabla1*.

```
delimiter $$
CREATE trigger modif_desp_actor before update on actor
for each row
BEGIN
declare v_cad_inser varchar(200);
set v_cad_inser=concat (current_time, OLD.first_name, NEW.first_name);

insert into tabla1 values (v_cad_inser);
END;
$$
```

8. Escribir un disparador para la tabla *film* de modo que cuando se realice una operación de actualización el costo de reemplazo no pueda sufrir un incremento mayor del 10% es decir si la operación de actualización va a suponer un aumento mayor del 10%, se pondrá como mucho una subida del 10%.

Comprueba el funcionamiento del trigger actualizando por ejemplo el costo de reemplazo de la película con identificador 1 a un valor de 60 ¿Qué costo de reemplazo tiene ahora el film ?

```
delimiter $$
CREATE trigger before_update_film before update on film
for each row
BEGIN
declare var int;
DECLARE EXIT HANDLER FOR 1305 set var=1;

if new.replacement_cost>old.replacement_cost*1.1 then
set new.replacement_cost=old.replacement_cost*1.1;

end if;
END;
$$
```

9. Crear un trigger que solo audite las modificaciones de título o descripción de la tabla film, de modo que si se ha producido una modificación de alguna de las dos columnas, se inserte en una nueva tabla el valor antiguo y el nuevo del título y/o la descripción. Realiza comprobaciones modificando alguno de los dos campos y también haciendo actualizaciones que no afecten a uno de esos campos (en este caso el trigger no debe dispararse)

```

delimiter $$
CREATE trigger after_update_film after update on film
for each row
BEGIN
  IF (old.title != new.title)
  THEN
    if (old.description != new.description) then
      insert into tabla1 values (concat (old.title, new.title, old.description, new.description));
    else
      insert into tabla1 values (concat (old.title, new.title));
    end if;
  END IF;
  if (old.description != new.description) then
    insert into tabla1 values (concat (old.description, new.description));
  end if;
end;
$$

```

10. Busca el trigger rental_rate y explica qué es lo que hace.

11. Busca el trigger ins_film y explica qué es lo que hace.

12. Crear un trigger de modo que cada vez que se haga una operación de inserción sobre la tabla country de la BD world, automáticamente se calcule por cada continente correspondiente a ese país, el número de países y la media de la población de esos países. Estos datos se introducirán en una tabla llamada estadísticas que tendrá los siguientes campos estadísticas(id int primary key auto_increment, categoría varchar(50), num_libros int)

create table estadísticas

(id smallint(5) primary key auto_increment , num_paises int, media_poblacion int, continente varchar(50), fecha date);

delimiter \$\$

```

CREATE trigger after_insert_country after insert on country
for each row
BEGIN
  declare campo1, campo2 integer;
  select count(*), avg(population) into campo1, campo2 from country where
  continent=new.continent;
  insert into estadísticas(num_paises, media_poblacion, continente, fecha) values (
  campo1, campo2, new.continent, current_date());

  END;
  $$

```

insert into country values ('XXY','continent1','Asia','dddd',18,1901,11111,99,100,1000, 'hola2','dddd','ddd',1111,'A3')

13. Buscar ejemplos de triggers en páginas web y aplicarlos a nuestra BD sakila.