

DEFENSA HITO 2

TAREA FINAL

Estudiante: Machaca Quispe Jose

Asignatura: Base de Datos II

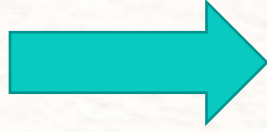
Carrera: Ingeniería de Sistemas

Docente: Lic. William Barra Paredes

Fecha: 11/09/2022

PREGUNTAS TEÓRICAS.

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



Una base de datos relacional es un tipo de base de datos que almacena y proporciona acceso a puntos de datos relacionados entre sí.

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



Las bases de datos no relacionales son un sistema de almacenamiento de información que se caracteriza por no usar el lenguaje SQL para las consultas. Esto no significa que no puedan usar el lenguaje SQL, pero no lo hacen como herramienta de consulta, sino como apoyo.

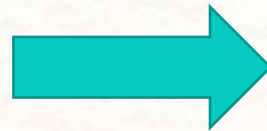
3. ¿Qué es MySQL y MariaDB?. Explique si existen diferencias o son iguales, etc.



Aunque MariaDB es una bifurcación de MySQL, estos dos sistemas de gestión de bases de datos siguen siendo bastante diferentes:

- MariaDB tiene licencia GPL mientras que MySQL tiene un enfoque de doble licencia.
 - Cada mango se acumula de una manera diferente.
- MariaDB soporta muchos motores de almacenamiento diferentes.
- En muchos escenarios, MariaDB ofrece un mejor rendimiento.

4. ¿Qué son las funciones de agregación?



Una función de agregación es una función que resume las filas de un grupo en un solo valor. COUNT , MIN y MAX

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



Son una distribución de Apache que incluye diferentes softwares libres, además son servidores web multiplataforma más utilizados, que ayuda a los desarrolladores a crear y probar sus programas en un servidor web local.

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



Una función de agregación es una función que resume las filas de un grupo en un solo valor. Las funciones de DBA se basa principalmente en reglas y hechos que son almacenados en la base de datos.

7. ¿Para qué sirve el comando USE?



La sentencia USE db_name indica a MySQL que use la base de datos db_name como la base de datos por defecto (actual) en sentencias subsiguientes. La base de datos sigue siendo la base de datos por defecto hasta el final de la sesión o hasta que se use otra sentencia USE

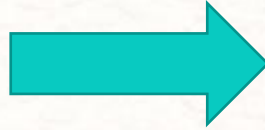
8. Que es DML y DDL?



- DML significa Data Manipulation Language o Lenguaje de Manipulación de Datos, en español. Este lenguaje permite realizar diferentes acciones a los datos que se encuentran en una base de datos. Permite recuperar, almacenar, modificar, eliminar, insertar y actualizar datos de una base de datos.
- DDL significa Data Definition Language o Lenguaje de Definición de Datos, en español. Este lenguaje permite definir las tareas de las estructuras que almacenarán los datos.

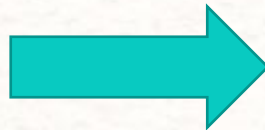
9. ¿Qué cosas características debe de tener una función?

Explique sobre el nombre, el return, parametros, etc.



- MySQL permite nombres que consisten en un único identificador o en múltiples identificadores. Los componentes de un nombre compuesto deben estar separados por un punto ('.'). Las partes iniciales de un nombre compuesto actúan como calificadores que afectan al contexto dentro de cual, se interpreta el identificador final.
- RETURN se utiliza para concluir la ejecución de una rutina. Para las funciones o métodos de SQL, devuelve el resultado de la función o método. Para un procedimiento SQL, devuelve opcionalmente un valor de estado de tipo entero.
- Son parámetros a los que vamos a poder dar un valor inicial, llamar al procedimiento y que pueda hacer uso del valor enviado y dentro del procedimiento va a poder cambiar su valor y este será reflejado desde donde se llamó

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



1. Creación de la base de datos: CREAM (CREATE).
2. Segundo, vea la lista de la base de datos: MOSTRAR (SHOW).
3. Modificación de la base de datos: ALERTA Modificar la codificación de la base de datos (ALTER).
4. Cuarto, elimine la base de datos. (DROP)

11. 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 crear las tablas:

- cliente
- detalle_pedido
- pedido

- Adjuntar el código SQL generado.

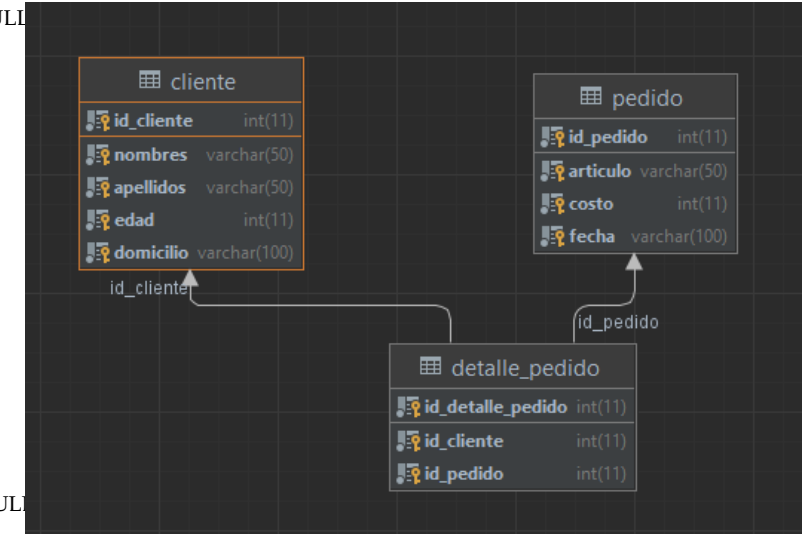
```
CREATE DATABASE POLLOS_COPA;
USE POLLOS_COPA;
CREATE TABLE cliente
(
  id_cliente INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
  nombres VARCHAR(50),
  apellidos VARCHAR(50),
  edad INTEGER,
  domicilio VARCHAR(100)
);
```

```
INSERT INTO cliente(nombres,apellidos,edad,domicilio)
VALUES
('Angel','Gonzales Mamani','20','Av. Calacoto'),
('David','Alcon Garmendia','25','Av. Sucre'),
('Ema','Suri Belgrano','24','Av. La Paz'),
('Angelica','Flores Bella','20','Av. Calacoto'),
('Joel','Gutierrez','26','Av. Libertad');
```

```
CREATE TABLE pedido
(
  id_pedido INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
  articulo VARCHAR(50),
  costo INTEGER,
  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)
);
```

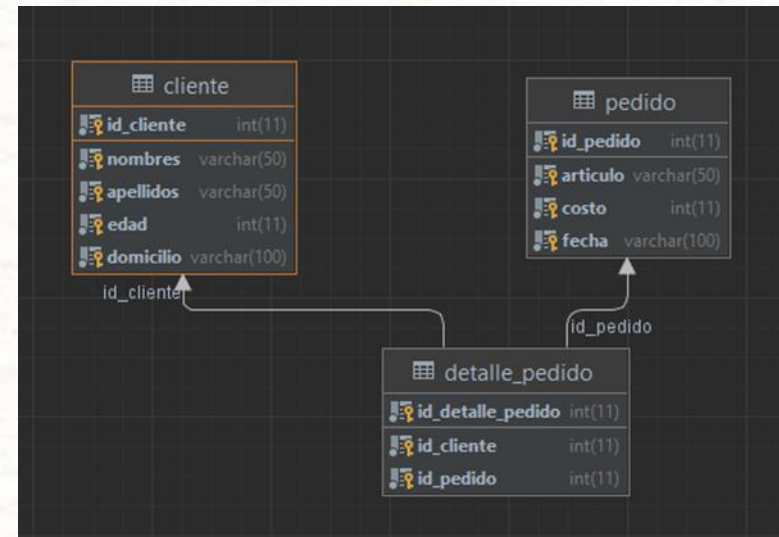
```
INSERT INTO pedido(articulo,costo,fecha)VALUES
('Combo de papafritas','15','01-Abril-2022'),
('Combo de pollo','35','02-Abril-2022'),
('Combo de papafritas','15','02-Abril-2022'),
('Combo de mixto','15','03-Abril-2022'),
('Combo grande de pollo','15','04-Abril-2022');
```



12. 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.

```
47 SELECT*FROM cliente
48     INNER JOIN pedido
49     ON pedido.id_pedido=cliente.id_cliente;
50
51 SELECT*FROM cliente
52     INNER JOIN detalle_pedido
53     ON detalle_pedido.id_detalle_pedido=cliente.id_cliente;
```



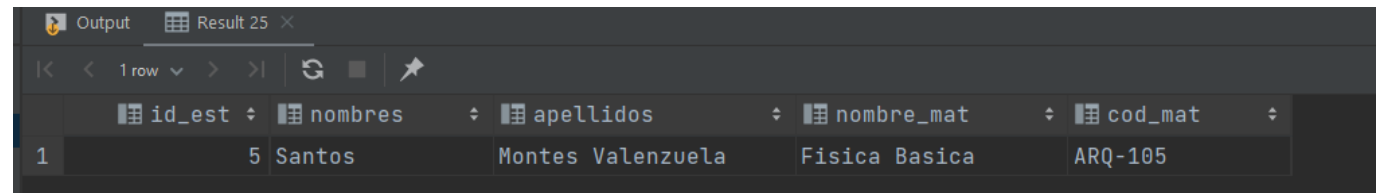
13. Crear un función que compare dos códigos de materia.

○ 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.

```
407 CREATE FUNCTION ins_est(cod_mat VARCHAR (50))
408 RETURNS INTEGER
409 BEGIN
410     DECLARE parametro INTEGER DEFAULT 0;
411     SELECT est.id_est INTO parametro
412     FROM estudiantes AS est
413     INNER JOIN inscripcion AS ins 1<->1..n: ON est.id_est =ins.id_est
414     INNER JOIN materias AS mat 1..n<->1: ON ins.id_mat = mat.id_mat
415     WHERE mat.cod_mat = cod_mat;
416     RETURN parametro;
417
418 END;
419
420 SELECT est.id_est, est.nombres, est.apellidos, mat.nombre_mat, mat.cod_mat
421 FROM estudiantes AS est
422 INNER JOIN inscripcion AS ins 1<->1..n: ON est.id_est = ins.id_est
423 INNER JOIN materias AS mat 1..n<->1: ON ins.id_mat = mat.id_mat
424 WHERE ins_est( cod_mat: 'ARQ-105') = est.id_est;
```



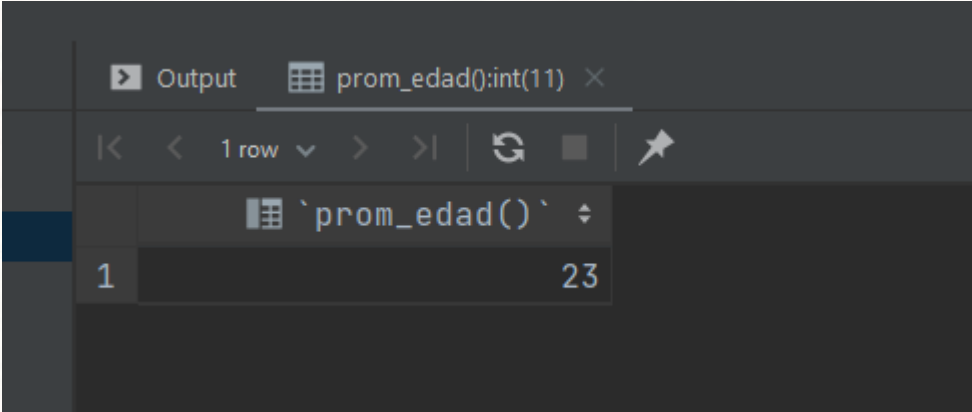
The screenshot shows a database interface with a tab labeled 'Result 25'. It displays a single row of data from a query. The columns are: id_est, nombres, apellidos, nombre_mat, and cod_mat. The values in the row are: 5, Santos, Montes Valenzuela, Fisica Basica, and ARQ-105.

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

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 solo el género.
- La función retorna un valor numérico.

```
426 # EJERCICIO 14 DE LA PRACTICA
427
428 CREATE OR REPLACE FUNCTION prom_edad()
429 RETURNS INTEGER
430 BEGIN
431     return(
432
433         SELECT AVG(est.edad)
434         FROM estudiantes AS est
435         WHERE est.genero = 'femenino'
436     );
437 end;
438 SELECT prom_edad();
439
```



The screenshot shows a database interface with an 'Output' window. The window title is 'prom_edad():int(11)'. It displays a single row of data. The first column is labeled with the function name 'prom_edad()' and the second column contains the value 23.

prom_edad()
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

```
440 # EJERCICIO 15 DE LA PRACTICA
441
442 CREATE OR REPLACE FUNCTION concat_nombre(par1 VARCHAR(50), par2 VARCHAR(50), par3 VARCHAR(50))
443 RETURNS VARCHAR(50)
444 BEGIN
445     DECLARE concatenado VARCHAR(50) DEFAULT '';
446     SET concatenado = CONCAT(par1, ' - ', par2, ' - ', par3);
447     RETURN concatenado;
448 end;
449
450 SELECT concat_nombre( par1: 'Pepito', par2: 'Pep', par3: '50');
```

Output	
concat_nombre('Pepit...p', '50'):varchar(50) ×	
1 row	
`concat_nombre('Pepito', 'Pep', '50')`	
1	Pepito - Pep - 50

16. Crear una función de acuerdo a lo siguiente:

- Mostrar el nombre, apellidos y el semestre de todos los estudiantes que estén inscritos. Siempre y cuando la suma de las edades del sexo femenino o 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).

```

452 # EJERCICIO 16 DE LA PRACTICA
453
454 CREATE FUNCTION est_ins(genero varchar(10), edad INT)
455 RETURNS INTEGER
456 BEGIN
457     DECLARE sumaEdad INTEGER DEFAULT 0;
458     SELECT SUM(est.edad) INTO sumaEdad
459     FROM estudiantes AS est
460     WHERE est.genero = genero AND est.edad >= edad;
461     RETURN sumaEdad;
462 end;
463
464 SELECT est.nombres, est.apellidos, ins.semestre
465 FROM estudiantes AS est
466 INNER JOIN inscripcion AS ins ON est.id_est = ins.id_est
467 WHERE est_ins(genero: 'masculino', edad: '22') %2 =0;

```

Output Result 11

8 rows

	nombres	apellidos	semestre
1	Miguel	Gonzales Veliz	1er Semestre
2	Miguel	Gonzales Veliz	2do Semestre
3	Sandra	Mavir Uria	1er Semestre
4	Sandra	Mavir Uria	2do Semestre
5	Joel	Adubiri Mondar	2do Semestre
6	Joel	Adubiri Mondar	3er Semestre
7	Andrea	Arias Ballesteros	4to Semestre
8	Santos	Montes Valenzuela	5to Semestre

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 el nombre y sus apellidos.

```
469 # EJERCICIO 17 DE LA PRACTICA
470
471 CREATE OR REPLACE FUNCTION comparar(nombre VARCHAR(50), apellido VARCHAR(50))
472 RETURNS BOOLEAN
473 BEGIN
474     DECLARE validar BOOLEAN;
475     SELECT est.id_est INTO validar
476     FROM estudiantes AS est
477     WHERE est.nombres = nombre AND est.apellidos = apellido;
478     RETURN validar;
479 end;
480
481 SELECT est.*
482 FROM estudiantes AS est
483 WHERE comparar( nombre: 'Miguel', apellido: 'Gonzales Veliz') = est.id_est;
```

Output function_aggregation.estudiantes									
1 row									
	id_est	nombres	apellidos	edad	gestion	fono	email	direccion	genero
1	1	Miguel	Gonzales Veliz	20	2022	<null>	miguel@gmail.com	Av. 6 de Agosto	masculino