

EVALUACION PROCESUAL HITO 2  
Base de Datos II

---

Estudiante:

ILIA ARACELI SARZO LAURA

Gestión:

3er SEMESTRE INGENIERA DE SISTEMAS

# MANEJO DE CONCEPTOS

---

- ¿A que se refiere cuando se habla de bases de datos relacionales?

Es una recopilación de elementos de datos con relaciones predefinidas entre ellos.

- ¿A que se refiere cuando se habla de bases de datos no relacionales?

Se caracterizan por tener una mayor escalabilidad y por soportar una estructura distribuida, son mas flexibles y permiten hacer cambios en los esquemas son para la base de datos.

- ¿Qué es MySQL y MariaDB?. Explique si existen diferencias o son iguales, etc.

MariaDB. - Es un sistema de gestión de base de datos.

MySQL. – Permite almacenar y acceder a los datos a través de multiples motores de almacenamiento.

MariaDB tiene licencia GPL mientras que MySQL tiene un enfoque de doble licencia, cada uno se acumula de una manera diferente. MariaDB soporta muchos motores de almacenamiento diferentes.



- ¿Qué son las funciones de agregación?

Es una función en la que los valores de varias filas se agrupan para formar un unico valor de resumen.

- ¿Qué llegaría a ser XAMPP, WAMP SERVER, o LAMP?

---

Xampp. - Es una distribución de Apache que incluye varios software libres.

Wamp Server. – El uso de Wamp permite subir paginas html a internet, ademas de poder gestionar datos en ellas.

Lamp. – Sistema operativo Linux, un servidor web Apache, una base de datos MySQL y lenguaje de Programación PHP.

¿Cuál es la diferencia entre las funciones de agregación y funciones creados por el DBA? Es decir funciones creadas por el usuario.

Las funciones de agregacion son las que ya vienen con la base de datos.

Las creadas por el usuario son las que necesitan ser creadas, estas pueden ser funciones de agregacion.

- ¿Para que sirve el comando USE?

Indica al usuario cual base de datos usar.

- ¿Qué es DML y DDL?

DDL. – Son sentencias utilizadas para la creación de base de datos, tablas, triggers, etc.

---

¿Qué cosas características debe tener una función? Explique sobre el nombre, el return, parámetros, etc.

9. ¿Qué cosas características debe de tener una función? Explique sobre el nombre, el return, parametros, etc.

Una función debe tener las siguientes partes

**CREATE FUNCTION OR REPLACE NombreDeLaFuncion** (parametros a recibir)

**RETURNS INTEGER** <---Dato\_A\_devolver

**BEGIN**

**DECLARE NUMERO INTEGER DEFAULT 0** < -----declaración de datos para usar, si son necesarios

**SET NUMERO=2;** <----- Procedimiento de la funcion

**RETURN NUMERO;** <----dato\_a\_enviar

**END;**

**SELECT NombreDeLaFuncion** (parámetros a enviar);

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

Para crear una función el código es **CREATE FUNCTION**

Para modificar una función es **REPLACE**

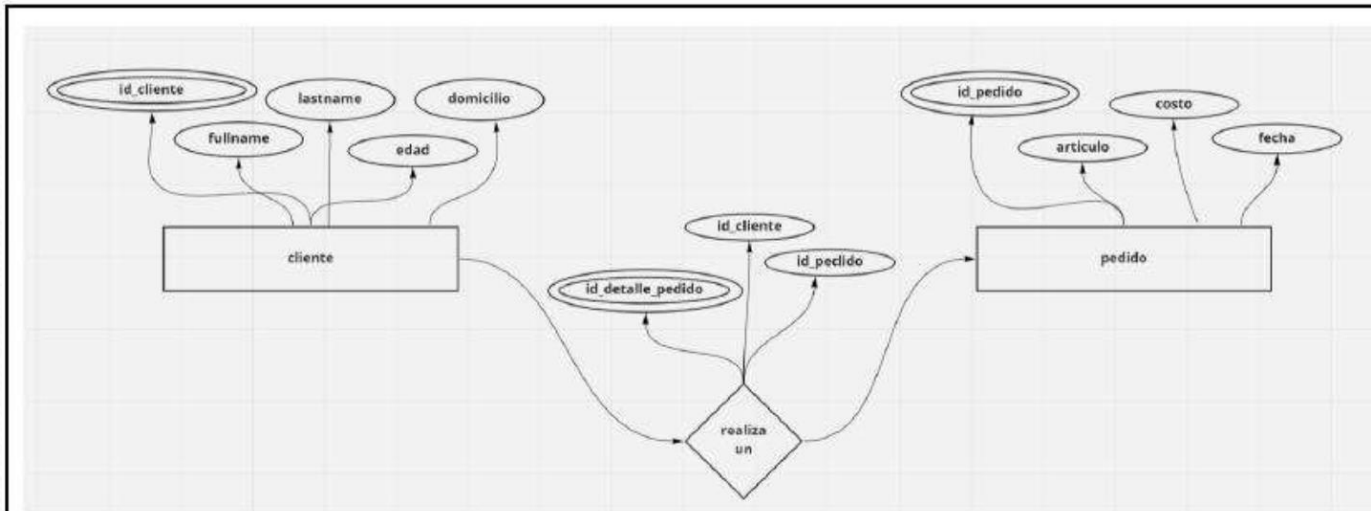
Para eliminar la función es **DROP FUNCTION (nombre de la funcion);**

- ¿Cómo crear, modificar y como eliminar una función?
- Para crear una función se pone el código: Create Function
- Para modificar una función se pone el código: Replace
- Para eliminar una función se pone el código: Drop Function.



# PARTE PRACTICA

- Crear las tablas y 2 registros para cada tabla para el siguiente modelo ER.



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

- cliente
- detalle pedido
- pedido
- adjuntar código SQL

```

create database Pollos_Copa;
use Pollos_Copa;

create table Cliente(
    id_cliente integer auto_increment primary key not null,
    fullname varchar(100),
    lastname varchar(100),
    edad integer,
    domicilio varchar(100)
);

create table Pedido(
    id_pedido integer auto_increment primary key not null,
    articulo varchar(100),
    costo varchar(100),
    fecha varchar(100)
);

create table Detalle_Pedido
(
    id_detalle_pedido integer auto_increment primary key 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)
);

```

```

    id_pedido int not null,
    foreign key (id_cliente) references cliente (id_cliente),
    foreign key (id_pedido) references pedido (id_pedido)
);

insert into Cliente(fullname , lastname , edad , domicilio) values
('Vladimir' , 'Putin' , 60 , 'Moscu'),
('Homero' , 'Simpson' , 40 , 'Springfield');

insert into Pedido(articulo , costo , fecha) values
('Celular' , '4000bs' , '05/09/22'),
('Reloj' , '1300bs' , '02/05/22');

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

select c.fullname , c.lastname , c.domicilio
from detalle_pedido
inner join cliente c on Detalle_Pedido.id_cliente = c.id_cliente
inner join pedido p on Detalle_Pedido.id_pedido = p.id_pedido
where c.domicilio = 'Moscu';

```

cliente	
id_cliente	int(11)
fullname	varchar(100)
lastname	varchar(100)
edad	int(11)
domicilio	varchar(100)

pedido	
id_pedido	int(11)
articulo	varchar(100)
costo	varchar(100)
fecha	varchar(100)

id\_cliente

id\_pedido

detalle_pedido	
id_detalle_pedido	int(11)
id_cliente	int(11)
id_pedido	int(11)



1	1	Vladimir	Putin	60	Moscu
2	2	Homero	Simpson	40	Springf...

<div> <div> <div>&lt;</div> <div>&gt;</div> <div>2 rows</div> </div> <div> <div>↺</div> <div>■</div> <div>+</div> <div>-</div> <div>↻</div> <div>🔄</div> <div>↑</div> </div> <div> <div>Tx: Auto</div> <div>DDL</div> <div>🌟</div> </div> </div>					
	id_pedido	articulo	costo	fecha	
1	1	Celular	4000bs	05/09/22	
2	2	Reloj	1300bs	02/05/22	

	id_detalle_pedido	id_cliente	id_pedido
1	1	1	2
2	2	1	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 detalle_pedido. id_detalle_pedido as compra,  
concat(c.fullname, c.lastname) as nombreCompleto, p.articulo, p.costos  
from detalle_pedido  
inner join pedido p on Detalle_Pedido.id_pedido = p.id_pedido  
inner join cliente c on Detalle_Pedido.id_cliente = c.id_cliente;
```

Output Result 10

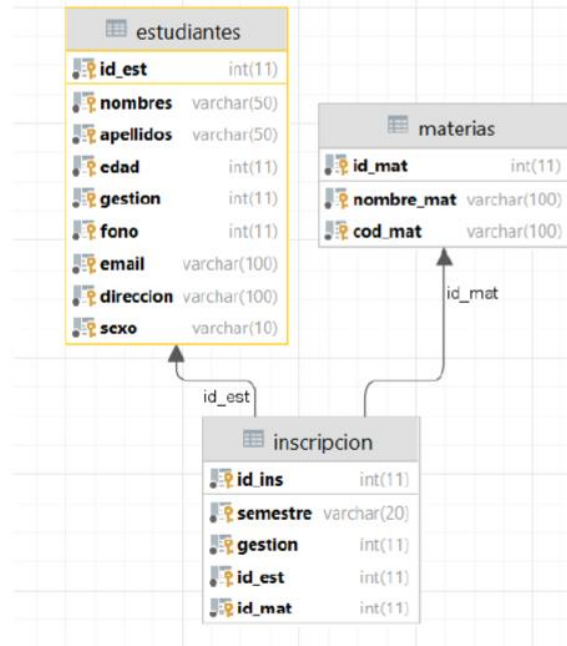
6 rows

	compra	nombreCompleto	articulo	costo
1	1	VladimirPutin	Reloj	1300bs
2	2	VladimirPutin	Reloj	1300bs



- Crear un función que compare dos códigos de materia.
- ○ Recrear la siguiente base de datos:

```
CREATE DATABASE tareaHito2;
USE tareaHito2;
```



```
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', 'Adubiri 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');
```

```
INSERT INTO materias (nombre_mat, cod_mat)
VALUES ('Introduccion a la Arquitectura', 'ARQ-101'),
('Urbanismo y Diseno', 'ARQ-102'),
('Dibujo y Pintura Arquitectonico', 'ARQ-103'),
('Matematica discreta', 'ARQ-104'),
('Fisica Basica', 'ARQ-105');
```

```
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);
```

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.

```
57 # mostrar los nombres y apellidos de los estudiantes inscritos en la
58 # materia ARQ-105, adicionalmente mostrar el nombre de la materia.
59 create or replace function comparar_materias(cod_mat varchar(50) , nombre_mat varchar(50)) returns boolean
60 begin
61     declare respuesta boolean;
62
63     if cod_mat = nombre_mat
64     then
65         set respuesta = 1;
66     end if;
67     return respuesta;
68 end;
69
70 select est.id_est , est.nombres , est.apellidos , mat.nombre_mat , mat.cod_mat
71 from inscripcion
72 inner join estudiantes est on Inscripcion.id_est = est.id_est
73 inner join materias mat on Inscripcion.id_mat = mat.id_mat
74 where comparar_materias(mat.cod_mat, 'ARQ-105');
75
```

Output Result 15

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



- 14. 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.

```
5
6
7 # 14. Crear una función que permita obtener el promedio de las edades del género
8 # masculino o femenino de los estudiantes inscritos en la asignatura ARQ-104.
9 # ○ La función recibe como parámetro el género y el código de materia.
10 create or replace function promedio(genero varchar(40) , materia_cod varchar(40)) returns integer
11 begin
12     declare promedio integer default 0;
13     select avg(est.edad) into promedio
14     from Incripcion
15     inner join estudiantes as est on Incripcion.id_est = est.id_est
16     inner join materias as mat on Incripcion.id_mat = mat.id_mat
17     where est.sexo = genero and mat.cod_mat = materia_cod;
18     return promedio;
19 end;
20 ✓ select promedio('Femenino' , 'ARQ-104');
```

Output promedio('Femenino' , 'ARQ-104');int(11) ×

1	23

15. 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)

○ La función creada utilizarlo en una consulta SQL.

```
# 15. 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)
# ○ La función creada utilizarlo en una consulta SQL.

create or replace function concatenado_tres_cadenas(m1 varchar(40) , m2 varchar(40) , m3 varchar(40)) returns varchar(100)
begin
    declare cadena varchar(50) default '';
    set cadena = concat('('||m1||') ('||m2||') ('||m3||')');
    return cadena;
end;

select concatenado_tres_cadenas('Andres' , 'Andresito' , '20') as concatenado;
```

Output concatenado:varchar(100) x

1 row

concatenado

1 (Andres) (Andresito) (20)



16. 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 (también 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).

```
115 # ○ Debe de crear una función que sume las edades (recibir como parámetro el
116 # sexo, y la edad).
117 # ■ Ejemplo: sexo='Masculino' y edad=22
118 # ■ Note que la función recibe 2 parámetros.
119 # ○ La función creada anteriormente debe utilizarse en la consulta SQL.
120 # (Cláusula WHERE).
121
122 create or replace function fullname(sexo varchar(50) , edad integer) returns boolean
123 begin
124     declare suma integer default 0;
125     declare YesorNo boolean;
126     select sum(est.edad) into suma
127     from estudiantes as est
128     where est.sexo=sexo;
129     if suma %2=0 and suma>edad
130     then
131         set YesorNo = 1;
132     end if;
133     return YesorNo;
134 end;
135
136 ✓ select est.nombres , est.apellidos , est.edad , inscripcion.semestre
137 from inscripcion
138 inner join estudiantes est on Incripcion.id_est = est.id_est
139 where fullname('Masculino' , 23);
```

	📄 nombres	↕	📄 apellidos	↕	📄 edad	↕	📄 semestre	↕
1	Miguel		Gonzales Veliz		20		1er Semestre	
2	Miguel		Gonzales Veliz		20		2do Semestre	
3	Sandra		Mavir Uria		25		1er Semestre	
4	Sandra		Mavir Uria		25		2do Semestre	
5	Joel		Adubiri Mondar		30		2do Semestre	
6	Joel		Adubiri Mondar		30		3er Semestre	
7	Andrea		Arias Ballesteros		21		4to Semestre	
8	Santos		Montes Valenzuela		24		5to Semestre	
9	Miguel		Gonzales Veliz		20		1er Semestre	
10	Miguel		Gonzales Veliz		20		2do Semestre	
11	Sandra		Mavir Uria		25		1er Semestre	

```
select sum(est.edad)
from estudiantes as est
group by (est.sexo);
```

Output sum(est.edad):int(11) ✕

2 rows

📄 `sum(est.edad)`

1	184
2	296



- 17. 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.
  - ■ Similar al siguiente ejemplo.

```
# 17. 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.
# ■ Similar al siguiente ejemplo.
```

```
create or replace function nombres_apellidos(nombres varchar(100) , apellidos varchar(100) , nombre1 varchar(100) , apellidos1 varchar(100)) returns
begin
    declare comparar boolean;
    if nombres = nombre1 and apellidos = apellidos1
    then
        set comparar = 1;
    end if;
    return comparar;
end;

select *
from estudiantes as est
where nombres_apellidos(est.nombres , est.apellidos , 'Sandra' , 'Mavir Uria');
```

Output tareahito2.estudiantes

	id_est	nombres	apellidos	edad	gestion	fono	email	direccion	sexo
1	2	Sandra	Mavir Uria	25	<null>	2832116	sandra@gmail.com	Av. 6 de Agosto	femenino



# GRACIAS POR SU ATENCIÓN

[eate.iliaaraceli.sarzo.la@unifranz.edu.bo](mailto:eate.iliaaraceli.sarzo.la@unifranz.edu.bo)