

**UNIVERSIDAD PRIVADA FRANZ TAMAYO**

**EVALUACIÓN PROCESUAL - HITO 2**



**ESTUDIANTE:** DERECK DANNER HECTOR FERNANDEZ MENDOZA

**ASIGNATURA:** BASE DE DATOS II

**CARRERA:** INGENIERÍA DE SISTEMAS

**PARALELO:** BDA - 312

**DOCENTE:** WILLIAM RODDY BARRA PAREDES

**FECHA:** 11/09/2022

**GITHUB:** <https://github.com/DereckFM>

## **A) MANEJO DE CONCEPTOS**

### **1. ¿A qué se refiere cuando se habla de bases de datos relacionales?**

Estas bases de datos almacenan la información en tablas de datos estructurados y sus datos se pueden relacionar entre 2 y más tablas, esta base de datos funciona mejor cuando los datos son precisos y no cambian.

### **2. ¿A qué se refiere cuando se habla de bases de datos no relacionales?**

Estas bases de datos se diferencian de las anteriores en que estas en lugar de almacenar sus datos en tablas lo almacenan en documentos, además las bases de datos no relacionales son más flexibles que las relacionales.

### **3. ¿Qué es MySQL Y MariaDB?**

Ambas son bases de datos relacionales además de que tienen estructuras y funcionalidades muy similares. Una de las principales diferencias entre estos 2 es que MySQL es de código cerrado mientras que MariaDB es de código abierto, agregado a eso MariaDB es más rápido y ligero que MySQL, lo que hace que tenga un mejor rendimiento que este, en cambio MySQL es más adaptable a diferentes casos en su uso.

### **4. ¿Qué son las funciones de agregación?**

Las funciones de agregación se utilizan para agrupar datos de varias filas, columnas o tablas para formar un resultado que cumpla con las condiciones y parámetros solicitados por la función.

### **5. ¿Qué llegaría a ser XAMPP, WAMP SERVER o LAMP?**

Son paquetes de servidores informáticos de código libre que pueden ejecutarse en servidores web, el nombre de estos 3 servidores se debe a las siglas que conforman el sistema operativo que utilizan, el servidor web, el gestor de base de datos y el lenguaje de programación que manejan

X = Gran mayoría de los sistemas operativos existentes

W = Usa Windows como sistema operativo

L = Usa Linux como sistema operativo

A = Apache como servidor web

M = MySQL como sistema de gestor de datos

P = Generalmente Python, PHP o Perl

**6. ¿Cuál es la diferencia entre las funciones de agregación y funciones creadas por DBA?**

La principal diferencia es que las funciones de agregación ya están definidas por el sistema de gestión de datos (sum, avg, max, min) y estas devuelven un único valor, en cambio una función creada por un usuario se puede personalizar y puede recibir y devolver más de un parámetro.

**7. ¿Para qué sirve el comando USE?**

USE se utiliza para designar una base de datos para usar los datos almacenados de esta y recibir las próximas consultas SQL que se vayan a hacer.

**8. ¿Qué es DML y DDL?**

DML(data manipulation language) son las sentencias o palabras claves (INSERT, UPDATE, DELETE) que se utilizan para editar y manipular el contenido de la información de la tabla de una base de datos.

DDL(data definition language) son sentencias(CREATE, ALTER, DROP) que definen la estructura de la base de datos, sus tablas y subtablas.

**9. ¿Qué cosas características debe tener una función?**

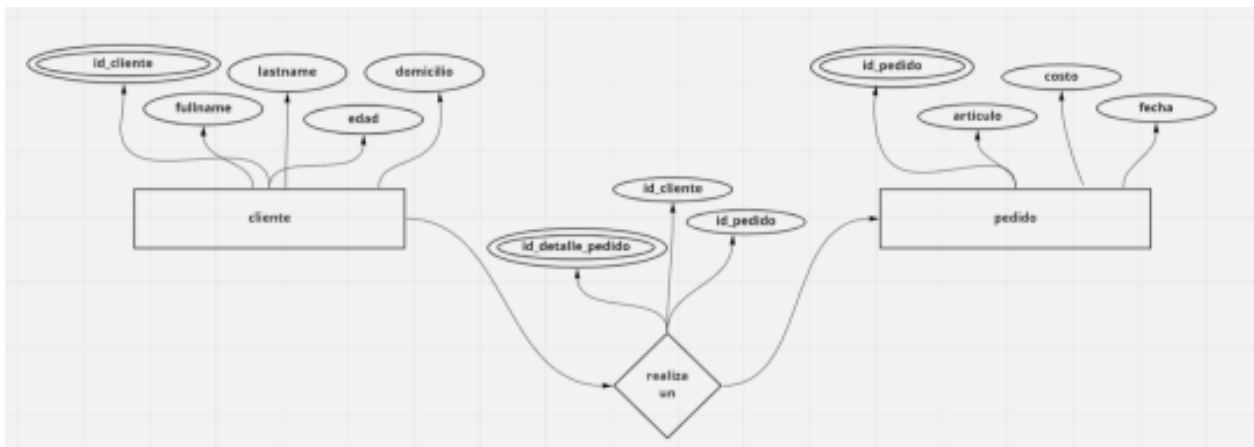
Algunas de las cosas características que debe tener una función son por ejemplo, el nombre ya que este ayuda a diferenciar entre funciones y facilita su uso dentro de consultas, luego está el returns que nos permite ingresar el tipo de dato que nos debe retornar la función, dentro del cuerpo de la función contiene las instrucciones para que nos devuelva el valor deseado, dentro de estas instrucciones están los parámetros.

## 10. ¿Cómo crear, modificar y eliminar una función?

Para crear una función se usa la sentencia “CREATE FUNCTION”, posteriormente para su modificación se le agrega a la sentencia anterior antes de “FUNCTION” las palabras “OR REPLACE” aunque esta puede variar dependiendo del gestor de datos que se esté utilizando, finalmente para su eliminación se utiliza la sentencia “DROP FUNCTION”.

### B) PARTE PRÁCTICA

1. Crear las tablas y 2 registros para cada tabla del siguiente modelo ER



Se sugiere crear una base de datos de nombre POLLOS\_COPA y en ella crear las tablas:

- cliente.
- detalle pedido.
- pedido.

Adjuntar el código SQL generado.

```

create database pollos_copa;

use pollos_copa;

create table cliente(
  id_cliente int primary key auto_increment not null,
  fullname varchar(100) not null,
  lastname varchar(100) not null,
  edad int not null,
  domicilio varchar(200) not null
);

insert into cliente(fullname, lastname, edad, domicilio)
values ('Juan','Perez',24,'Av Bolivia'),
('Pepito','Garcia',34,'Av 6 de agosto');

create or replace table pedido(
  id_pedido int primary key auto_increment not null,
  articulo varchar(100) not null,
  costo varchar(100) not null,
  fecha varchar(100) not null
);

insert into pedido(articulo, costo, fecha)
values ('Cubeta de pollo','50 Bs','2022-9-14'),
('Hamburguesa','15 Bs','2022-9-10');

create or replace table detalle_pedido(
  id_detalle_pedido int primary key auto_increment not null,
  id_cliente int not null,
  id_pedido int not null,
  foreign key(id_cliente) references cliente(id_cliente),
  foreign key(id_pedido) references pedido(id_pedido)
);

insert into detalle_pedido(id_cliente, id_pedido)
values(1,1),
(2,2);

```

Crear una consulta SQL en base al ejercicio anterior.

- Debe de utilizar las 3 tablas creadas anteriormente.
- Para relacionar las tablas utilizar JOINS.
- Adjuntar el código SQL generado

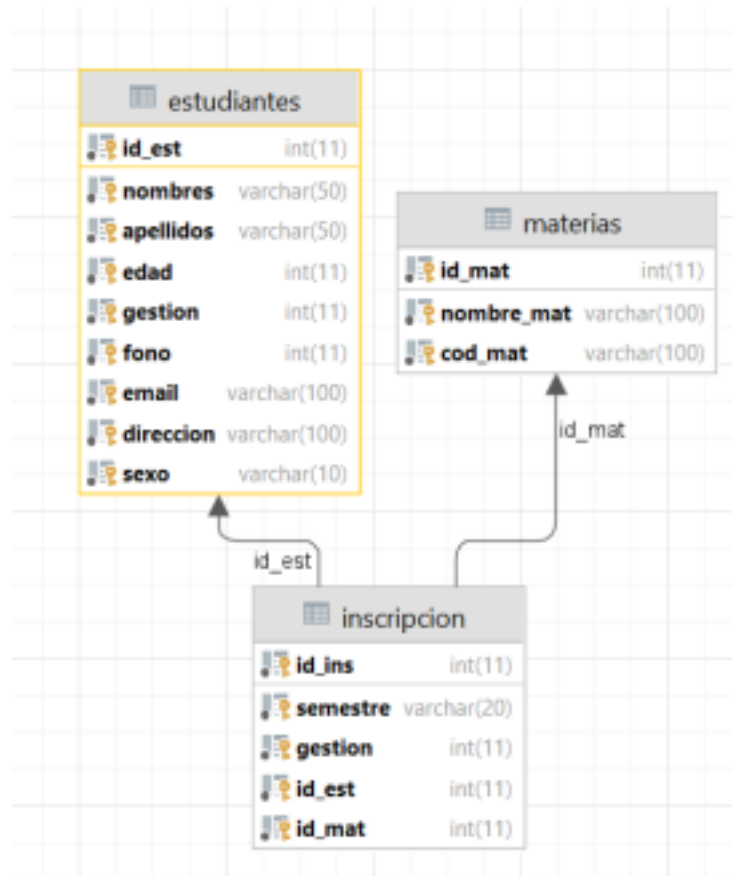
```

select c.lastname, c.fullname, c.domicilio, p.costo, p.fecha
from detalle_pedido as dp
inner join cliente c on dp.id_cliente = c.id_cliente
inner join pedido p on dp.id_pedido = p.id_pedido
where p.articulo = 'Hamburguesa'

```

Output					
Result 28					
	lastname	fullname	domicilio	costo	fecha
1	Garcia	Pepito	Av 6 de agosto	15 Bs	2022-9-10

2. Recrear la siguiente base de datos:



```
create database TareaHito2;
```

```
use TareaHito2;
```

```
create table estudiantes(  
  id_est int primary key auto_increment not null,  
  nombres varchar(50) not null,  
  apellidos varchar(50) not null,  
  edad int not null,  
  fono int not null,  
  email varchar(100) not null,  
  direccion varchar (100) not null,  
  sexo varchar(20) not null  
);
```

```
insert into estudiantes(nombres,apellidos,edad,fono,email,direccion,sexo)  
values('Miguel','Gonzales Veliz',20,2832115,'Miguel@gmail.com','Av. 6 de agosto','masculino'),  
      ('Sandra','Mavir Uria',25,2832116,'Sandra@gmail.com','Av. 6 de agosto','femenino'),  
      ('Joel','Aduburi Mondar',30,2832117,'Joel@gmail.com','Av. 6 de agosto','masculino'),  
      ('Andrea','Arias Ballesteros',21,2832118,'Andrea@gmail.com','Av. 6 de agosto','femenino'),  
      ('Santos','Montes Valenzuela',24,2832119,'Santos@gmail.com','Av. 6 de agosto','masculino');
```

```
create table materias(  
  id_mat int primary key auto_increment not null,  
  nombre_mat varchar(100) not null,  
  cod_mat varchar(100) not null  
);
```

```
insert into materias (nombre_mat,cod_mat)  
values ('introduccion a la arquitectura','ARQ-101'),  
      ('Urbanismo y diseño','ARQ-102'),  
      ('Dibujo y pintura arquitectonico','ARQ-103'),  
      ('Matematica discreta','ARQ-104'),  
      ('Fisica basica','ARQ-105');
```

```
create or replace table inscripcion(  
  id_ins int primary key auto_increment not null,  
  semestre varchar(20) not null,  
  gestion int not null,  
  id_est int not null,  
  id_mat int not null,  
  foreign key(id_est) references estudiantes(id_est),  
  foreign key(id_mat) references materias(id_mat)  
);
```

```
insert into inscripcion(id_est,id_mat,semestre,gestion)  
values(1,1,'1er Semestre',2018),  
(1,2,'2do Semestre',2018),  
(2,4,'1er Semestre',2019),  
(2,3,'2do Semestre',2019),  
(3,3,'2do Semestre',2020),  
(3,1,'3er Semestre',2020),  
(4,4,'4to Semestre',2021),  
(5,5,'5to Semestre',2021);
```

3. Resolver lo siguiente:

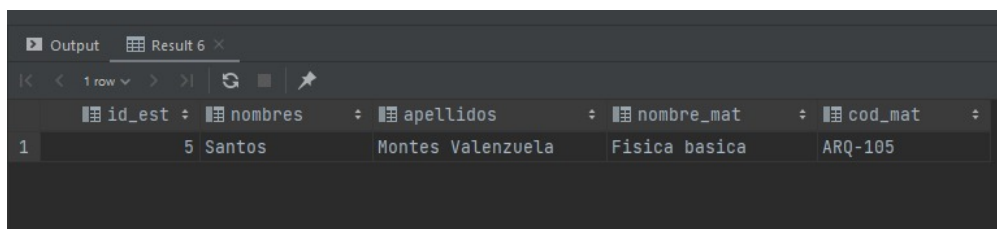
- Mostrar los nombres y apellidos de los estudiantes inscritos en la materia ARQ-105, adicionalmente mostrar el nombre de la materia.
- Deberá de crear una función que reciba dos parámetros y esta función deberá ser utilizada en la cláusula WHERE

```
select est.nombres, est.apellidos, mat.nombre_mat
from estudiantes as est
inner join inscripcion as ins on ins.id_est=est.id_est
inner join materias as mat on mat.id_mat=ins.id_mat
where cod_mat = 'ARQ-105';

create function Comparamaterias(cod_mat varchar(50), nombre_mat varchar(50)) returns boolean
begin
    declare answer boolean;

    if cod_mat = nombre_mat
    then
        set answer = 1;
    end if;
    return answer;
end;

select e.id_est, e.nombres, e.apellidos, m.nombre_mat, m.cod_mat
from inscripcion
inner join estudiantes e on inscripcion.id_est = e.id_est
inner join materias m on inscripcion.id_mat = m.id_mat
where Comparamaterias(m.cod_mat, 'ARQ-105');
```



The screenshot shows a database query output window with a table of results. The table has five columns: id\_est, nombres, apellidos, nombre\_mat, and cod\_mat. The first row contains the values 5, Santos, Montes Valenzuela, Fisica basica, and ARQ-105.

	id_est	nombres	apellidos	nombre_mat	cod_mat
1	5	Santos	Montes Valenzuela	Fisica basica	ARQ-105

4. Crear una función que permita obtener el promedio de las edades del género masculino o femenino de los estudiantes inscritos en la asignatura ARQ-104.

- La función recibe como parámetro el género y el código de materia.

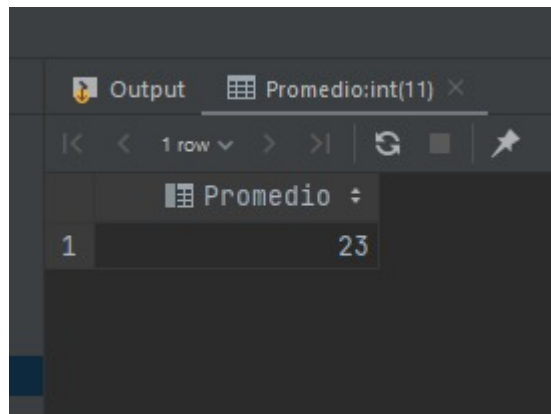


```

create function promedio(genero varchar(50), codmat varchar(50)) returns int
begin
    declare promedio int default 0;
    select avg(e.edad) into promedio
    from inscripcion
    inner join estudiantes as e on inscripcion.id_est = e.id_est
    inner join materias as m on inscripcion.id_mat = m.id_mat
    where e.sexo = genero and m.cod_mat = codmat;
    return promedio;
end;

select promedio('femenino','ARQ-104') as Promedio;

```



	Promedio
1	23

5. Crear una función que permita concatenar 3 cadenas.

- La función recibe 3 parámetros.
- Si las cadenas fuesen:
  - Pepito
  - Pep
  - 50
- La salida debería ser: (Pepito), (Pep), (50)
- Utilizar la función creada en una consulta SQL.

```

create function concatcadenas(c1 varchar(40), c2 varchar(40), c3 varchar(40) ) returns varchar(200)
begin
    declare chain varchar(70) default "";
    set chain = concat('(',c1,') (' ,c2,') (' ,c3,')');
    return chain;
end;
select concatcadenas('Pepito','Pep','50') as Datos;

```

1	(Pepito)	(Pep)	(50)
---	----------	-------	------

6. Crear una función de acuerdo a lo siguiente:

- Mostrar el nombre, apellidos, edad y el semestre de todos los estudiantes que estén inscritos.
- Siempre y cuando la suma de las edades del sexo femenino(tambien puede ser masculino) sea par y mayores a cierta edad.
- Debe de crear una función que sume las edades (recibir como parámetro el sexo, y la edad).
  - Ejemplo: sexo='Masculino' y edad=22
  - Note que la función recibe 2 parámetros.
- La función creada anteriormente debe utilizarse en la consulta SQL. (Cláusula WHERE).

```
create or replace function nombre(sexo varchar(50),edad int) returns boolean
begin
  declare suma int default 0;
  declare YN boolean default 0;
  select sum(est.edad) into suma
  from estudiantes as est
  where est.sexo = sexo;
  if suma >= edad and suma % 2=0
  then
    set YN = 1;
  end if;
  return YN;
end;
select sum(e.edad)
from estudiantes as e
where sexo = 'masculino'
group by (e.sexo);
select e.nombres, e.apellidos, e.edad, i.semestre
from inscripcion as i
inner join estudiantes e on i.id_est = e.id_est
where nombre('Masculino' ,22) and e.edad > 22;
```

Output Result 27				
	nombres	apellidos	edad	semestre
1	Sandra	Mavir Uria	25	1er Semestre
2	Sandra	Mavir Uria	25	2do Semestre
3	Joel	Aduburi Mondar	30	2do Semestre
4	Joel	Aduburi Mondar	30	3er Semestre
5	Santos	Montes Valenzuela	24	5to Semestre

7. Crear una función de acuerdo a lo siguiente:

- Crear una función sobre la tabla estudiantes que compara un nombre y apellidos. (si existe este nombre y apellido mostrar todos los datos del estudiante).
  - La función devuelve un boolean.
  - La función debe recibir 4 parámetros, nombres y apellidos.
  - La función debería ser usada en la cláusula WHERE.
  - El objetivo es buscar a estudiantes a través de sus nombres y apellidos.

```
create or replace function nombreCompleto(nombre varchar(100), nombre2 varchar(100), apellido
varchar(100), apellido2 varchar(100)) returns varchar(500)
begin
  declare comparador boolean;
  if nombre = nombre2 and apellido = apellido2
    then set comparador = 1;
  end if;
  return comparador;
end;

select *
from estudiantes as e
where nombreCompleto(e.nombres,'Andrea',e.apellidos,'Arias Ballesteros')
```

Ejemplo correcto.

```
select *
from estudiantes as e
where nombreCompleto( nombre: e.nombres, nombre2: 'Andrea', apellido: e.apellidos, apellido2: 'Arias Ballesteros')
```

Output

tareahito2.estudiantes

1 row

id\_est : nombres : apellidos : edad : fono : email : direccion : sexo :

1 4 Andrea Arias Ballesteros 21 2832118 Andrea@gmail.com Av. 6 de agosto femenino

Ejemplo equivocado:

```
select *
from estudiantes as e
where nombreCompleto( nombre: e.nombres, nombre2: 'Joel', apellido: e.apellidos, apellido2: 'Arias Ballesteros')
```

Output

tareahito2.estudiantes

0 rows

id\_est : nombres : apellidos : edad : fono : email : direccion : sexo :