



UNIVERSIDAD FRANZ TAMAYO

FACULTA DE TECNOLOGIA

CARRERA: INGENIERIA EN SISTEMAS

DEFENSA HITO 4

- **Nombre Completo:** Julio Marco Medrano
- **Asignatura:** BASE DE DATOS 2
- **Carrera:** INGENIERÍA DE SISTEMAS
- **Paralelo:** DBAII (1)
- **Docente:** Lic. William R. Barra Paredes
- **fecha:** 02/12\2019

COCHABAMBA-BOLIVIA

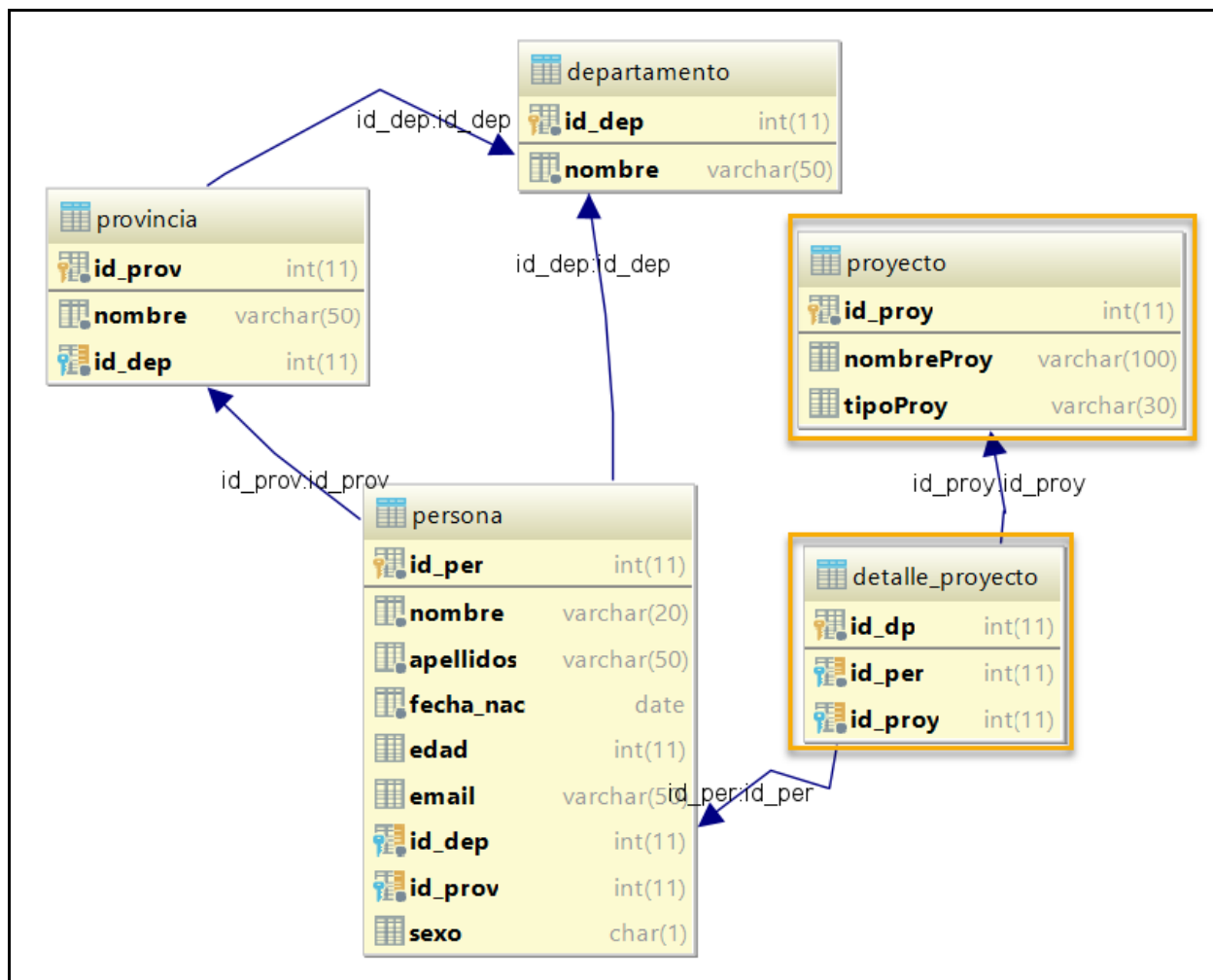
BASE DE DATOS II

LENGUAJE PROCEDURAL

TAREA ACUMULATIVA PARA EL HITO 4

Requisitos previos para poder dar solución:

- ❖ Crear la base de datos ONG.



```
USE ong;
```

```
CREATE TABLE proyecto  
(  
    id_proy integer auto_increment primary key not null,
```

```

nombreProy varchar(100),
tipoproj varchar(30)
);

CREATE TABLE detalle_proyecto
(
    id_dp INTEGER auto_increment primary key not null,
    id_per INTEGER,
    id_proy integer,
    foreign key(id_per) REFERENCES persona (id_persona),
    foreign key (id_proy) REFERENCES proyecto (id_proy)
);

ALTER TABLE provincia ADD column id_dep integer;
ALTER TABLE provincia ADD FOREIGN KEY (id_dep) REFERENCES
departamento(id_departamento);

```

Resolver los siguientes ejercicios.

→ Crear una Vista.

- La Vista debe de llamarse **personasMujeresDepartamento**.
- La consulta de la vista debe reflejar como campos nombres y apellidos concatenados, la edad y la fecha de nacimiento.
- Obtener todas las personas del sexo femenino que hayan nacido en el departamento de Cochabamba en donde la fecha sea:
 - o `fecha_nac = '1993-10-10'`.

```

DROP FUNCTION ObtenerPersona;
CREATE FUNCTION ObtenerPersona(fecha_nac date, departamento
TEXT, sexo TEXT) RETURNS BOOLEAN
BEGIN
    DECLARE result BOOLEAN DEFAULT FALSE;
    IF (fecha_nac = '1993-10-10' and departamento ='Cochabamba'
AND sexo = 'femenino')
    THEN
        Set result= true;
    end if;
    RETURN result;

end;

CREATE VIEW personasMujeresDepartamento AS
    SELECT CONCAT(p.nombres, '-', p.apellidos, '-', p.f_nacimiento, '-'
', d.nom_departamento)
As Detalle_persona

```

```

FROM persona p INNER JOIN departamento d on p.id_depto =
d.id_departamento
WHERE ObtenerPersona(p.f_nacimiento,d.nom_departamento,p.sexo);

SELECT v1.Detalle_persona FROM personasMujeresDepartamento AS
v1;

```

Detalle_persona	
1	Marlene-Uriarte-1993-10-10-Cochabamba
2	Eiza-Duarte-1993-10-10-Cochabamba

→ Crear una TRIGGER.

- El trigger debe de llamarse **backupPersonasUpdate**.
- El evento debe de ejecutarse en un **BEFORE UPDATE**.
- Crear un tabla llamada **backupPersonasUpdate**.
- Esta nueva tabla tiene que tener 2 campos **oldNombre** y **newNombre**.
- Cada vez que se modifique el registro de una persona, se debe de insertar un nuevo registro en la tabla **backupPersonasUpdate**, en donde el primer campo es el nombre que se está modificando y el segundo campo es el nuevo nombre.

```

DROP TABLE backupPersonasUpdate;
CREATE TABLE backupPersonasUpdate
(
    operation      CHAR(1)      NOT NULL, -- ('D', 'U', 'I')
    stamp          TIMESTAMP NOT NULL,
    userid         TEXT         NOT NULL,
    hostname       TEXT         NOT NULL,
    oldNombre      varchar(50),
    newNombre      VARCHAR(50),
    id_Persona     integer not null
);
DROP TRIGGER backupPersonasUpdate;
CREATE TRIGGER backupPersonasUpdate
    BEFORE UPDATE ON persona
    FOR EACH ROW
BEGIN
    INSERT INTO backupPersonasUpdate(operation, stamp, userid,
hostname,oldNombre, newNombre, id_Persona)
    SELECT 'U'
    ,now(),user(),@@hostname,old.nombres,NEW.nombres,OLD.id_persona;
end;

UPDATE persona set nombres= 'Javier'
WHERE id_persona =1;

```

	id_persona	nombres	apellidos	direccion	f_nacimiento	telefono	email
1	1	Javier	Vargas	Los Alamos 123	1993-10-10	65987512	jaime@gmail.com
2	2	Marlene	Uriarte	Los Claveles 123	1993-10-10	75087312	marlene@gmail.com
3	3	Eiza	Duarte	Las Gladiolas 123	1993-10-10	68592174	eiza@gmail.com

→ Crear una TRIGGER.

- El trigger debe de llamarse **calculaEdad**.
- El evento debe de ejecutarse en un **BEFORE INSERT**.
- Cada vez que se inserte un registro en la tabla PERSONA, el trigger debe de calcular la edad en función a la edad en función a la fecha de nacimiento.

```

ALTER TABLE persona ADD COLUMN edad integer;
CREATE TRIGGER calculaedad
BEFORE INSERT ON persona #NEW
FOR EACH ROW
BEGIN
    DECLARE edad INTEGER DEFAULT 0;
    SELECT TIMEDIFF(YEAR, NEW.f_nacimiento, CURDATE())
    INTO edad;
    SET NEW.edad = edad;
end;

INSERT INTO persona ( nombres, apellidos, direccion,
f_nacimiento, telefono, email, sexo, id_depto, id_prov, edad)
VALUES ('Roberto', 'Pena', 'Almafuerte 123', '1991-10-
28', 49118657, 'robertito@gmail.com', 'masculino', 3, 1, 0);

```

	id_persona	nombres	apellidos	direccion	f_nacimiento	telefono	email	sexo	id_depto
1	1	Javier	Vargas	Los Alamos 123	1993-10-10	65987512	jaime@gmail.com	masculino	
2	2	Marlene	Uriarte	Los Claveles 123	1993-10-10	75087312	marlene@gmail.com	femenino	
3	3	Eiza	Duarte	Las Gladiolas 123	1993-10-10	68592174	eiza@gmail.com	femenino	
4	4	Julio	Medrano	Lynch 3829	1992-09-09	65464448	ripperbios@gmail.com	masculino	
5	5	Deyna	Gutierrez	Martin Cardenas 123	2000-05-29	49118218	deygutierrez@gmail.com	femenino	
6	6	Remberto	Flores	Diaz Salgado 123	1989-04-25	65998512	remberto@gmail.com	masculino	
7	7	Roberto	Pena	Almafuerte 123	1991-10-28	49118657	robertito@gmail.com	masculino	


→ Crear un TRIGGER BEFORE o AFTER INSERT para la tabla PROYECTO.




- El nombre del TRIGGER deberá ser **triggerInsert_Proyecto**
- Deberá de crear una tabla de AUDITORIA en donde esta tabla deberá de tener 2 columnas.
 - El 1er campo debe de guardar el nuevo idProy insertado.
 - El 2do campo debe de guardar el nombre de proyecto y tipo de proyecto concatenados separados por un espacio.
 - Ejemplo: nombreProy: "Educacion para Ancianos", tipoProy: "Educacion".
 - Resultado: "Educacion para Ancianos - Educacion".

```
CREATE TABLE auditoria
(
operation      CHAR(1)      NOT NULL,
stamp          TIMESTAMP NOT NULL,
userid        TEXT        NOT NULL,
hostname      TEXT        NOT NULL,
nombreProy    VARCHAR(100),
id_Proj       integer not null
);

CREATE TRIGGER triggerInsert_Proyecto
BEFORE INSERT on proyecto
FOR EACH ROW
BEGIN
INSERT INTO auditoria(operation, stamp, userid, hostname,
nombreProy, id_Proj) SELECT
'I',NOW(),USER(),@@hostname,CONCAT(New.nombreProy,' -
',NEW.tipoproj),NEW.id_proj;
end;
DROP TRIGGER triggerInsert_Proyecto;

INSERT INTO proyecto (id_proy, nombreProy, tipoproj)
VALUES (1, 'Logitech', 'Tecnologia');
```

 <Filter criteria>

	 id_proy	 nombreProy	 tipoproj
1	1	Logitech	Tecnologia

→ Crear un TRIGGER BEFORE o AFTER para INSERT, UPDATE y DELETE para la tabla DETALLE_PROYECTO.

- El nombre del TRIGGER deberá ser **triggerForDetProy**
- Deberá de crear una tabla de AUDITORIA similar al siguiente ejemplo.

○

```
CREATE TABLE auditoria_detalle_proyecto (  
  operation      CHAR(1) NOT NULL, -- ('D', 'U', 'I')  
  stamp          TIMESTAMP NOT NULL,  
  userid         TEXT NOT NULL,  
  .....  
  todos los campos de la tabla detalle_proyecto  
);
```

○

- Debe de crear un Procedimiento Almacenado o Stored Procedure (SP).
 - Este SP debe recibir parámetros de entrada con los valores a insertar en la tabla de AUDITORIA.
- Los TRIGGERS deben de utilizar este SP, cada trigger debe de enviar los parámetros de inserción de la tabla de AUDITORIA

```
CREATE TABLE auditoria_detalle_proyecto (  
  operation      CHAR(1) NOT NULL,  
  stamp          TIMESTAMP NOT NULL,  
  userid         TEXT NOT NULL,  
  id_dp         INTEGER NOT NULL,  
  nombreProy     TEXT NOT NULL,  
  nombreProyold  text not null  
);  
  
create procedure insert_detalleProy_en_tablas(IN operacion char, IN  
id_dp integer, IN id_per integer, IN id_proy integer)  
begin  
  insert into auditoria_detalle_proyecto  
    (operation, stamp, userid, id_dp, nombreProy, nombreProyold)  
  select operacion, now(), user(), id_dp, id_per, id_proy;  
end;  
  
create trigger triggerForDetProy  
  after insert on auditoria_detalle_proyecto  
  for each row  
  begin  
    call insert_detalleProy_en_tablas('I', NEW.id_dp, NEW.nombreProy,  
new.nombreProyold);  
  end;  
  
create trigger triggerForDetProy_update  
  before update on auditoria_detalle_proyecto  
  for each row  
  begin  
    call insert_detalleProy_en_tablas('U', NEW.id_dp, NEW.nombreProy,  
new.nombreProyold);  
  end;
```

```

create trigger triggerForDetProy_delete
before delete on auditoria_detalle_proyecto
for each row
begin
    call insert_detalleProy_en_tablas('D',OLD.id_dp, old.nombreProy,
OLD.nombreProyold);
end;

```

→ Crear un TRIGGER BEFORE INSERT para la tabla DETALLE_PROYECTO.

- Si es LUNES o MARTES o MIÉRCOLES o JUEVES o VIERNES insertar adicionalmente los datos en su tabla de AUDITORÍA.
 - Adicionar un nuevo campo (diaDelaSemana varchar(12)) a la tabla auditoria_detalle_proyecto.
 - En este campo debe de almacenarse el dia en se insertó los nuevos registros.
- Si es SÁBADO o DOMINGO mostrar un mensaje indicando que no se permite inserciones los fines de semana. Por lo tanto no se debe de insertar en la tabla detalle_proyecto y tampoco en su tabla de auditoria.

```

ALTER TABLE auditoria_detalle_proyecto ADD COLUMN diaDelaSemana
varchar(12);
DROP TRIGGER DaysOfTheWeek;
CREATE TRIGGER DaysOfTheWeek
BEFORE INSERT on auditoria_detalle_proyecto
FOR EACH ROW
BEGIN
    if (diaDelaSemana='LUNES' OR diaDelaSemana='MARTES' OR
diaDelaSemana='MIERCOLES' OR diaDelaSemana='JUEVES' OR
diaDelaSemana='VIERNES')
    THEN
        INSERT INTO auditoria_detalle_proyecto(operation, stamp,
userid, id_dp,nombreProy,nombreProyold,diaDelaSemana) SELECT
'D',now(),user(),new.id_dp,new.nombreProy,
new.nombreProyold,new.diaDelaSemana;
    ELSE
        if (diaDelaSemana='SABADO' OR diaDelaSemana='DOMINGO')
        THEN SIGNAL sqlstate '45000' SET MESSAGE_TEXT = ' No se
permite inserciones los fines de semana.';

        end if;
    end if;
end;

INSERT INTO auditoria_detalle_proyecto (id_dp,
nombreProy,nombreProyold,diaDelaSemana)
VALUES (1,'Logitech','Genius','LUNES');

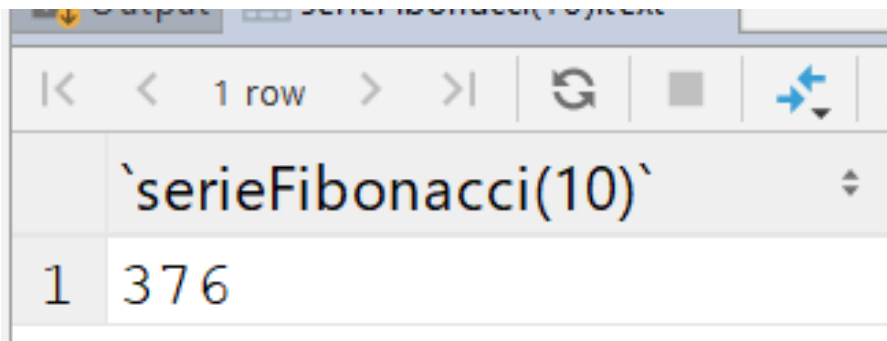
```


RESOLUCION DE SUMA DE SERIE FIBONACCI

```
DROP FUNCTION serieFibonacci;
CREATE FUNCTION serieFibonacci(num int) returns text
BEGIN
  DECLARE cont integer default 0;
  declare x integer default 1;
  declare y integer default 1;
  DECLARE aux integer default 0;
  DECLARE result TEXT DEFAULT '';
  DECLARE suma TEXT DEFAULT '';
  WHILE cont < num DO
SET aux=x+y;
  SET result=concat(result, aux,',');
  SET x=y;
  SET y=aux;

  SET cont=cont+1;
end while;
SET suma= aux+y+x-1;
RETURN suma;
end;

SELECT serieFibonacci(10);
```



The screenshot shows a database query result window. The title bar indicates the query is 'serieFibonacci(10)'. The window displays a single row of results. The first column contains the number '1', and the second column contains the string '376'. The window also shows navigation controls and a status bar indicating '1 row'.

`serieFibonacci(10)`	
1	376