**AP06-AA7-EV03. Usar lenguaje Transaccional sobre la Base de Datos**

**Mauricio Andrés Pulgarín Araque**

**Husney Bayona Rincón**

**Cristian Leonardo Ospina Hamon**

**SENIGRAFT**

**2019**

**INTRODUCCIÓN**

En el presente documento se realizan los ejercicios propuestos en el laboratorio Usar el lenguaje transaccional sobre la base de datos, con los SGBD Oracle y Mysql respectivamente.

**Tabla de Contenido**

Mysql ………………………………………………………………………………………….4

Tablas………………………………………………………………………………………….4

Ejercicio 1 Procedimientos Almacenados Inserción de Datos Mysql…………………………5

Ejercico 3 Procedimientos Almacenados Mysql………………………………………………9

Ejercicio 5 Funciones Mysql…………………………………………………………………10

Ejercicio 8 Triggers Mysql…………………………………………………………………...13

Oracle………………………………………………………………………………………...21

Tablas………………………………………………………………………………………...21

Ejercicio 1 Procedimientos Almacenados de Isercion de Datos Oracle……………………..29

Ejercicio 2 Procedimientos Almacenados Oracle……………………………………………23

Ejercicio 6 Funciones Oracle………………………………………………………………...38

Ejercicio 8 Triggers Oracle…………………………………………………………………..43

Anexos………………………………………………………………………………………..60

**MYSQL:**

create database escuela;



USE escuela;



----------------------------------------------------------------------------------------------------------------

CREATE TABLE profesor(

doc\_prof VARCHAR(11) PRIMARY KEY,

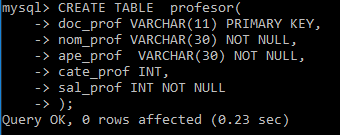
nom\_prof VARCHAR(30) NOT NULL,

ape\_prof VARCHAR(30) NOT NULL,

cate\_prof INT,

sal\_prof INT NOT NULL

);



----------------------------------------------------------------------------------------------------------------

CREATE TABLE curso(

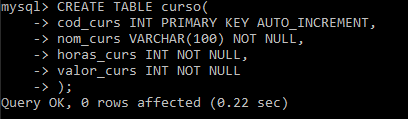
cod\_curs INT PRIMARY KEY AUTO\_INCREMENT,

nom\_curs VARCHAR(100) NOT NULL,

horas\_curs INT NOT NULL,

valor\_curs INT NOT NULL

);



----------------------------------------------------------------------------------------------------------------

CREATE TABLE estudiante(

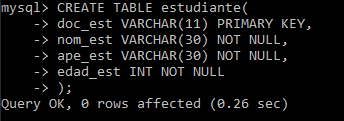
doc\_est VARCHAR(11) PRIMARY KEY,

nom\_est VARCHAR(30) NOT NULL,

ape\_est VARCHAR(30) NOT NULL,

edad\_est INT NOT NULL

);



----------------------------------------------------------------------------------------------------------------

CREATE TABLE estudiantexcurso(

cod\_cur\_estcur INT,

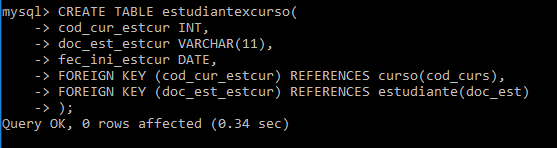
doc\_est\_estcur VARCHAR(11),

fec\_ini\_estcur DATE,

FOREIGN KEY (cod\_cur\_estcur) REFERENCES curso(cod\_curs),

FOREIGN KEY (doc\_est\_estcur) REFERENCES estudiante(doc\_est)

);



----------------------------------------------------------------------------------------------------------------

**EJERCICIO 1:**

**Procedimientos de Inserción de Datos:**

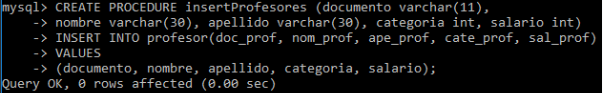
***Procedimiento Insertar Profesores.***

CREATE PROCEDURE insertProfesores(documento varchar(11),

nombre varchar(30), apellido varchar(30), categoria int, salario int)

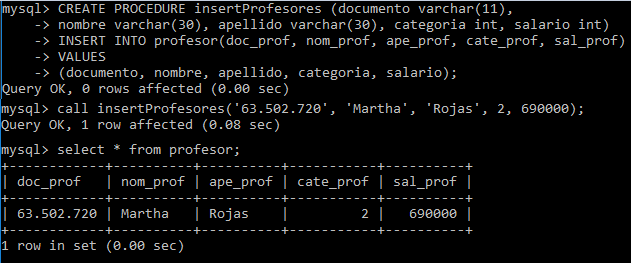
INSERT INTO profesor(doc\_prof, nom\_prof, ape\_prof, cate\_prof, sal\_prof)

VALUES

(documento, nombre, apellido, categoria, salario);

call insertProfesores('63.502.720', 'Martha', 'Rojas', 2, 690000);





----------------------------------------------------------------------------------------------------------------

***Procedimiento Insertar Estudiantes.***

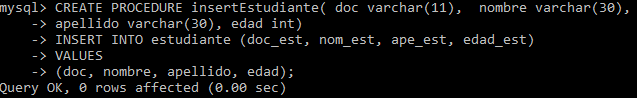
CREATE PROCEDURE insertEstudiante( doc varchar(11), nombre varchar(30),

apellido varchar(30), edad int)

INSERT INTO estudiante (doc\_est, nom\_est, ape\_est, edad\_est)

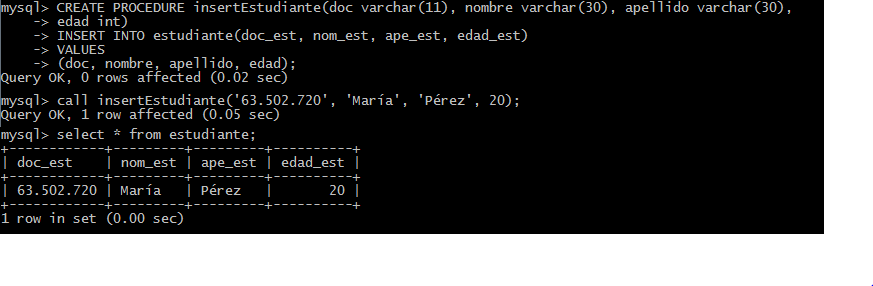
VALUES

(doc, nombre, apellido, edad);



call insertEstudiante('63.502.720', 'María', 'Pérez', 20);



----------------------------------------------------------------------------------------------------------------

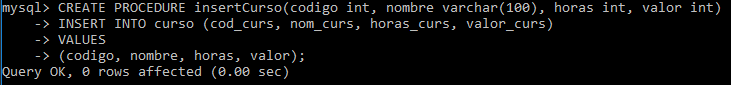
***Procedimiento Insertar Curso.***

CREATE PROCEDURE insertCurso(codigo int, nombre varchar(100), horas int, valor int)

INSERT INTO curso (cod\_curs, nom\_curs, horas\_curs, valor\_curs)

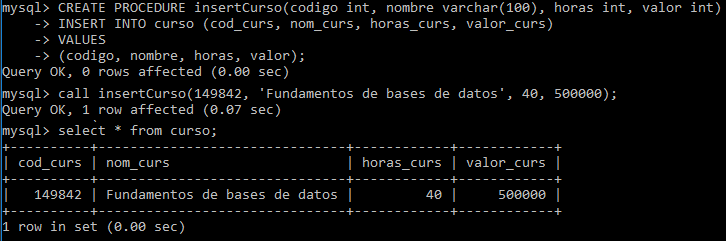
VALUES

(codigo, nombre, horas, valor);



call insertCurso(149842, 'Fundamentos de bases de datos', 40, 500000);



----------------------------------------------------------------------------------------------------------------

***Procedimiento Insertar Estudiantesxcursos.***

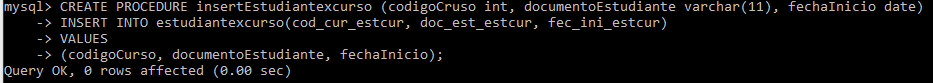
CREATE PROCEDURE insertEstudiantexcurso(codigoCurso int,

documentoEstudiante varchar(11), fechaInicio date)

INSERT INTO estudiantexcurso(cod\_cur\_estcur, doc\_est\_estcur, fec\_ini\_estcur)

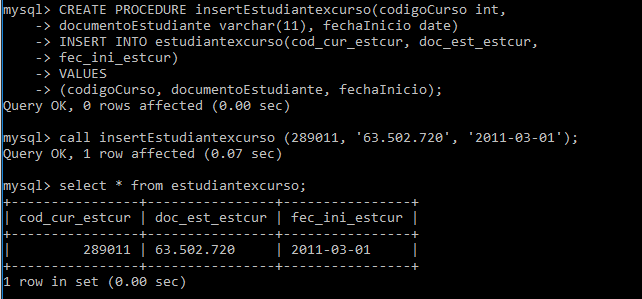
VALUES

(codigoCurso, documentoEstudiante, fechaInicio);



call insertEstudiantexcurso (289011, '63.502.720', '2011-03-01');



----------------------------------------------------------------------------------------------------------------

***Ejercicio 3 Procedimientos Almacenados.***

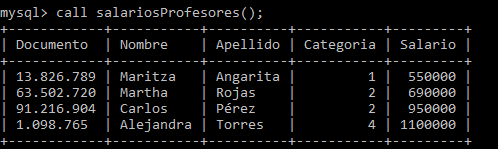
Mostrar los salarios de los profesores ordenados por categoría:

create procedure salariosProfesores () select doc\_prof Documento, nom\_prof Nombre, ape\_prof Apellido, cate\_prof Categoria, sal\_prof Salario

from profesor order by cate\_prof;



call salariosProfesores();



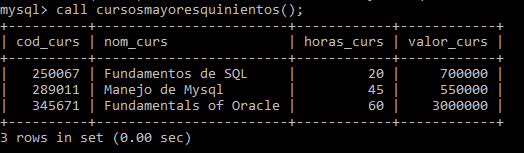
----------------------------------------------------------------------------------------------------------------

Mostrar los cursos cuyo valor sea mayor a $500.000:

create procedure cursosmayoresquinientos () select \* from curso where valor\_curs > 500000;



call cursosmayoresquinientos();



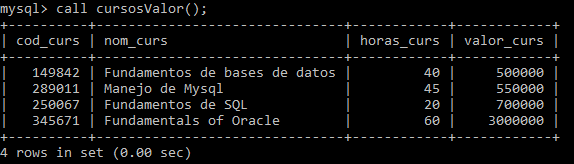
----------------------------------------------------------------------------------------------------------------

Mostrar todos los campos de la tabla curso en orden ascendente según el valor:

create procedure cursosValor() select \* from curso order by valor\_curs asc;



call cursosValor();



----------------------------------------------------------------------------------------------------------------

**Ejercicio 5:**

***FUNCIONES.***

Contar el número de estudiantes cuya edad sea mayor a 22:

DELIMITER //

CREATE FUNCTION estudiantesMayores22()

RETURNS INT

BEGIN

DECLARE cantidad int;

SELECT COUNT(\*) INTO cantidad FROM estudiante

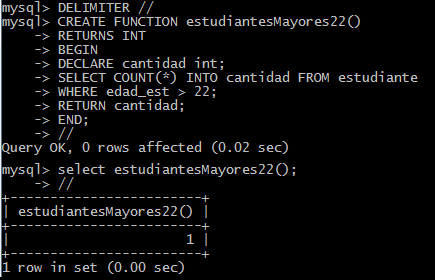
WHERE edad\_est > 22;

RETURN cantidad;

END;

//

SELECT estudiantesMayores22();

****

----------------------------------------------------------------------------------------------------------------Mostrar el nombre y la edad del estudiante más joven:

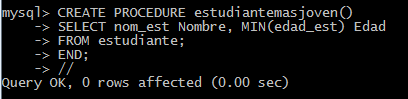
CREATE PROCEDURE estudiantemasjoven()

SELECT nom\_est Nombre, MIN(edad\_est) Edad

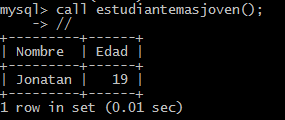
FROM estudiante;

END;

//



call estudiantemasjoven();//



Calcular el valor promedio de los cursos cuyas horas sean mayores a 40:

DELIMITER //

CREATE FUNCTION promediocursos()

RETURNS INT

BEGIN

DECLARE promedio INT;

select AVG(valor\_curs) into promedio

FROM curso

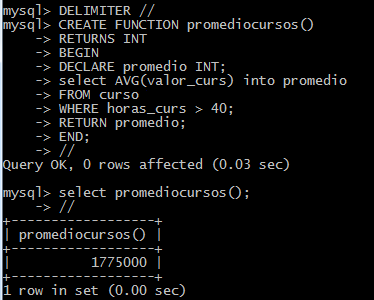
WHERE horas\_curs > 40;

RETURN promedio;

END;

//

select promediocursos();



----------------------------------------------------------------------------------------------------------------Obtener el sueldo promedio de los profesores de la categoría 1:

CREATE FUNCTION promediosueldoprofesores()

RETURNS INT

BEGIN

DECLARE promedio INT;

SELECT AVG(sal\_prof) into promedio

FROM profesor

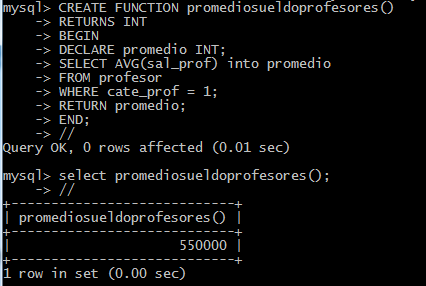
WHERE cate\_prof = 1;

RETURN promedio;

END;

//

select promediosueldoprofesores();



----------------------------------------------------------------------------------------------------------------

**Ejercico 8:**

***Triggers.***

Trigger Registro Profesores:

create table registros\_profesores(

idRegostro int primary key auto\_increment,

doc\_prof varchar(11),

old\_nom\_prof varchar(30),

old\_ape\_prof varchar(30),

old\_cate\_prof int,

old\_sal\_prof int,

new\_nom\_prof varchar(30),

new\_ape\_prof varchar(30),

new\_cate\_prof int,

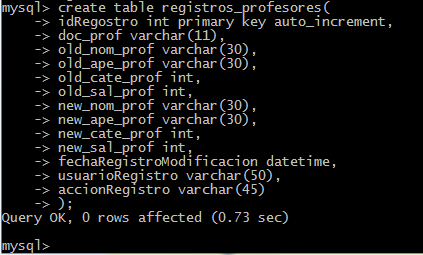
new\_sal\_prof int,

fechaRegistroModificacion datetime,

usuarioRegistro varchar(50),

accionRegistro varchar(45)

);



CREATE TRIGGER actualizacionProfesores BEFORE UPDATE ON profesor

FOR EACH ROW

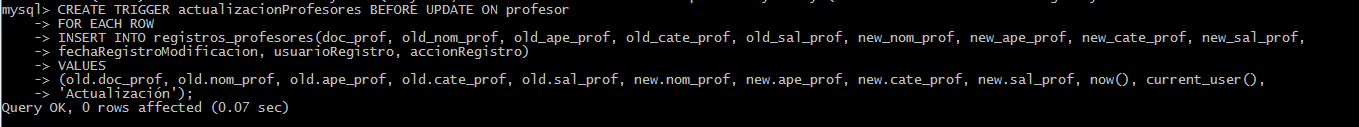
INSERT INTO registros\_profesores(doc\_prof, old\_nom\_prof, old\_ape\_prof, old\_cate\_prof, old\_sal\_prof, new\_nom\_prof, new\_ape\_prof, new\_cate\_prof, new\_sal\_prof,

fechaRegistroModificacion, usuarioRegistro, accionRegistro)

VALUES

(old.doc\_prof, old.nom\_prof, old.ape\_prof, old.cate\_prof, old.sal\_prof, new.nom\_prof, new.ape\_prof, new.cate\_prof, new.sal\_prof, now(), current\_user(),

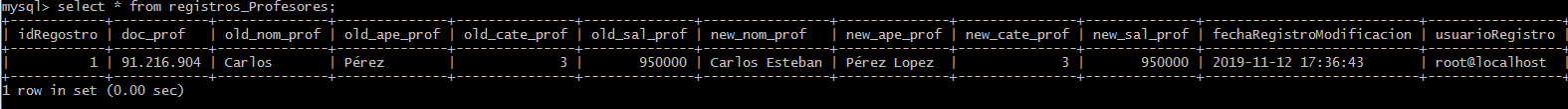
'Actualización');



update profesor set nom\_prof = 'Carlos Esteban', ape\_prof = 'Pérez Lopez' where doc\_prof = '91.216.904';

select \* from registros\_Profesores;





----------------------------------------------------------------------------------------------------------------

Trigger Registro Profesores Borrados:

CREATE TABLE registroProfesoresBorrados(

idBorrado int primary key auto\_increment,

doc\_prof\_borrado varchar(11),

nom\_prof\_borrado varchar(30),

ape\_prof\_borrado varchar(30),

cate\_prof\_borrado int,

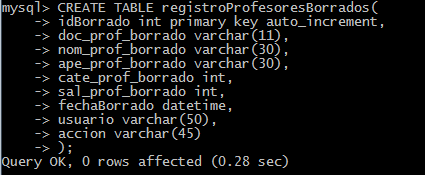
sal\_prof\_borrado int,

fechaBorrado datetime,

usuario varchar(50),

accion varchar(45)

);



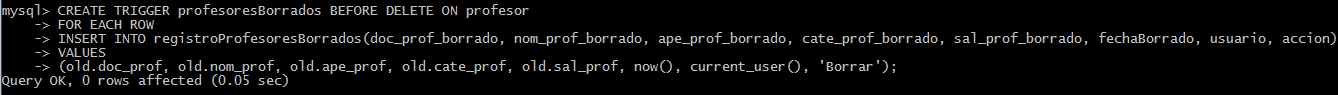
CREATE TRIGGER profesoresBorrados BEFORE DELETE ON profesor

FOR EACH ROW

INSERT INTO registroProfesoresBorrados(doc\_prof\_borrado, nom\_prof\_borrado, ape\_prof\_borrado, cate\_prof\_borrado, sal\_prof\_borrado, fechaBorrado, usuario, accion)

VALUES

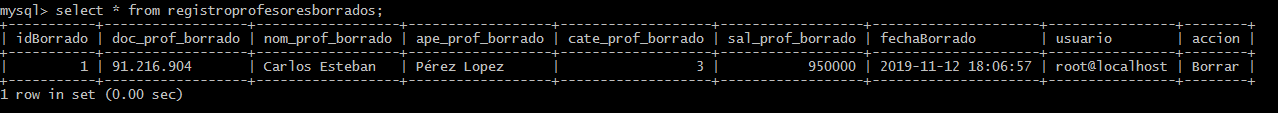
(old.doc\_prof, old.nom\_prof, old.ape\_prof, old.cate\_prof, old.sal\_prof, now(), current\_user(), 'Borrar');



delete from profesor where nom\_prof = ‘Carlos Esteban’;



select \* from registroprofesoresborrados;



----------------------------------------------------------------------------------------------------------------

Trigger Registro Curso:

CREATE TABLE registrocursos(

idRegistro int primary key auto\_increment,

old\_cod\_curs int,

old\_nom\_curs varchar(100),

old\_horas\_curs int,

old\_valor\_curs int,

new\_nom\_curs varchar(100),

new\_horas\_curs int,

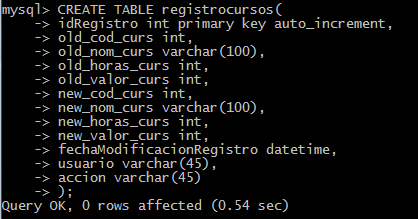
new\_valor\_curs int,

fechaModificacionRegistro datetime,

usuario varchar(45),

accion varchar(45)

);



CREATE TRIGGER actualizacionRegistrosCursos BEFORE UPDATE ON curso

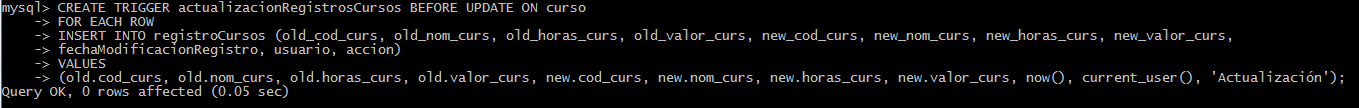
FOR EACH ROW

INSERT INTO registroCursos (old\_cod\_curs, old\_nom\_curs, old\_horas\_curs, old\_valor\_curs, new\_cod\_curs, new\_nom\_curs, new\_horas\_curs, new\_valor\_curs,

fechaModificacionRegistro, usuario, accion)

VALUES

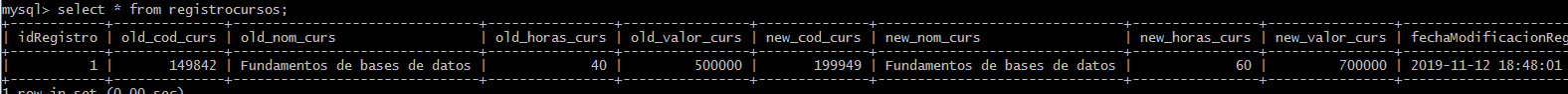
(old.cod\_curs, old.nom\_curs, old.horas\_curs, old.valor\_curs, new.cod\_curs, new.nom\_curs, new.horas\_curs, new.valor\_curs, now(), current\_user(), 'Actualización');



update curso set cod\_curs = 199949, horas\_curs = 60, valor\_curs = 700000 where nom\_curs = 'Fundamentos de bases de datos';



select \* from registrocursos;



Trigger Registro Cursos Borrados:

CREATE TABLE registroCursosBorrados(

idRegistro int primary key auto\_increment,

cod\_curs\_borrado int,

nom\_curs\_borrado varchar(100),

horas\_curs\_borrado int,

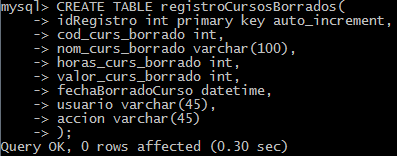
valor\_curs\_borrado int,

fechaBorradoCruso datetime,

usuario varchar(45),

accion varchar(45)

);



CREATE TRIGGER cursosBorrados BEFORE DELETE ON curso

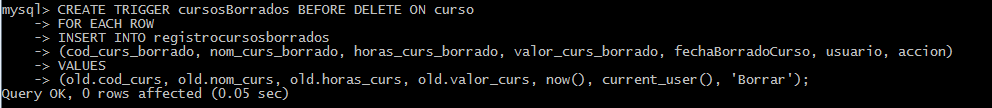
FOR EACH ROW

INSERT INTO registrocursosborrados

(cod\_curs\_borrado, nom\_curs\_borrado, horas\_curs\_borrado, valor\_curs\_borrado, fechaBorradoCurso, usuario, accion)

VALUES

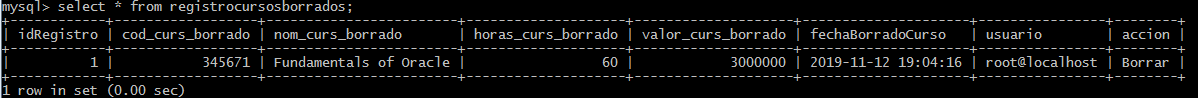
(old.cod\_curs, old.nom\_curs, old.horas\_curs, old.valor\_curs, now(), current\_user(), 'Borrar');



DELETE FROM curso WHERE id\_curs = 345671;



select \* from registrocursosborrados;

----------------------------------------------------------------------------------------------------------------

Trigger Registro Actualización Estudiante:

CREATE TABLE registroEstudiantes(

idRegistro int primary key auto\_increment,

old\_doc\_est varchar(11),

old\_nom\_est varchar(30),

old\_ape\_est varchar(30),

old\_edad\_est int,

new\_doc\_est varchar(11),

new\_nom\_est varchar(30),

new\_ape\_est varchar(30),

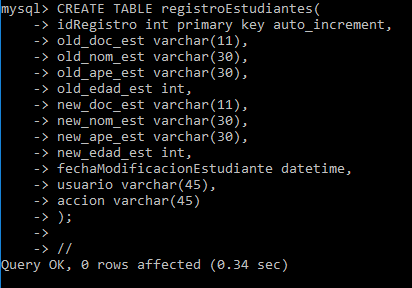
new\_edad\_est int,

fechaModificacionEstudiante datetime,

usuario varchar(45),

accion varchar(45)

);



CREATE TRIGGER acutalizacionestudiantes BEFORE UPDATE ON estudiante

FOR EACH ROW

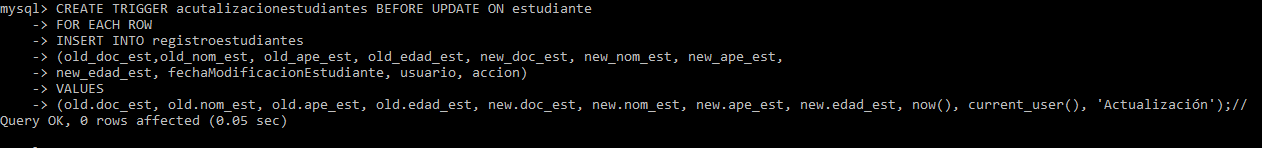
INSERT INTO registroestudiantes

(old\_doc\_est,old\_nom\_est, old\_ape\_est, old\_edad\_est, new\_doc\_est, new\_nom\_est, new\_ape\_est,

new\_edad\_est, fechaModificacionEstudiante, usuario, accion)

VALUES

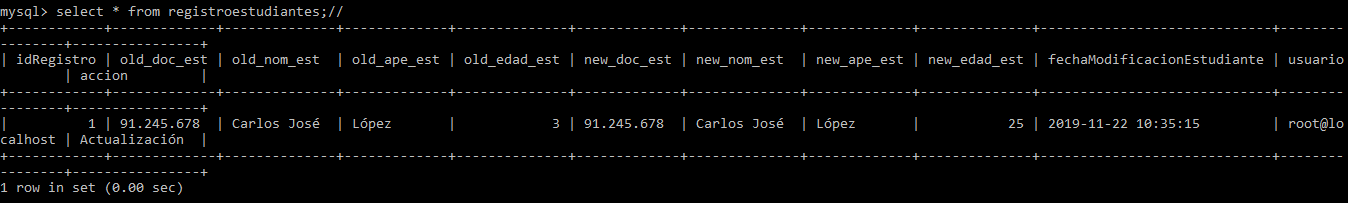
(old.doc\_est, old.nom\_est, old.ape\_est, old.edad\_est, new.doc\_est, new.nom\_est, new.ape\_est, new.edad\_est, now(), current\_user(), 'Actualización');



UPDATE estudiante SET edad\_est = 25 WHERE doc\_est = '91.245.678';//



SELECT \* FROM registroestudiantes;



Trigger Registro Estudiantes Borrados:

CREATE TABLE registroEstudiantesBorrados(

idRegistro int primary key auto\_increment,

doc\_est\_borrado varchar(11),

nom\_est\_borrado varchar(30),

ape\_est\_borrado varchar(30),

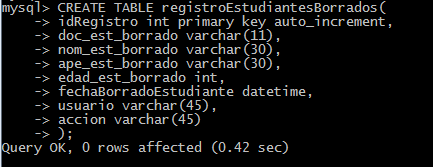
edad\_est\_borrado int,

fechaBorradoEstudiante datetime,

usuario varchar(45),

accion varchar(45)

);

CREATE TRIGGER estudiantesBorrados BEFORE DELETE ON estudiante

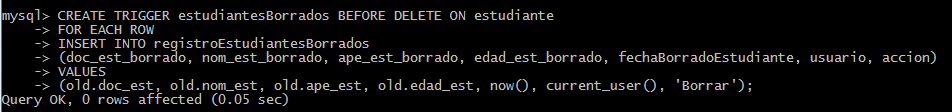
FOR EACH ROW

INSERT INTO registroEstudiantesBorrados

(doc\_est\_borrado, nom\_est\_borrado, ape\_est\_borrado, edad\_est\_borrado, fechaBorradoEstudiante, usuario, accion)

VALUES

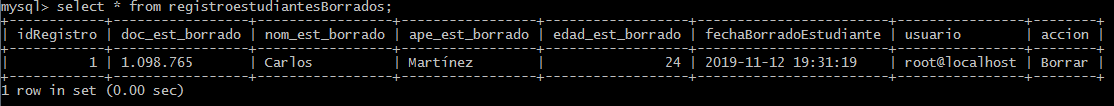
(old.doc\_est, old.nom\_est, old.ape\_est, old.edad\_est, now(), current\_user(), ‘Borrar’);



delete from estudiante where ape\_est = ‘Martínez’;



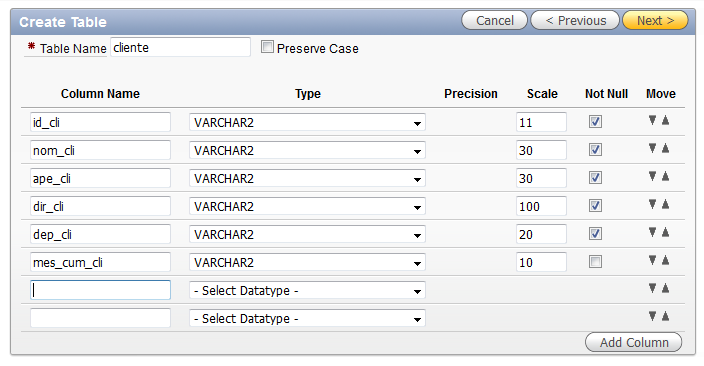
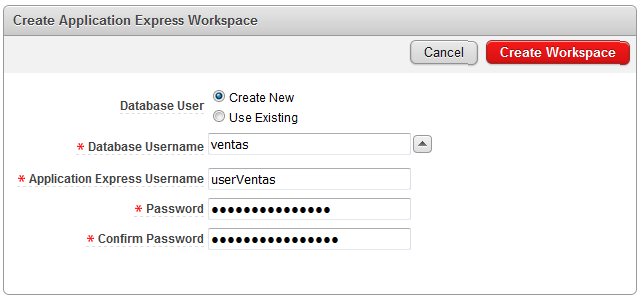
select \* from registroestudiantesborrados;

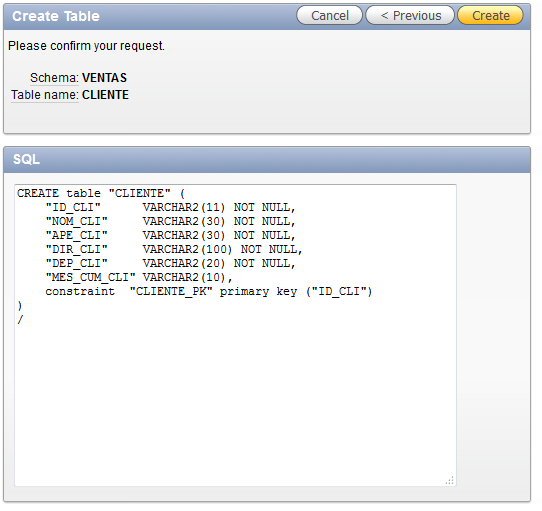
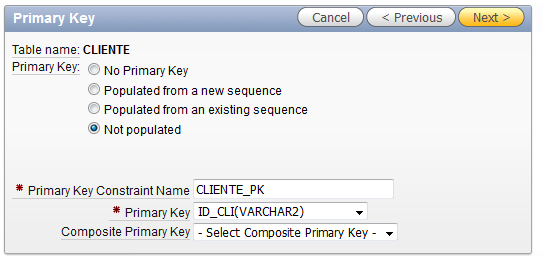


----------------------------------------------------------------------------------------------------------------

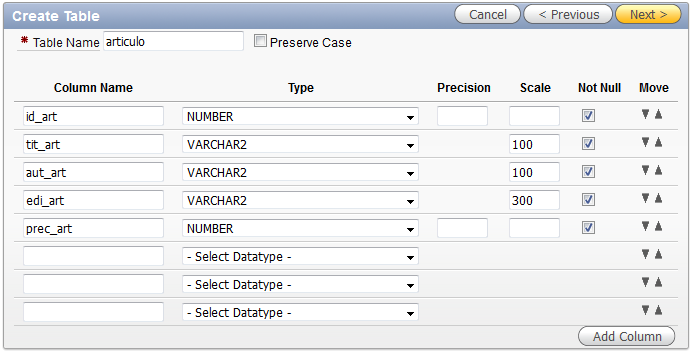
**ORACLE:**

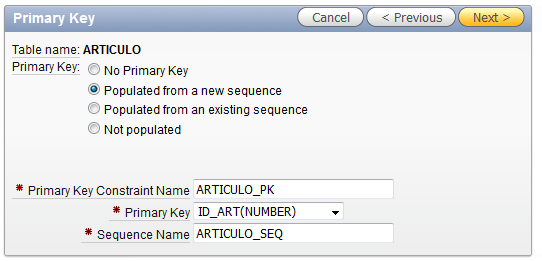
----------------------------------------------------------------------------------------------------------------

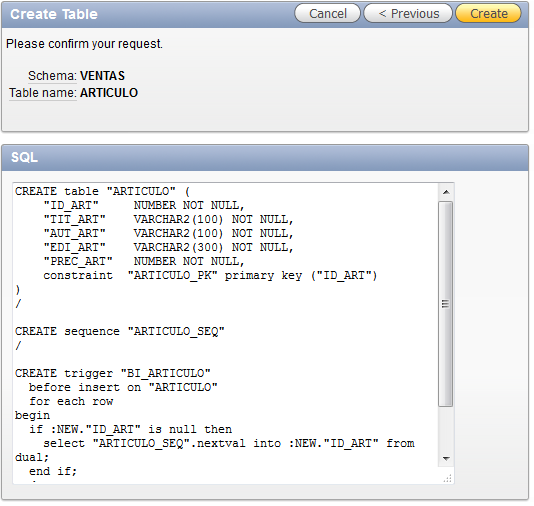


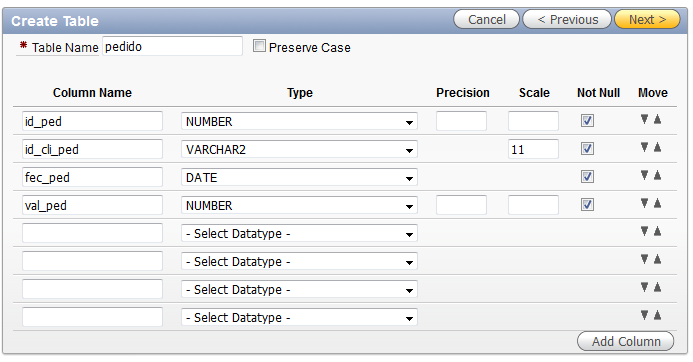


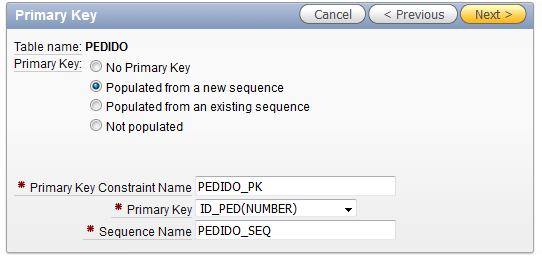
----------------------------------------------------------------------------------------------------------------

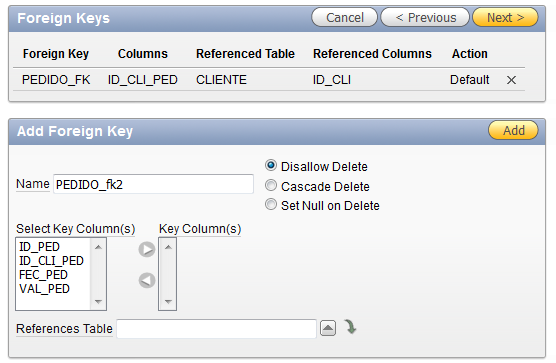


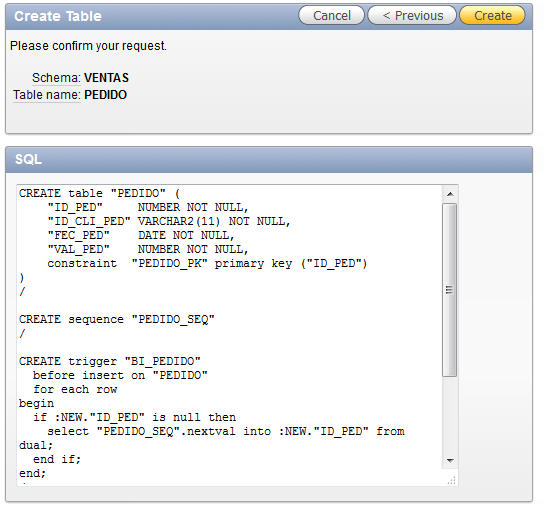


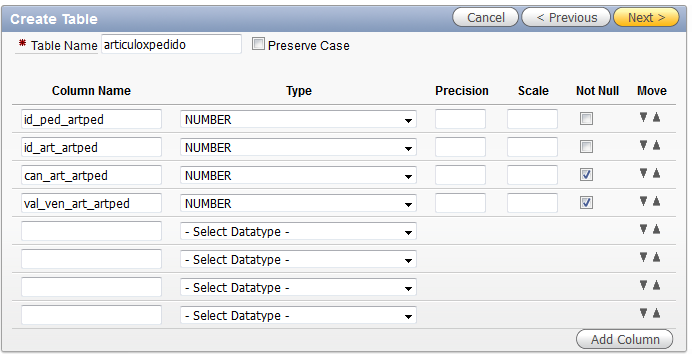


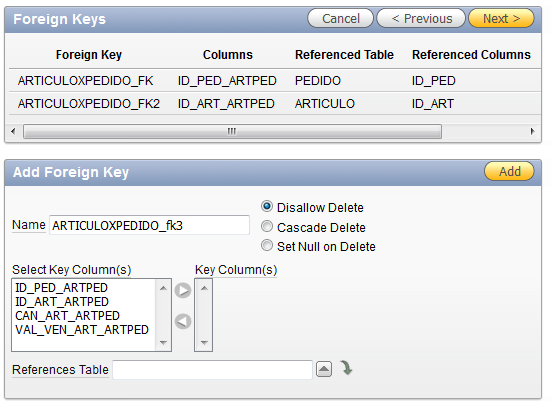
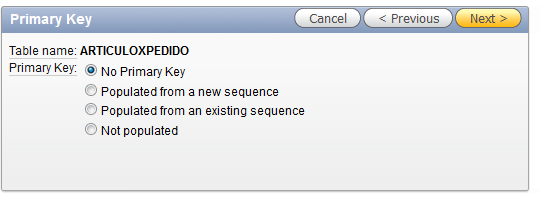
----------------------------------------------------------------------------------------------------------------

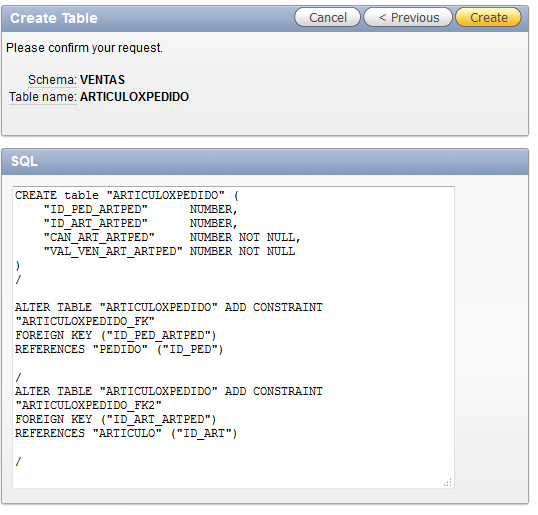
****





--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------





----------------------------------------------------------------------------------------------------------------

**Ejercicio 1:**

**Procedimientos Almacenados Para Insertar Datos.**

***Insertar Datos Clientes.***

CREATE OR REPLACE PROCEDURE INSERTCLIENTES (ID IN VARCHAR2, NOMBRE IN VARCHAR2, APELLIDO IN VARCHAR2,DIRECCION IN VARCHAR2, DEPARTAMENTO IN VARCHAR2, CUMPLEAÑOS IN VARCHAR2)

IS

BEGIN

INSERT INTO CLIENTE(ID\_CLI, NOM\_CLI, APE\_CLI, DIR\_CLI, DEP\_CLI, MES\_CUM\_CLI)

VALUES

(ID, NOMBRE, APELLIDO, DIRECCION, DEPARTAMENTO, CUMPLEAÑOS);

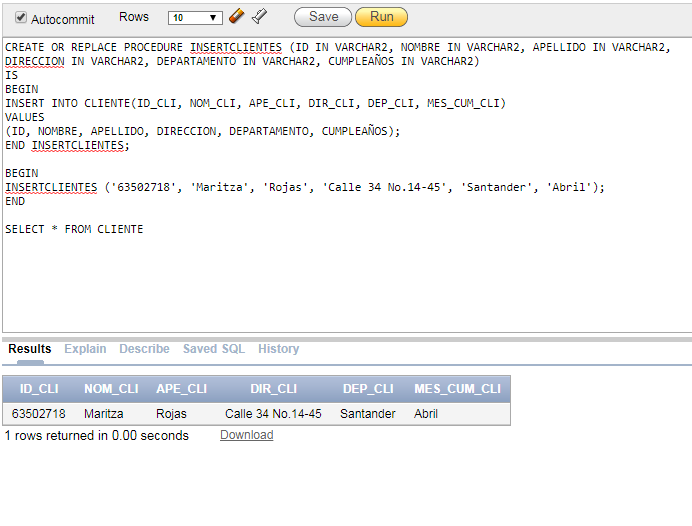
END INSERTCLIENTES;

BEGIN

INSERTCLIENTES ('63502718', 'Maritza', 'Rojas', 'Calle 34 No.14-45', 'Santander', 'Abril');

END

SELECT \* FROM CLIENTE

****----------------------------------------------------------------------------------------------------------------

***Insertar Datos Artículos.***

CREATE OR REPLACE PROCEDURE INSERTARTICULOS (ID IN NUMBER, TITULO IN VARCHAR2, AUTOR IN VARCHAR2,

EDITOR IN VARCHAR2, PRECIO IN NUMBER)

IS

BEGIN

INSERT INTO ARTICULO(ID\_ART, TIT\_ART, AUT\_ART, EDI\_ART, PREC\_ART)

VALUES

(ID, TITULO, AUTOR, EDITOR, PRECIO);

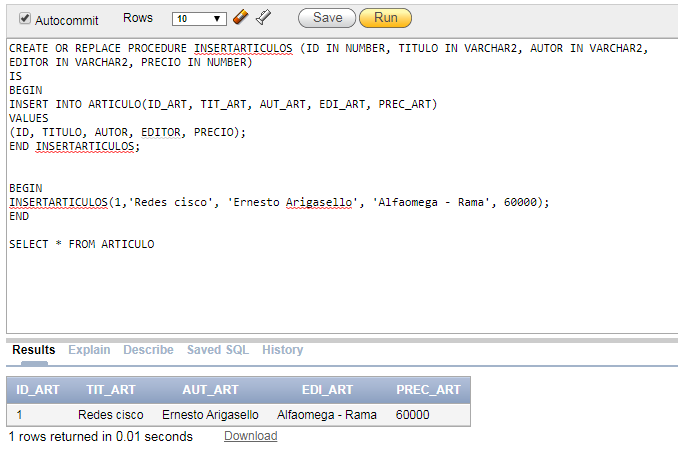
END INSERTARTICULOS;

BEGIN

INSERTARTICULOS(1,'Redes cisco', 'Ernesto Arigasello', 'Alfaomega - Rama', 60000);

END

SELECT \* FROM ARTICULO

----------------------------------------------------------------------------------------------------------------

***Insert Datos Pedido.***

CREATE OR REPLACE PROCEDURE INSERTPEDIDO(ID IN NUMBER, IDCLIENTE IN VARCHAR2, FECHA IN DATE, VALOR IN NUMBER)

IS

BEGIN

INSERT INTO PEDIDO(ID\_PED, ID\_CLI\_PED, FEC\_PED, VAL\_PED)

VALUES

(ID, IDCLIENTE, FECHA, VALOR);

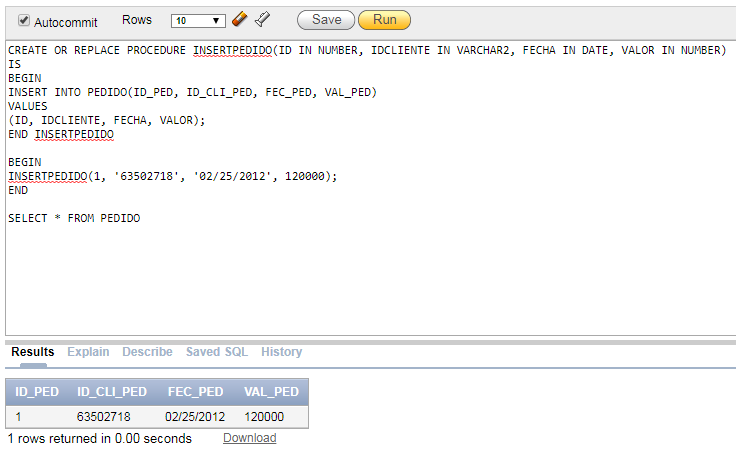
END INSERTPEDIDO;

BEGIN

INSERTPEDIDO(1, '63502718', '02/25/2012', 120000);

END

SELECT \* FROM PEDIDO



----------------------------------------------------------------------------------------------------------------

***Insert Datos Articuloxpedido.***

CREATE OR REPLACE PROCEDURE INSERTARTICULOXPEDIDO (IDPEDIDO IN NUMBER, IDARTICULO IN NUMBER, CANTIDAD IN NUMBER,

VALOR IN NUMBER)

IS

BEGIN

INSERT INTO ARTICULOXPEDIDO (ID\_PED\_ARTPED, ID\_ART\_ARTPED, CAN\_ART\_ARTPED, VAL\_VEN\_ART\_ARTPED)

VALUES

(IDPEDIDO, IDARTICULO, CANTIDAD, VALOR);

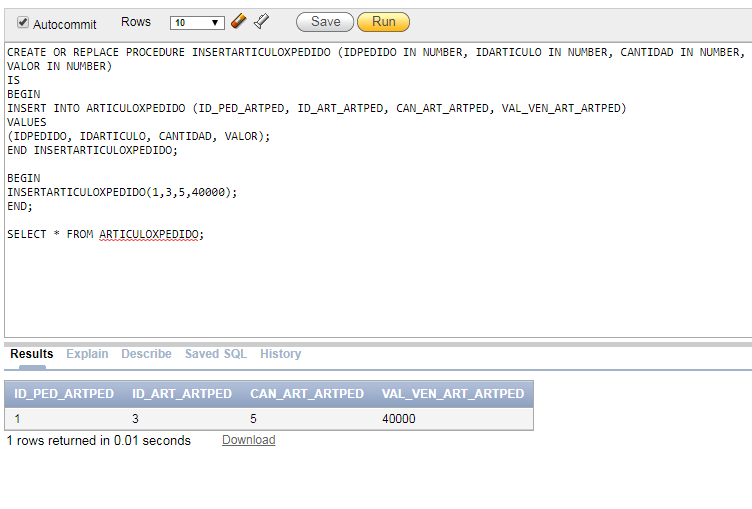
END INSERTARTICULOXPEDIDO;

BEGIN

INSERTARTICULOXPEDIDO(1,3,5,40000);

END;

SELECT \* FROM ARTICULOXPEDIDO;

----------------------------------------------------------------------------------------------------------------

**Ejercicio 2 Procedimientos Almacenados:**

Visualizar el nombre, apellido y dirección de todos aquellos clientes que hayan realizado un pedido el día 25/02/2012.

CREATE OR REPLACE PROCEDURE CLIENTESXFECHA(fec DATE)

IS

CURSOR DATOS IS

SELECT CLIENTE.NOM\_CLI, CLIENTE.DIR\_CLI

FROM CLIENTE INNER JOIN PEDIDO

ON CLIENTE.ID\_CLI = PEDIDO.ID\_CLI\_PED

WHERE PEDIDO.FEC\_PED = fec;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('NOMBRE DIRECCIÓN');

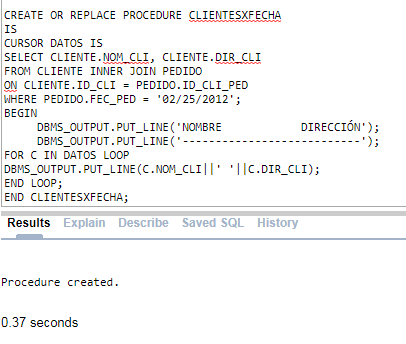
DBMS\_OUTPUT.PUT\_LINE('---------------------------');

FOR C IN DATOS LOOP

DBMS\_OUTPUT.PUT\_LINE(C.NOM\_CLI||' '||C.DIR\_CLI);

END LOOP;

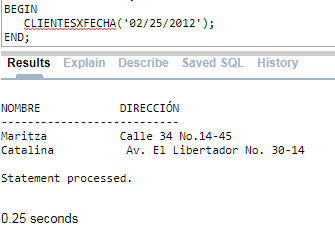
END CLIENTESXFECHA;



BEGIN

CLIENTESXFECHA(‘02/25/2012’);

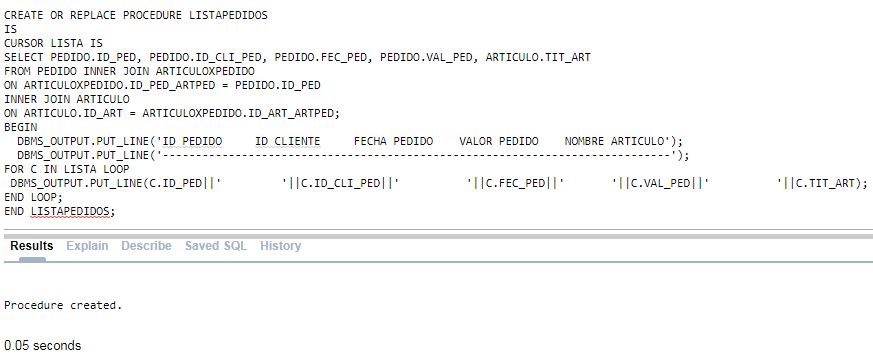
END;



----------------------------------------------------------------------------------------------------------------

Listar todos los pedidos realizados incluyendo el nombre del artículo:

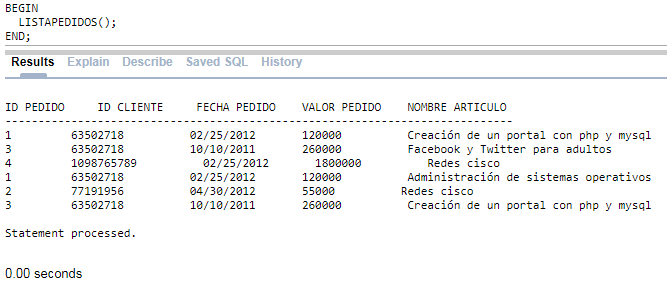
CREATE OR REPLACE PROCEDURE LISTAPEDIDOS IS CURSOR LISTA IS SELECT PEDIDO.ID\_PED, PEDIDO.ID\_CLI\_PED, PEDIDO.FEC\_PED, PEDIDO.VAL\_PED, ARTICULO.TIT\_ART FROM PEDIDO INNER JOIN ARTICULOXPEDIDO ON ARTICULOXPEDIDO.ID\_PED\_ARTPED = PEDIDO.ID\_PED INNER JOIN ARTICULO ON ARTICULO.ID\_ART = ARTICULOXPEDIDO.ID\_ART\_ARTPED; BEGIN DBMS\_OUTPUT.PUT\_LINE('ID PEDIDO ID CLIENTE FECHA PEDIDO VALOR PEDIDO NOMBRE ARTICULO'); DBMS\_OUTPUT.PUT\_LINE('-----------------------------------------------------------------------------'); FOR C IN LISTA LOOP DBMS\_OUTPUT.PUT\_LINE(C.ID\_PED||' '||C.ID\_CLI\_PED||' '||C.FEC\_PED||' '||C.VAL\_PED||' '||C.TIT\_ART); END LOOP; END LISTAPEDIDOS;



BEGIN

LISTAPEDIDOS();

END;



----------------------------------------------------------------------------------------------------------------

Mostrar los pedidos con los respectivos artículos (código, nombre, valor y cantidad pedida).

CREATE OR REPLACE PROCEDURE PEDIDOSXARTICULOS

IS

CURSOR DATOS IS

SELECT PEDIDO.ID\_PED, PEDIDO.ID\_CLI\_PED, PEDIDO.FEC\_PED, PEDIDO.VAL\_PED, ARTICULO.ID\_ART,

ARTICULO.TIT\_ART, ARTICULO.PREC\_ART, ARTICULOXPEDIDO.CAN\_ART\_ARTPED

FROM PEDIDO INNER JOIN ARTICULOXPEDIDO

ON PEDIDO.ID\_PED = ARTICULOXPEDIDO.ID\_PED\_ARTPED

INNER JOIN ARTICULO

ON ARTICULO.ID\_ART= ARTICULOXPEDIDO.ID\_ART\_ARTPED;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('ID PEDIDO ID CLIENTE FECHA PEDIDO VALOR PEDIDO ID ARTICULO NOMBRE ARTICULO PRECIO ARTICULO CANTIDAD ARTICULO');

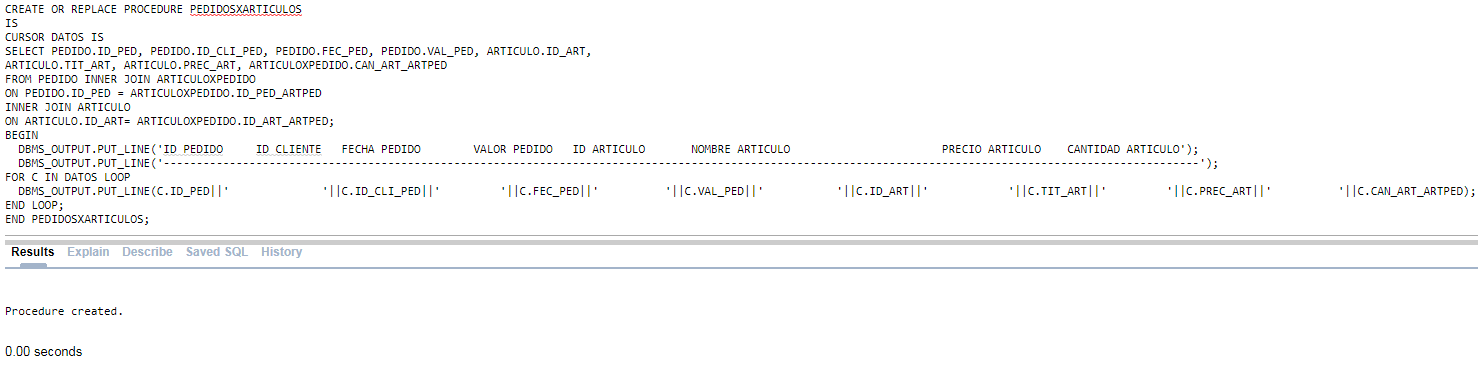
DBMS\_OUTPUT.PUT\_LINE('-------------------------------------------------------------------------------------------------------------------------------------------------------------');

FOR C IN DATOS LOOP

DBMS\_OUTPUT.PUT\_LINE(C.ID\_PED||' '||C.ID\_CLI\_PED||' '||C.FEC\_PED||' '||C.VAL\_PED||' '||C.ID\_ART||' '||C.TIT\_ART||' '||C.PREC\_ART||' '||C.CAN\_ART\_ARTPED);

END LOOP;

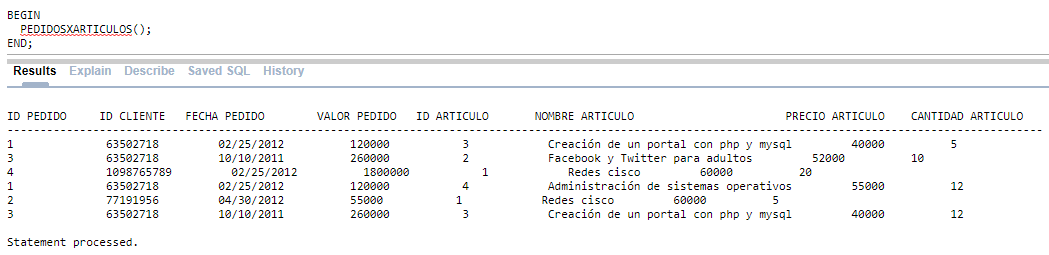
END PEDIDOSXARTICULOS;



BEGIN

PEDIDOSXARTICULOS();

END;



----------------------------------------------------------------------------------------------------------------

Visualizar todos los clientes organizados por apellido

CREATE OR REPLACE PROCEDURE CLIENTESORDENADOS

IS

CURSOR LISTA IS

SELECT ID\_CLI, NOM\_CLI, APE\_CLI, DIR\_CLI, DEP\_CLI, MES\_CUM\_CLI

FROM CLIENTE

ORDER BY APE\_CLI, NOM\_CLI;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('ID NOMBRE APELLIDO DIRECCIÓN DEPARTAMENTO MES CUMPLEAÑOS');

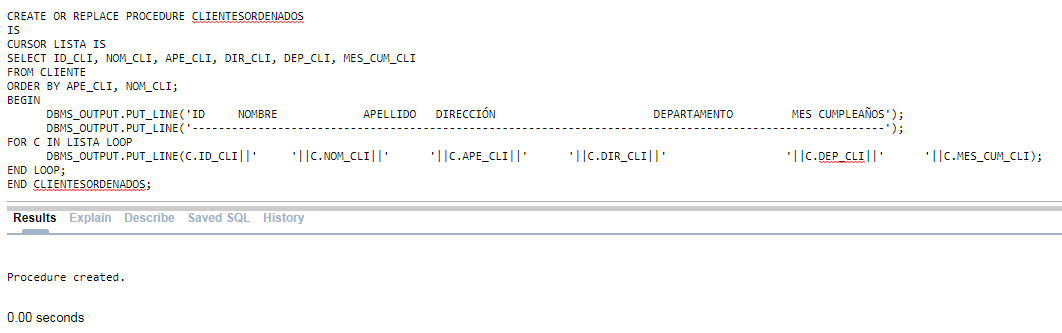
DBMS\_OUTPUT.PUT\_LINE('---------------------------------------------------------------------------------------------------------');

FOR C IN LISTA LOOP

DBMS\_OUTPUT.PUT\_LINE(C.ID\_CLI||' '||C.NOM\_CLI||' '||C.APE\_CLI||' '||C.DIR\_CLI||' '||C.DEP\_CLI||' '||C.MES\_CUM\_CLI);

END LOOP;

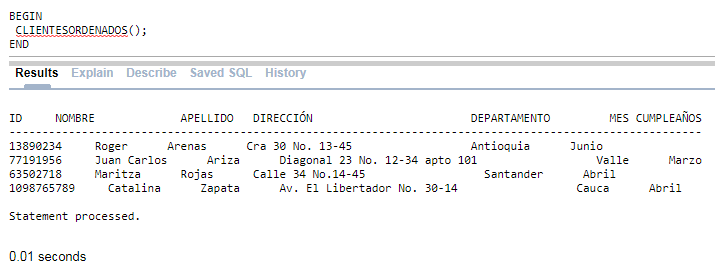
END CLIENTESORDENADOS;



BEGIN

CLIENTESORDENADOS();

END



----------------------------------------------------------------------------------------------------------------

**Ejercicio 6:**

**Funciones.**

Visualizar el nombre del cliente, la fecha y el valor del pedido más costoso:

CREATE OR REPLACE PROCEDURE PEDIDOMAYOR

IS

CURSOR DATOS IS

SELECT CLIENTE.NOM\_CLI, CLIENTE.APE\_CLI, PEDIDO.VAL\_PED

FROM CLIENTE INNER JOIN PEDIDO

ON PEDIDO.ID\_CLI\_PED = CLIENTE.ID\_CLI

WHERE PEDIDO.VAL\_PED = (SELECT MAX(VAL\_PED) FROM PEDIDO);

BEGIN

DBMS\_OUTPUT.PUT\_LINE('NOMBRE APELLIDO VALOR PEDIDO');

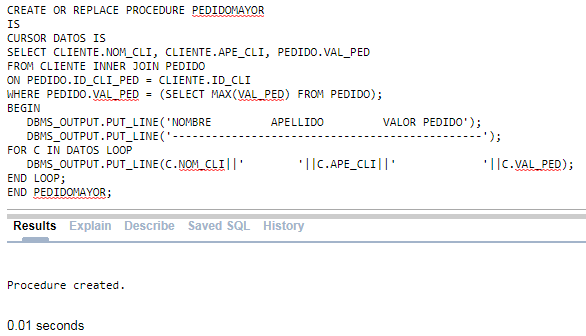
DBMS\_OUTPUT.PUT\_LINE('-----------------------------------------------');

FOR C IN DATOS LOOP

DBMS\_OUTPUT.PUT\_LINE(C.NOM\_CLI||' '||C.APE\_CLI||' '||C.VAL\_PED);

END LOOP;

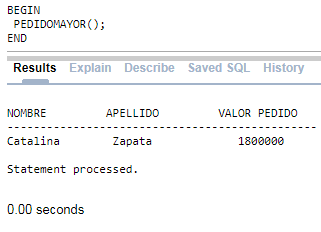
END PEDIDOMAYOR;



BEGIN

PEDIDOMAYOR();

END



CREATE OR REPLACE FUNCTION pedidosuperior

RETURN VARCHAR2

IS

informe varchar2(150);

BEGIN

select (NOM\_CLI||' '||FEC\_PED||' '||VAL\_PED) into informe

FROM PEDIDO

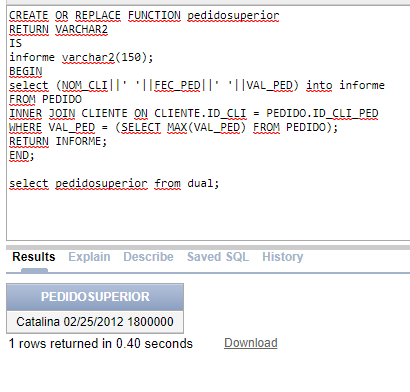
INNER JOIN CLIENTE ON CLIENTE.ID\_CLI = PEDIDO.ID\_CLI\_PED

WHERE VAL\_PED = (SELECT MAX(VAL\_PED) FROM PEDIDO);

RETURN INFORME;

END;

select pedidosuperior from dual;



----------------------------------------------------------------------------------------------------------------

Mostrar cuantos articulos se tienen en cada editorial:

CREATE OR REPLACE PROCEDURE ARTICULOSXEDITORIAL

IS

CURSOR CANTIDAD IS

SELECT EDI\_ART, COUNT(EDI\_ART) AS "NUMERO"

FROM ARTICULO

GROUP BY EDI\_ART;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('EDITORIAL CANTIDAD ARTICULOS');

DBMS\_OUTPUT.PUT\_LINE('------------------------------');

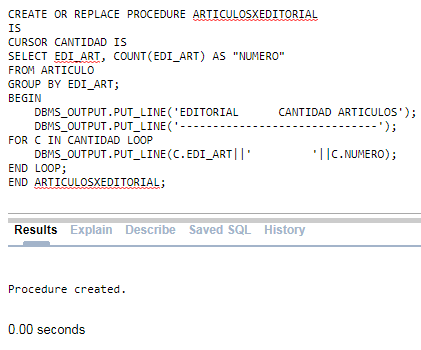
FOR C IN CANTIDAD LOOP

DBMS\_OUTPUT.PUT\_LINE(C.EDI\_ART||' '||C.NUMERO);

END LOOP;

END ARTICULOSXEDITORIAL;

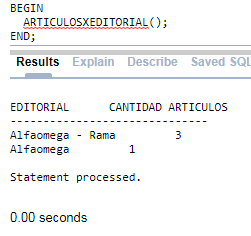
---------------------------



BEGIN

ARTICULOSXEDITORIAL();

END;



CREATE OR REPLACE FUNCTION editoriales

RETURN NUMBER

IS cantidad NUMBER;

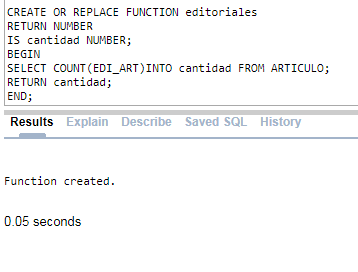
BEGIN

SELECT COUNT(EDI\_ART) INTO cantidad

FROM ARTICULO;

RETURN cantidad;

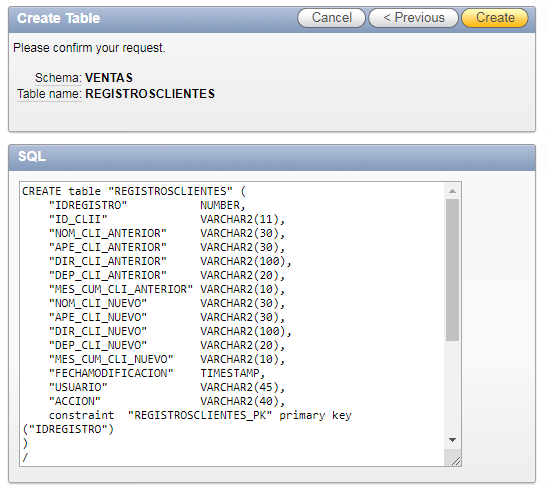
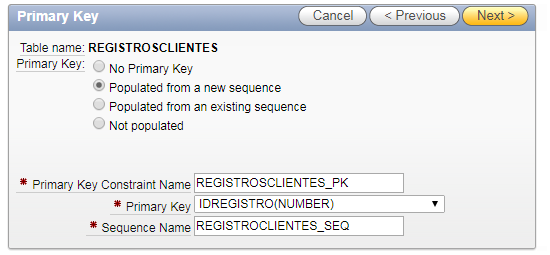
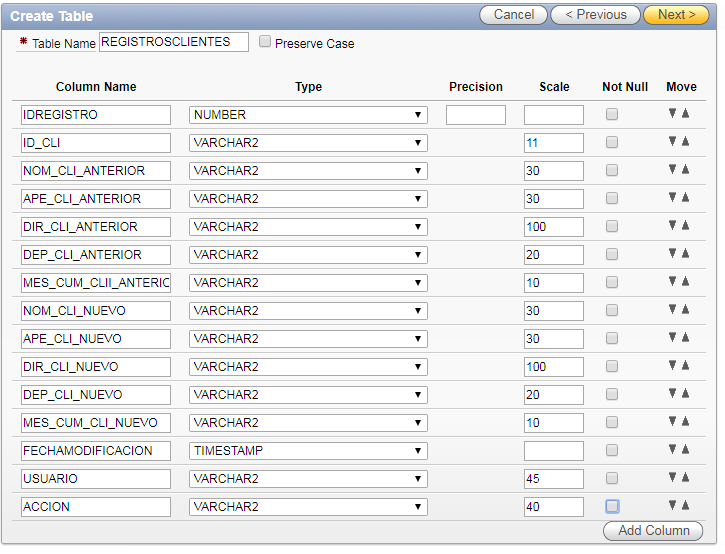
END;



----------------------------------------------------------------------------------------------------------------

**Ejercicio 8:**

**Triggers.**

Trigger Actualización Cliente: 

PL/SQL:

CREATE OR REPLACE TRIGGER "REGISTROSCLIENTES"

BEFORE

update on "CLIENTE"

for each row

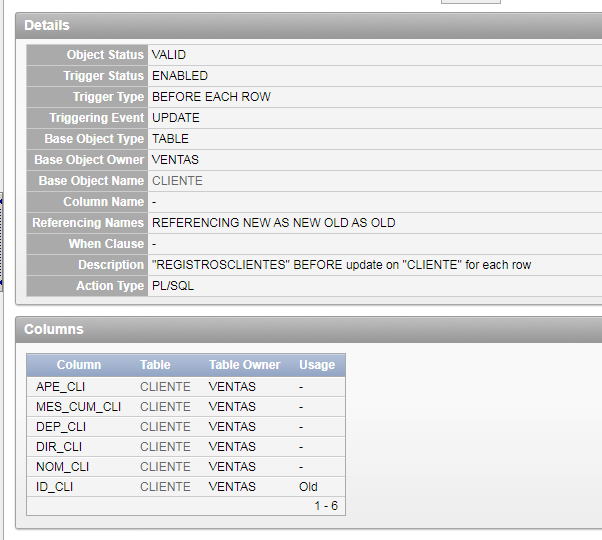
begin

INSERT INTO REGISTROSCLIENTES (ID\_CLI, NOM\_CLI\_ANTERIOR, APE\_CLI\_ANTERIOR, DIR\_CLI\_ANTERIOR, DEP\_CLI\_ANTERIOR, MES\_CUM\_CLI\_ANTERIOR, NOM\_CLI\_NUEVO, APE\_CLI\_NUEVO, DIR\_CLI\_NUEVO, DEP\_CLI\_NUEVO, MES\_CUM\_CLI\_NUEVO, FECHAMODIFICACION, USUARIO, ACCION)

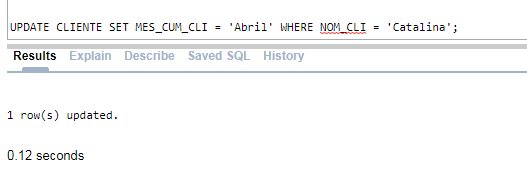
VALUES

(:OLD.ID\_CLI, :OLD.NOM\_CLI, :OLD.APE\_CLI, :OLD.DIR\_CLI, :OLD.DEP\_CLI, :OLD.MES\_CUM\_CLI, :NEW.NOM\_CLI, :NEW.APE\_CLI, :NEW.DIR\_CLI, :NEW.DEP\_CLI, :NEW.MES\_CUM\_CLI, SYSDATE, USER, 'Actualizar');

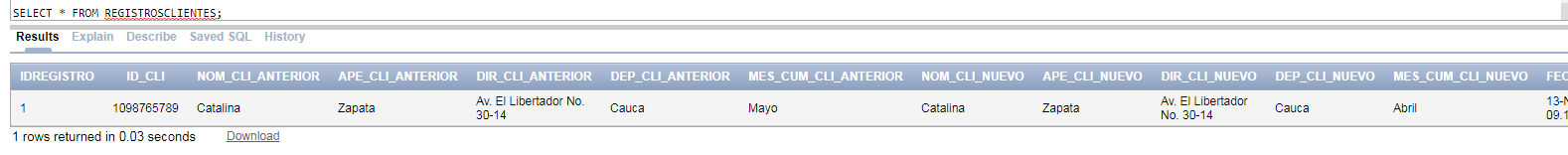
end;



UPDATE CLIENTE SET MES\_CUM\_CLI = 'Abril' WHERE NOM\_CLI = 'Catalina';

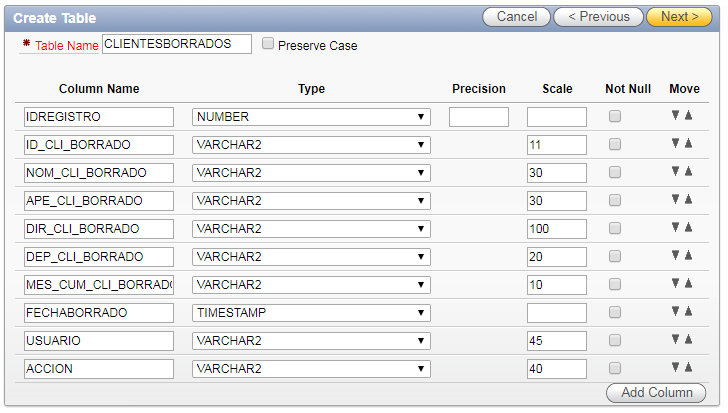


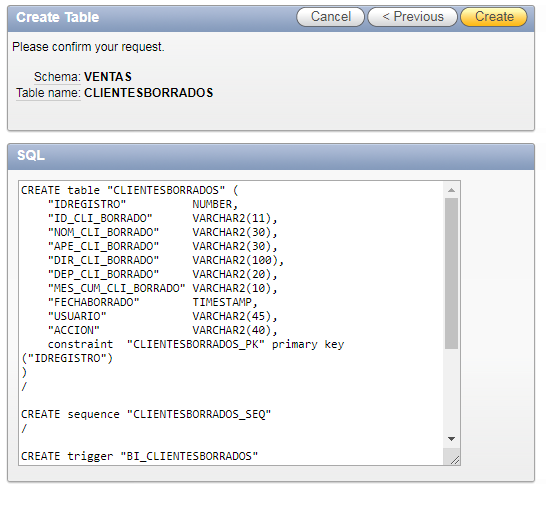
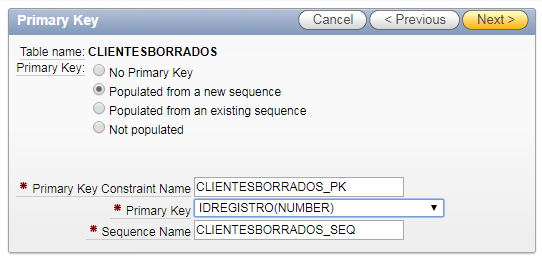
SELECT \* FROM REGISTROSCLIENTES;



----------------------------------------------------------------------------------------------------------------

Registros Clientes Borrados:





PL/SQL:

CREATE OR REPLACE TRIGGER "CLIENTESELIMINADOS"

BEFORE

delete on "CLIENTE"

for each row

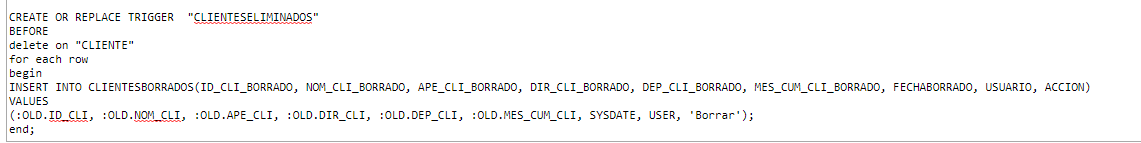
begin

INSERT INTO CLIENTESBORRADOS(ID\_CLI\_BORRADO, NOM\_CLI\_BORRADO, APE\_CLI\_BORRADO, DIR\_CLI\_BORRADO, DEP\_CLI\_BORRADO, MES\_CUM\_CLI\_BORRADO, FECHABORRADO, USUARIO, ACCION)

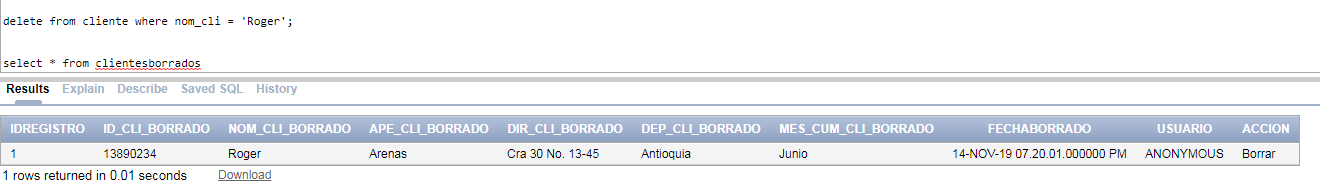
VALUES

(:OLD.ID\_CLI, :OLD.NOM\_CLI, :OLD.APE\_CLI, :OLD.DIR\_CLI, :OLD.DEP\_CLI, :OLD.MES\_CUM\_CLI, SYSDATE, USER, 'Borrar');

end;

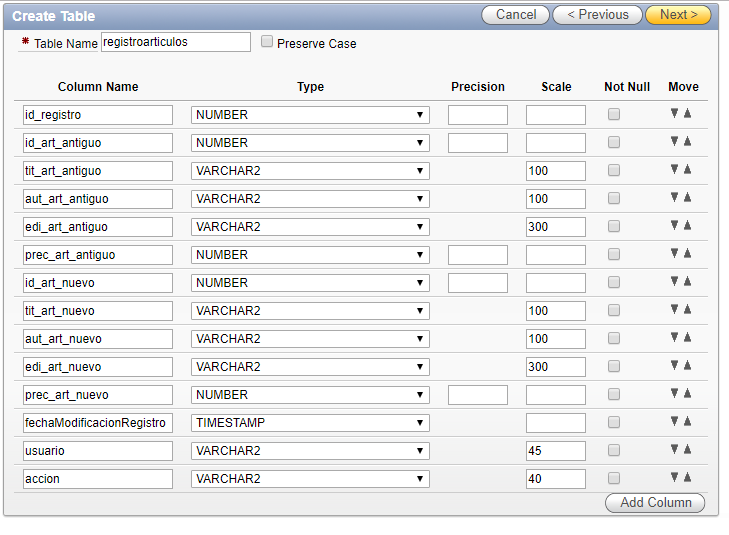


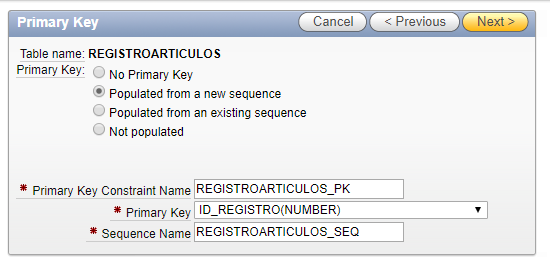
select \* from clientesborrados

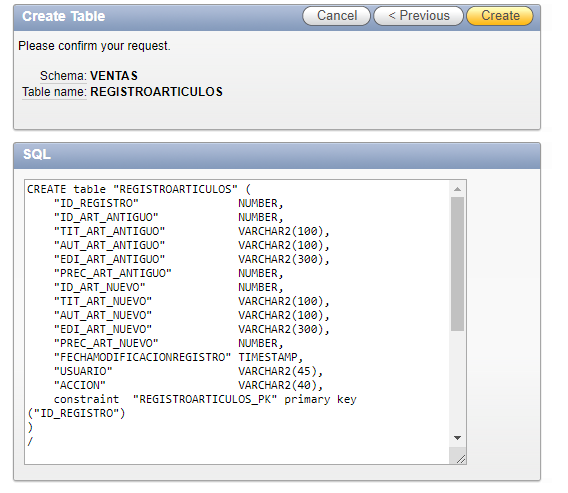


----------------------------------------------------------------------------------------------------------------

Registros Artículos Actualizados:







PL/SQL:

CREATE OR REPLACE TRIGGER "REGISTROARTICULOS"

BEFORE

update on "ARTICULO"

for each row

begin

INSERT INTO REGISTROARTICULOS (ID\_ART\_ANTIGUO, TIT\_ART\_ANTIGUO, AUT\_ART\_ANTIGUO, EDI\_ART\_ANTIGUO,

PREC\_ART\_ANTIGUO, ID\_ART\_NUEVO, TIT\_ART\_NUEVO, AUT\_ART\_NUEVO, EDI\_ART\_NUEVO, PREC\_ART\_NUEVO,

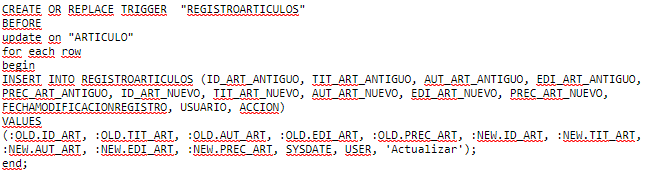
FECHAMODIFICACIONREGISTRO, USUARIO, ACCION)

VALUES

(:OLD.ID\_ART, :OLD.TIT\_ART, :OLD.AUT\_ART, :OLD.EDI\_ART, :OLD.PREC\_ART, :NEW.ID\_ART, :NEW.TIT\_ART,

:NEW.AUT\_ART, :NEW.EDI\_ART, :NEW.PREC\_ART, SYSDATE, USER, 'Actualizar');

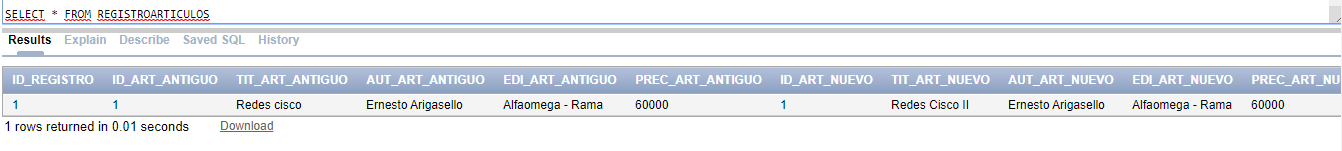
end;



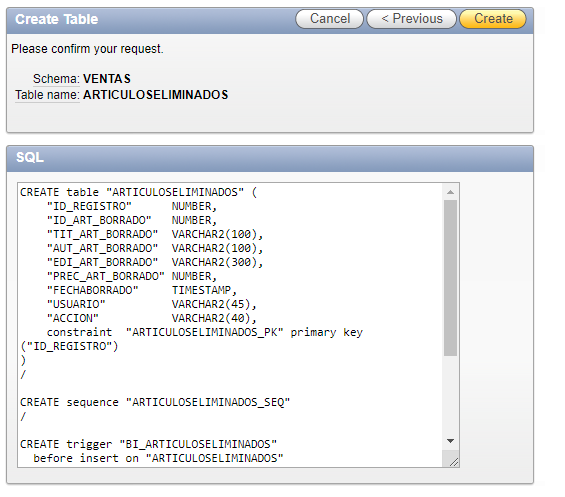
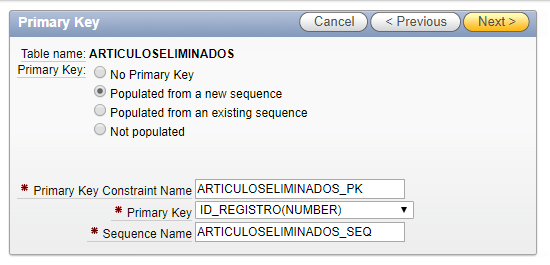
UPDATE ARTICULO SET TIT\_ART ‘Resdes Cisco II’ WHERE ID\_ART = 1;



SELECT \* FROM REGISTROARTICULOS;



----------------------------------------------------------------------------------------------------------------

Registro Artículos Eliminados:

PL:SQL:

CREATE OR REPLACE TRIGGER "ARTICULOSELIMINADOS"

BEFORE

delete on "ARTICULO"

for each row

begin

INSERT INTO ARTICULOSELIMINADOS(ID\_ART\_BORRADO, TIT\_ART\_BORRADO, AUT\_ART\_BORRADO, EDI\_ART\_BORRADO,

PREC\_ART\_BORRADO, FECHABORRADO, USUARIO, ACCION)

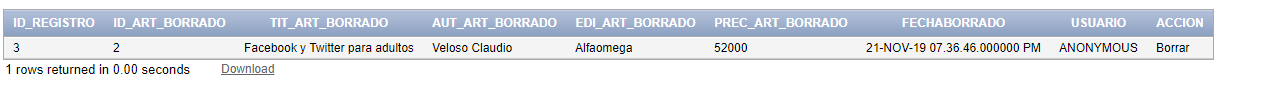
VALUES

(:OLD.ID\_ART, :OLD.TIT\_ART, :OLD.AUT\_ART, :OLD.EDI\_ART, :OLD.PREC\_ART, SYSDATE, USER, 'Borrar');

end;

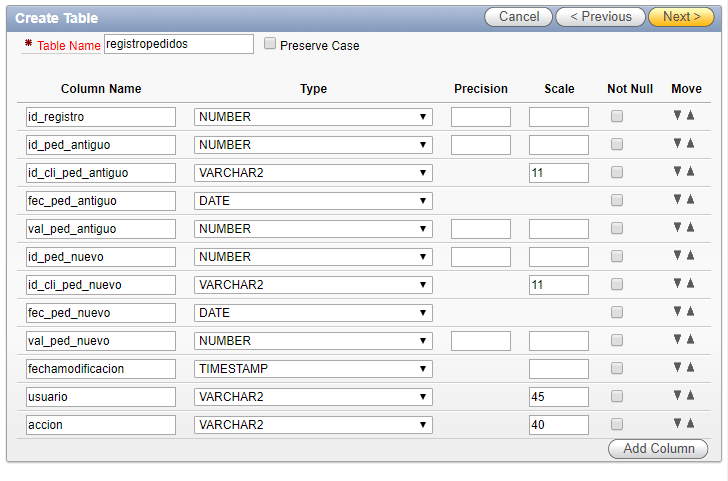
DELETE FROM ARTICULO WHERE ID\_ART = 2;

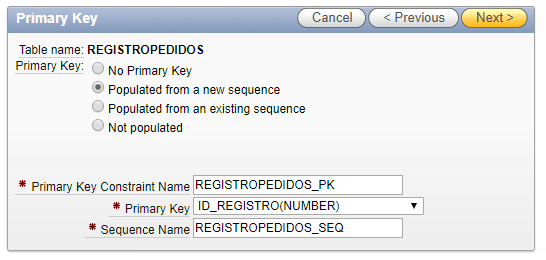
select \* from ARTICULOSELIMINADOS;

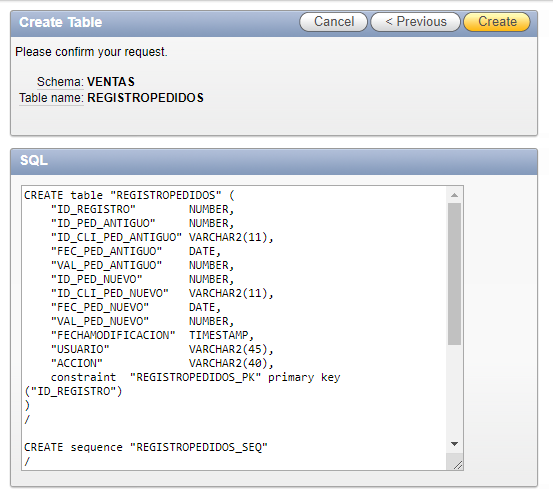


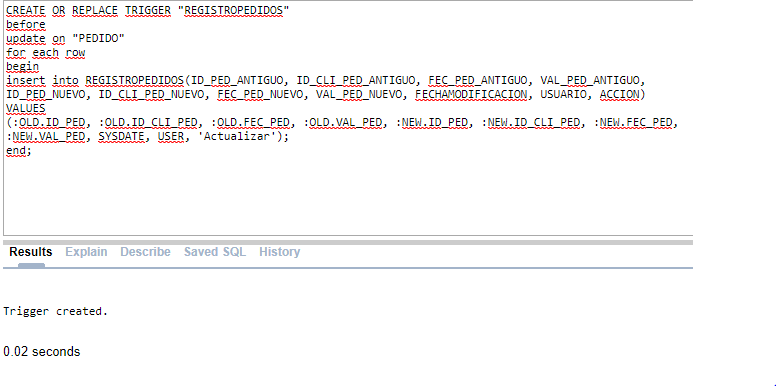
----------------------------------------------------------------------------------------------------------------

Registro Pedidos Actualizados:









PL/SQL:

CREATE OR REPLACE TRIGGER "REGISTROPEDIDOS"

before

update on "PEDIDO"

for each row

begin

insert into REGISTROPEDIDOS(ID\_PED\_ANTIGUO, ID\_CLI\_PED\_ANTIGUO, FEC\_PED\_ANTIGUO, VAL\_PED\_ANTIGUO,

ID\_PED\_NUEVO, ID\_CLI\_PED\_NUEVO, FEC\_PED\_NUEVO, VAL\_PED\_NUEVO, FECHAMODIFICACION, USUARIO, ACCION)

VALUES

(:OLD.ID\_PED, :OLD.ID\_CLI\_PED, :OLD.FEC\_PED, :OLD.VAL\_PED, :NEW.ID\_PED, :NEW.ID\_CLI\_PED, :NEW.FEC\_PED,

:NEW.VAL\_PED, SYSDATE, USER, 'Actualizar');

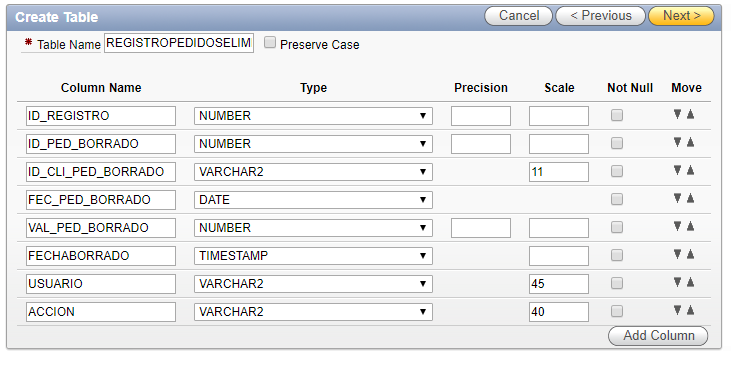
end;

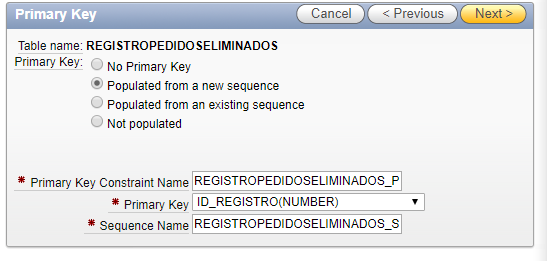
UPDATE PEDIDO SET VAL\_PED = 130000 WHERE ID\_PE D= 1;

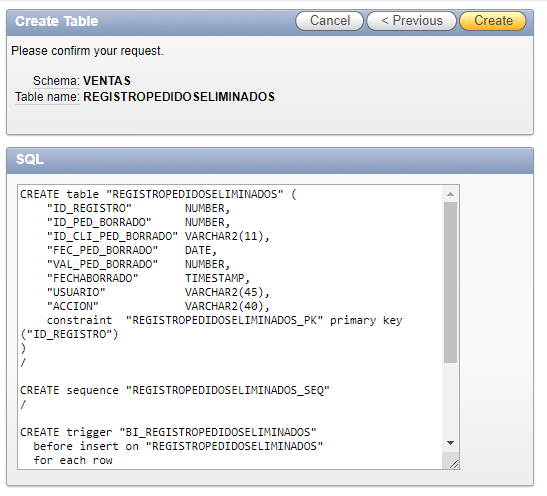
SELECT \* FROM REGISTROPEDIDOS;

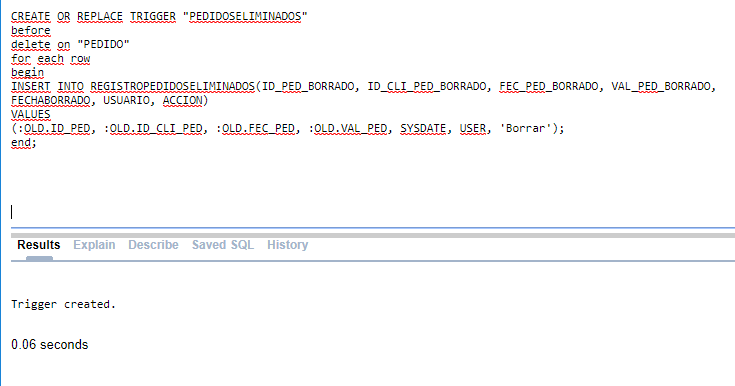
----------------------------------------------------------------------------------------------------------------

Registro Pedidos Eliminados:









CREATE OR REPLACE TRIGGER "PEDIDOSELIMINADOS"

before

delete on "PEDIDO"

for each row

begin

INSERT INTO REGISTROPEDIDOSELIMINADOS(ID\_PED\_BORRADO, ID\_CLI\_PED\_BORRADO, FEC\_PED\_BORRADO, VAL\_PED\_BORRADO,

FECHABORRADO, USUARIO, ACCION)

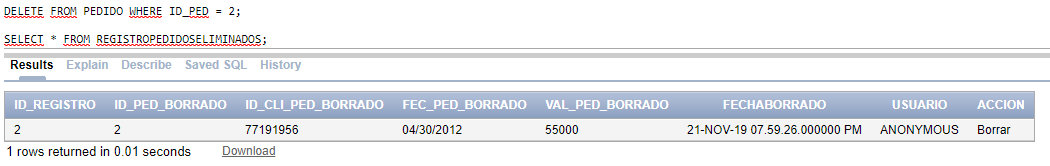
VALUES

(:OLD.ID\_PED, :OLD.ID\_CLI\_PED, :OLD.FEC\_PED, :OLD.VAL\_PED, SYSDATE, USER, 'Borrar');

end;

DELETE FROM PEDIDO WHERE ID\_PED = 2;

SELECT \* FROM REGISTROPEDIDOSELIMINADOS;



*Anexos.*

Se adjunta a este documento los dos respectivos Scripts de las bases de datos.