

DEFENSA HITO 4

Ana Cristina Calderón Ortega

TEORICA

1. Defina que es lenguaje procedural en MySQL.

Lenguajes procedurales o procedimentales: El usuario da órdenes para que se realicen las tareas pertinentes con el objetivo de recuperar los datos requeridos. Es la base del lenguaje de consulta SQL.

```
1 • use employees;
2
3 • select count(*)
4   into @numEmpleados
5   from dept_emp
6   where dept_no = 'd005';
7
8 • select @numEmpleados;
9
```

#	@numEmpleados
1	85707



2. Defina que es una FUNCTION en MySQL.

Las funciones son piezas de código que reciben datos de entrada, realizan operaciones con ellos y luego devuelven un resultado.

```
1 drop function if exists related_count;
2 create function related_count(parent int(11)) returns int(11)
3 begin
4   declare count int(11) default 0;
5   while parent!=0 and count<10 do
6     set count=count+1;
7     set parent=(select related from category where id=parent);
8   end while
9   return count
10 end
```


3.Cuál es la diferencia entre funciones y procedimientos almacenados.

Cuando llama al procedimiento almacenado, se debe especificar que es un parámetro externo. Una ventaja de los procedimientos almacenados es que puede obtener varios parámetros mientras que, en las funciones, solo se puede devolver una variable (función escalar) o una tabla (funciones con valores de tabla)

4. Cómo se ejecuta una función y un procedimiento almacenado.

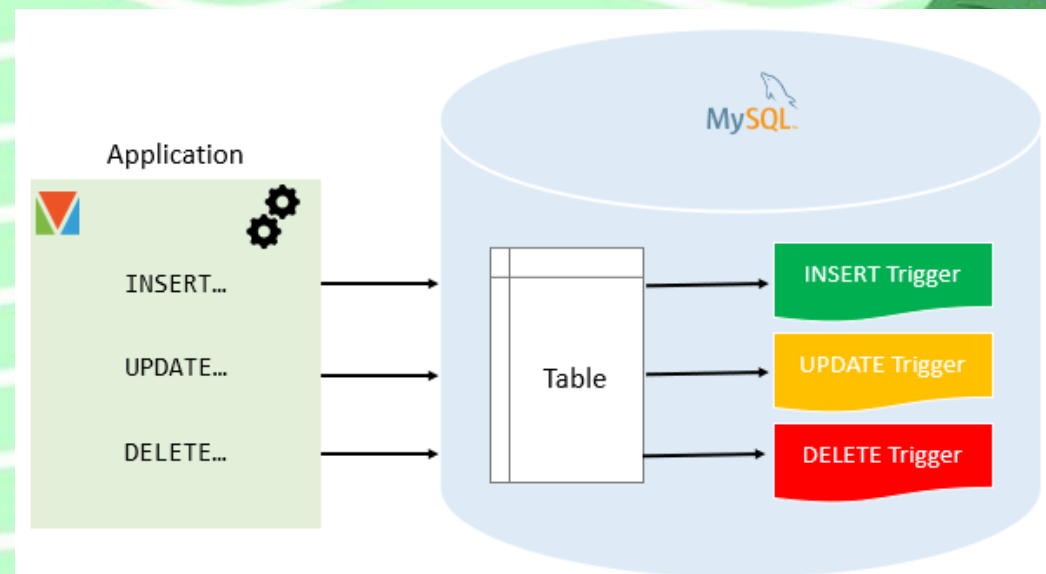
```
CREATE FUNCTION dbo.helloworldfunction()  
RETURNS varchar(20)  
AS  
BEGIN  
    RETURN 'Hello world'  
END
```

```
CREATE PROCEDURE contar_productos(IN gama VARCHAR(50), OUT total INT UNSIGNED)  
BEGIN  
    SELECT COUNT(*)  
    INTO total  
    FROM producto  
    WHERE producto.gama = gama;  
END
```

FUNCTION VERSUS PROCEDURE	
FUNCTION	PROCEDURE
Tipo de subprograma PL/SQL que siempre devuelve un valor	Tipo de subprograma PL/SQL que no devuelve un valor directamente
Se utiliza para hacer calculos y devolver un valor	Se utiliza para realizar una acción
Una función siempre devuelve un valor	Un procedimiento puede o no devolver un valor

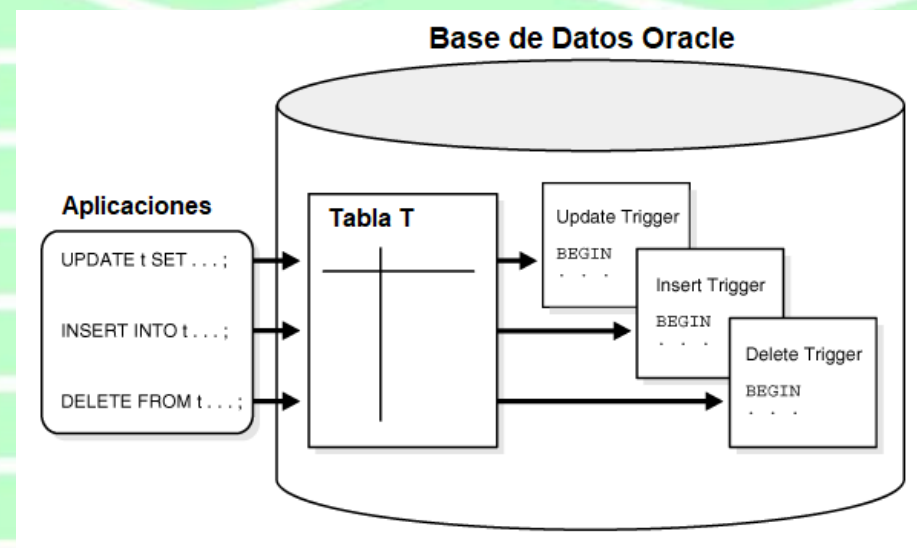
5. Defina que es una TRIGGER en MySQL.

Es un objeto que se crea con la sentencia CREATE TRIGGER y tiene que estar asociado a una tabla. Un *trigger* se activa cuando ocurre un evento de inserción, actualización o borrado, sobre la tabla a la que está asociado.



6. En un trigger que papel juega las variables OLD y NEW

Las columnas de la tabla asociada con el disparador pueden referenciarse empleando los alias **OLD** y **NEW**. **OLD.nombre_col** hace referencia a una columna de una fila existente, antes de ser actualizada o borrada. **NEW.nombre_col** hace referencia a una columna en una nueva fila a punto de ser insertada, o en una fila existente luego de que fue actualizada.

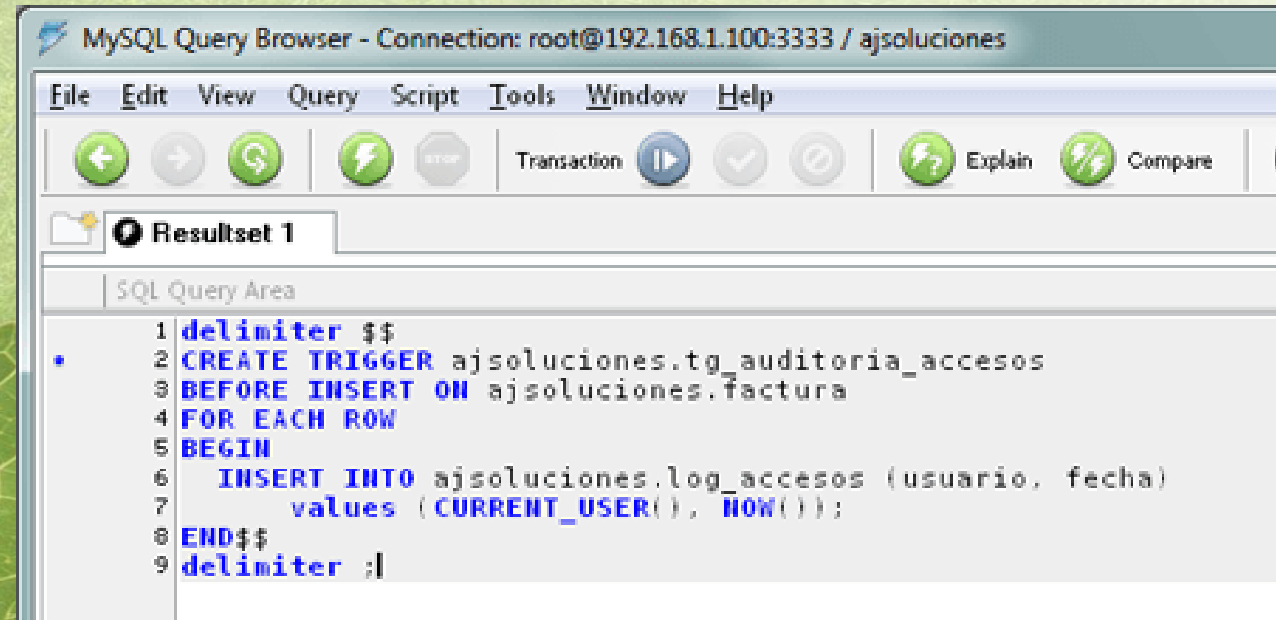


7. En un trigger que papel juega los conceptos(cláusulas) **BEFORE** o **AFTER**

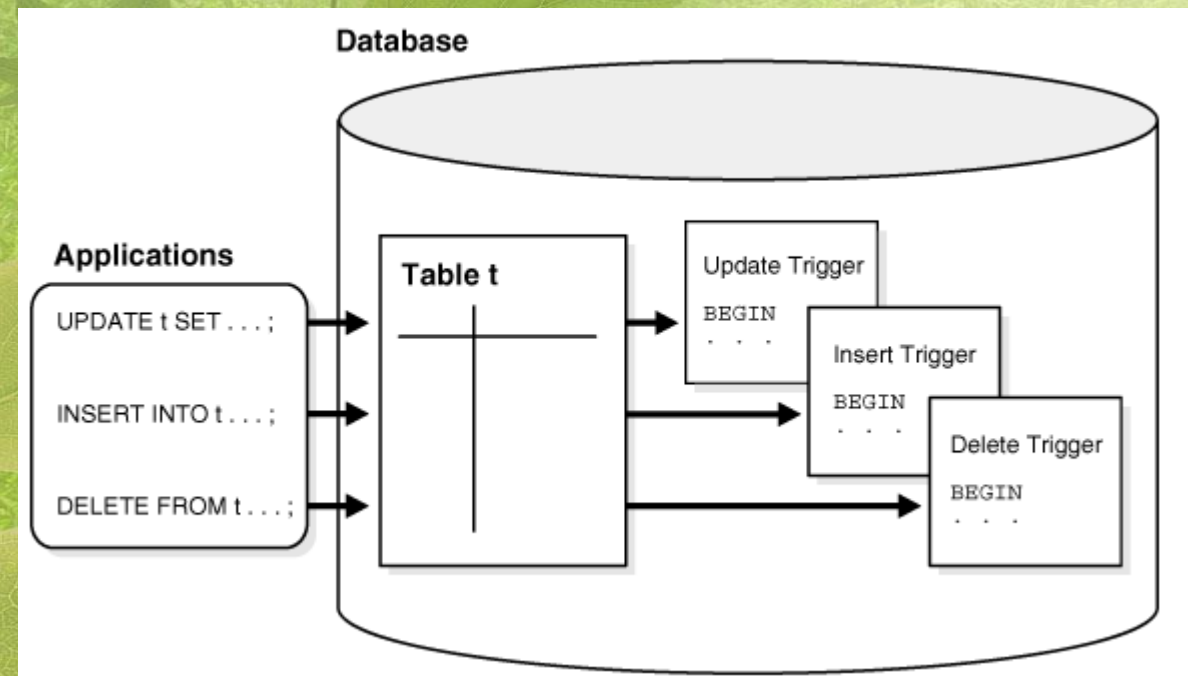
Puede ser **BEFORE** (antes) o **AFTER** (después), para indicar que el disparador se ejecute antes o después que la sentencia que lo activa.

8. A que se refiere cuando se habla de eventos en **TRIGGERS**

evento_disp indica la clase de sentencia que activa al disparador. Puede ser **INSERT**, **UPDATE**, o **DELETE**.



```
MySQL Query Browser - Connection: root@192.168.1.100:3333 / ajsoluciones
File Edit View Query Script Tools Window Help
Resultset 1
SQL Query Area
1 delimiter $$
2 CREATE TRIGGER ajsoluciones.tg_auditoria_accesos
3 BEFORE INSERT ON ajsoluciones.factura
4 FOR EACH ROW
5 BEGIN
6     INSERT INTO ajsoluciones.log_accesos (usuario, fecha)
7     values (CURRENT_USER(), NOW());
8 END$$
9 delimiter ;|
```



PRACTICA

9. Crear la siguiente Base de datos y sus registros.

```
CREATE DATABASE PROYECTO;  
USE PROYECTO;
```

```
CREATE TABLE DEPARTAMENTO(  
  ID_DEP INT AUTO_INCREMENT PRIMARY KEY NOT NULL,  
  NOMBRE VARCHAR(25)  
);
```

```
CREATE TABLE PROVINCIA(  
  ID_PROV INT AUTO_INCREMENT PRIMARY KEY NOT NULL,  
  NOMBRE VARCHAR(25),  
  ID_DEP INT,  
  FOREIGN KEY (ID_DEP) REFERENCES DEPARTAMENTO(ID_DEP)  
);
```

```
CREATE TABLE PERSONA(  
  ID_PER INT AUTO_INCREMENT PRIMARY KEY NOT NULL,  
  NOMBRE VARCHAR(25),  
  APELLIDOS VARCHAR(50),  
  FECHA_NAC DATE,  
  EDAD INT(11),  
  EMAIL VARCHAR(50),  
  ID_DEP INT,  
  ID_PROV INT,  
  SEXO CHAR(1),  
  FOREIGN KEY (ID_DEP) REFERENCES DEPARTAMENTO(ID_DEP),  
  FOREIGN KEY (ID_PROV) REFERENCES PROVINCIA(ID_PROV)  
);
```

```
CREATE TABLE PROYECTO(  
  ID_PROY INT(11) PRIMARY KEY AUTO_INCREMENT NOT NULL,  
  NOMBREPROY VARCHAR(100),  
  TIPOPROY VARCHAR(30)  
);
```

```
CREATE TABLE DETALLE_PROYECTO(  
  ID_DP INT(11) PRIMARY KEY AUTO_INCREMENT NOT NULL ,  
  ID_PER INT,  
  ID_PROY INT(11),  
  FOREIGN KEY (ID_PER) REFERENCES PERSONA(ID_PER),  
  FOREIGN KEY (ID_PROY) REFERENCES PROYECTO(ID_PROY)  
);
```

```
INSERT INTO DETALLE_PROYECTO(ID_PER, ID_PROY) VALUES  
(1, 1),  
(2, 2),  
(3, 2);
```

```
INSERT INTO PERSONA(NOMBRE, APELLIDOS, FECHA_NAC, EDAD, EMAIL, ID_DEP, ID_PROV, SEXO)  
VALUES  
( 'CARLOS', 'FERNANDEZ', '2000-06-24', 22, 'CARLOS@GMAIL.COM', 1, 1, 'M'),  
( 'ELENA', 'ROMERO ALARCON', '1997-09-1', 23, 'ROMAL@GMAIL.COM', 2, 3, 'F'),  
( 'KARINA', 'PACAJES PAZ', '2000-10-10', 22, 'KARINA@GMAIL.COM', 3, 6, 'F');
```

```
INSERT INTO PROVINCIA(NOMBRE, ID_DEP) VALUES  
( 'PACAJES', 1),  
( 'ITURRALDE', 1),  
( 'CERCADO', 1),  
( 'AZURDUY', 2),  
( 'SACABA', 2),  
( 'CEJA', 3);
```

```
INSERT INTO PROYECTO(NOMBREPROY, TIPOPROY) VALUES  
( 'PROYECTO PLANTAS', 'AGRICULTURA'),  
( 'PROYECTO SONRISAS', 'SALUD');
```

```
INSERT INTO DEPARTAMENTO (NOMBRE) VALUES  
( 'LA PAZ'),  
( 'COCHABAMBA'),  
( 'EL ALTO');
```


10. Crear una función que sume los valores de la serie Fibonacci.

```
CREATE FUNCTION FIBONACCI(LIM INT)
RETURNS TEXT
BEGIN
    DECLARE Y INT DEFAULT 0;
    DECLARE Z INT DEFAULT 1;
    DECLARE U INT DEFAULT 0;
    DECLARE X INT DEFAULT 1;
    DECLARE RESPONSE TEXT;
    IF LIM >= 1
    THEN
        SET RESPONSE = CONCAT(Y, ',');
    END IF;
    IF LIM >= 2
    THEN
        SET RESPONSE = CONCAT(RESPONSE, Z, ',');
    END IF;
    IF LIM >= 3
    THEN
        WHILE X <= (LIM - 2) DO
            SET U=Y+Z;
            SET RESPONSE = CONCAT(RESPONSE, U, ',');
            SET Y = Z;
            SET Z= U;
            SET X = X+1;
        END WHILE;
    END IF;
    RETURN RESPONSE;
END;
```

```
CREATE FUNCTION SUMANUMEROS(CADENA VARCHAR(100))
RETURNS INT
BEGIN
    DECLARE NUM VARCHAR(50) DEFAULT "";
    DECLARE RES INTEGER DEFAULT 0;
    DECLARE FIBO VARCHAR(1);
    DECLARE C INTEGER DEFAULT 1;

    IF LENGTH(CADENA) > 0 THEN
        WHILE(C <= LENGTH(CADENA)) DO
            SET FIBO= SUBSTRING(CADENA, C, 1);
            IF FIBO = ',' THEN
                SET RES = RES + NUM;
                SET NUM = "";
            ELSE
                SET NUM = CONCAT(NUM, FIBO);
            END IF;
            SET C = C + 1;
        END WHILE;
        RETURN RES;
    ELSE
        RETURN 0;
    END IF;
END;

SELECT SUMANUMEROS(FIBONACCI(10));
```

SQL> SUMANUMEROS(FIBONACCI(10))

1

88

11. Manejo de vistas.

- Crear una consulta SQL para lo siguiente.
- La consulta de la vista debe reflejar como campos:

1. nombres y apellidos concatenados
2. la edad
3. fecha de nacimiento.
4. Nombre del proyecto

- Obtener todas las personas del sexo femenino que hayan nacido en el departamento de El Alto en donde la fecha de nacimiento sea:

1. fecha_nac = '2000-10-10'

```
CREATE VIEW VISTA AS
SELECT CONCAT(P.NOMBRE, ' ', P.APELLIDOS) AS CONTNOM, P.EDAD,
P.FECHA_NAC, P3.NOMBREPROY
FROM PERSONA AS P
INNER JOIN DEPARTAMENTO D on P.ID_DEP = D.ID_DEP
INNER JOIN DETALLE_PROYECTO DP on P.ID_PER = DP.ID_PER
INNER JOIN PROVINCIA P2 on D.ID_DEP = P2.ID_DEP
INNER JOIN PROYECTO P3 on DP.ID_PROY = P3.ID_PROY
WHERE P.ID_DEP=3 AND P.FECHA_NAC='2000-10-10' AND P.SEXO='F';
```

```
SELECT *
FROM VISTA;
```

	CONTNOM	EDAD	FECHA_NAC	NOMBREPROY
1	KARINA PACAJES PAZ	22	2000-10-10	PROYECTO SONRISAS

Crear TRIGGERS Before or After para INSERT y UPDATE aplicado a la tabla PROYECTO

- Debera de crear 2 triggers minimamente.

```
CREATE TRIGGER TRIGGER1
BEFORE
  INSERT ON PROYECTO
FOR EACH ROW
BEGIN
  DECLARE NOMBRE VARCHAR(100);
  DECLARE TIPO VARCHAR(30);
  SET NOMBRE= NEW.NOMBREPROY;
  SET TIPO=NEW.TIOPROY;
end;

INSERT INTO PROYECTO(NOMBREPROY, TIOPROY)
VALUES('CORTE DE MADERAS', 'FORESTACION');
INSERT INTO PROYECTO(NOMBREPROY, TIOPROY)
VALUES('ANALFABETISMO', 'EDUCACION'),
      ('NOCHE DE MUSEOS', 'CULTURA');
INSERT INTO PROYECTO(NOMBREPROY, TIOPROY)
VALUES('DONACION DE SANGRE', 'FORESTACION');
```

	ID_PROY	NOMBREPROY	TIOPROY
1	1	PROYECTO PLANTAS	AGRICULTURA
2	2	PROYECTO SONRISAS	FORESTACION
3	4	CORTE DE MADERAS	FORESTACION
4	5	ANALFABETISMO	EDUCACION
5	6	NOCHE DE MUSEOS	CULTURA
6	7	DONACION DE SANGRE	FORESTACION


```
CREATE TRIGGER TRIGGER2
BEFORE
  UPDATE ON PROYECTO
FOR EACH ROW
BEGIN
  DECLARE TIPO VARCHAR(30);
  SET TIPO=NEW.TIPOPROY;
END;
```

```
UPDATE PROYECTO
SET TIPOPROY='FORESTACION'
WHERE TIPOPROY='SALUD';
```

- Agregar un nuevo campo a la tabla PROYECTO.
- El campo debe llamarse ESTADO

```
ALTER TABLE PROYECTO ADD ESTADO VARCHAR(50);
```

	ID_PROY	NOMBREPROY	TIPOPROY
1	1	PROYECTO PLANTAS	AGRICULTURA
2	2	PROYECTO SONRISAS	SALUD
3	4	CORTE DE MADERAS	SALUD
4	5	ANALFABETISMO	EDUCACION
5	6	NOCHE DE MUSEOS	CULTURA
6	8	DONACION DE SANGRE	SALUD

	ID_PROY	NOMBREPROY	TIPOPROY	ESTADO
1	1	PROYECTO PLANTAS	AGRICULTURA	<null>
2	2	PROYECTO SONRISAS	SALUD	<null>
3	4	CORTE DE MADERAS	SALUD	<null>
4	5	ANALFABETISMO	EDUCACION	<null>
5	6	NOCHE DE MUSEOS	CULTURA	<null>
6	8	DONACION DE SANGRE	SALUD	<null>

- o Actualmente solo se tiene habilitados ciertos tipos de proyectos.
- EDUCACION, FORESTACION y CULTURA
- o Si al hacer insert o update en el campo tipoProy llega los valores EDUCACION, FORESTACIÓN o CULTURA, en el campo ESTADO colocar el valor ACTIVO. Sin embargo se llega un tipo de proyecto distinto colocar INACTIVO
- o Adjuntar el código SQL generado y una imagen de su correcto funcionamiento.

```
CREATE TRIGGER TRIGGER3
BEFORE
INSERT ON PROYECTO
FOR EACH ROW
BEGIN
    DECLARE NOMBRE VARCHAR(100);
    DECLARE TIPO VARCHAR(30);
    SET NOMBRE= NEW.NOMBREPROY;
    SET TIPO=NEW.TIPOPROY;
    IF NEW.TIPOPROY='EDUCACION' OR
NEW.TIPOPROY='FORESTACION' OR NEW.TIPOPROY='CULTURA'
    THEN
        SET NEW.ESTADO='ACTIVO';
    ELSE
        SET NEW.ESTADO='INACTIVO';
    end if;
end;
INSERT INTO PROYECTO(NOMBREPROY, TIPOPROY)
VALUES('SEMILLAS', 'FORESTACION');
```

	ID_PROY	NOMBREPROY	TIPOPROY	ESTADO
1	1	PROYECTO PLANTAS	AGRICULTURA	<null>
2	2	PROYECTO SONRISAS	SALUD	<null>
3	4	CORTE DE MADERAS	SALUD	<null>
4	5	ANALFABETISMO	EDUCACION	<null>
5	6	NOCHE DE MUSEOS	CULTURA	<null>
6	8	DONACION DE SANGRE	SALUD	<null>
7	11	SEMILLAS	FORESTACION	ACTIVO



13. Manejo de Triggers II.

- El trigger debe de llamarse calculaEdad.
- El evento debe de ejecutarse en un BEFORE INSERT.
- Cada vez que se inserta un registro en la tabla PERSONA, el trigger debe de calcular la edad en función a la fecha de nacimiento.
- Adjuntar el código SQL generado y una imagen de su correcto funcionamiento.

```
CREATE TRIGGER CALCULAEDAD BEFORE INSERT
ON PERSONA
FOR EACH ROW
BEGIN
    DECLARE EDAD INTEGER DEFAULT 0;
    DECLARE EDAD2 INT DEFAULT 0;
    SET EDAD= (SELECT MAX(SUBSTR(NEW.FECHA_NAC, 1, 4))
    FROM PERSONA AS PER);
    SET EDAD2=(SELECT(SUBSTR(CURDATE(),1,4)));
    SET NEW.EDAD=EDAD2-EDAD;
END;
```

```
INSERT INTO PERSONA(NOMBRE, APELLIDOS, FECHA_NAC, EMAIL,
ID_DEP, ID_PROV, SEXO) VALUES('Anelis', 'Castillo', '2010-05-08',
'anelis@gmail.com', 3,6,'F');
```

	ID_PER	NOMBRE	APELLIDOS	FECHA_NAC	EDAD	EMAIL	ID_DEP	ID_PROV	SEXO
1	1	CARLOS	FERNANDEZ	2000-06-24	22	CARLOS@GMAIL.COM	1	1	M
2	2	ELENA	ROMERO ALARCON	1997-09-01	23	ROMAL@GMAIL.COM	2	3	F
3	3	KARINA	PACAJES PAZ	2000-10-10	22	KARINA@GMAIL.COM	3	6	F
4	4	Anelis	Castillo	2010-05-08	12	anelis@gmail.com	3	6	F

14. Manejo de TRIGGERS III.

- Crear otra tabla con los mismos campos de la tabla persona (Excepto el primary key id_per).
- No es necesario que tenga PRIMARY KEY.
- Cada vez que se haga un INSERT a la tabla persona estos mismos valores deben insertarse a la tabla copia.
- Para resolver esto deberá de crear un trigger before insert para la tabla PERSONA.
- Adjuntar el código SQL generado y una imagen de su correcto funcionamiento.

```
CREATE TRIGGER COPIA BEFORE INSERT
ON PERSONA
FOR EACH ROW
BEGIN
```

```
    INSERT INTO PERSONA_COPIA(nombre, apellidos, fecha_nac,
edad,email, id_dep, id_prov, sexo)
```

```
    VALUES(NEW.nombre, NEW.apellidos, NEW.fecha_nac,
NEW.edad,NEW.email, NEW.id_dep, NEW.id_prov, NEW.sexo);
END;
```

```
INSERT INTO PERSONA(NOMBRE, APELLIDOS, FECHA_NAC,
EDAD, EMAIL, ID_DEP, ID_PROV, SEXO) VALUES ('FERNADO',
'CLAROS AGUILAR', '2005-04-18',
17,'CLAROSFERNADO@GMAIL.COM',1,2, 'M');
```

	NOMBRE	APELLIDOS	FECHA_NAC	EDAD	EMAIL	ID_DEP	ID_PROV	SEXO
1	FERNADO	CLAROS AGUILAR	2005-04-18	17	CLAROSFERNADO@GMAIL.COM	1	2	M

	ID_PER	NOMBRE	APELLIDOS	FECHA_NAC	EDAD	EMAIL	ID_DEP	ID_PROV	SEXO
1	1	CARLOS	FERNANDEZ	2000-06-24	22	CARLOS@GMAIL.COM	1	1	M
2	2	ELENA	ROMERO ALARCON	1997-09-01	23	ROMAL@GMAIL.COM	2	3	F
3	3	KARINA	PACAJES PAZ	2000-10-10	22	KARINA@GMAIL.COM	3	6	F
4	4	Anelis	Castillo	2010-05-08	12	anelis@gmail.com	3	6	F
5	5	FERNADO	CLAROS AGUILAR	2005-04-18	17	CLAROSFERNADO@G...	1	2	M

15. Crear una consulta SQL que haga uso de todas las tablas.

- La consulta generada convertirlo a VISTA

```
CREATE VIEW VISTA2 AS
SELECT PER.NOMBRE, PER.APELLIDOS, PER.EDAD, D.NOMBRE AS
DEPARTAMENTO, P.NOMBRE AS PROVINCIA
FROM PERSONA AS PER
INNER JOIN DEPARTAMENTO D on PER.ID_DEP = D.ID_DEP
INNER JOIN DETALLE_PROYECTO DP on PER.ID_PER = DP.ID_PER
INNER JOIN PERSONA_COPIA PC on D.ID_DEP = PC.ID_DEP
INNER JOIN PROVINCIA P on D.ID_DEP = P.ID_DEP
INNER JOIN PROYECTO P2 on DP.ID_PROY = P2.ID_PROY
WHERE PER.EDAD >= 18 AND D.ID_DEP = 1;
```

	NOMBRE	APELLIDOS	EDAD	DEPARTAMENTO	PROVINCIA
1	CARLOS	FERNANDEZ	22	LA PAZ	PACAJES