TAREA HITO 2

MELANIE INGRID VILLCA COPA

MANEJO DE CONCEPTOS

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

Una base de datos relacional es básicamente un conjunto de tablas (relaciones bidimensional es), similares a las tablas de una hoja de cálculo, formadas por filas (registros) y columnas (campos).

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

Una base de datos no relacional también conocido no solo SQL (Not Only SQL) es una amplia clase de sistemas facilitando un crecimiento horizontal, enfocándose en rendimiento más que gestión de datos, caracterizado por no usar SQL como el principal lenguaje de consultas. Capaz de almacenar grandes cantidades de datos e en consistencia. "No SQL es realmente es no NoREL, es decir Base de Datos No-relacional"

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

MySQL es un sistema de base de datos relacional de código abierto que se basa en un lenguaje de consulta estructurado (SQL) y que puede ser ejecutado prácticamente en todas las plataformas, pero sobre todo en aquellas basadas en la web y la publicación en línea. Es un sistema muy utilizado por las empresas para promover datos, permitiendo a los desarrolladores y diseñadores web realizar cambios en los sitios de manera simple.

MANEJO DE CONCEPTOS

MariaDB es un sustituto de MySQL, con licencia GPL, en donde se incorporan todas las mejoras con más funcionalidades y un máximo rendimiento que permite modificar, almacenar y extraer información para servicios SQL sólidos y escalables. Fue desarrollado por Michael Widenius, fundador de MySQL y la comunidad de desarrolladores de software libre.

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

Las funciones de agregación se usan dentro de la cláusula SELECT en grupos de registros de devolver un único valor que se aplica a un grupo de registros.

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

XAMPP es un paquete de software libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script PHP y Perl.

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

MANEJO DE CONCEPTOS

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.

8. ¿Qué es DML y DDL?

DML es un lenguaje de programación que los **sistemas gestores de bases de datos** (Ejemplos: PostgreeSQL, Oracle y entre otros parecidos) implementan para que el usuario pueda realizar el **CRUD, consultas y demás acciones** con los datos/información almacenados en las bases de datos de estos sistemas gestores de bases de datos.

Es un lenguaje de programación que los sistemas gestores de bases de datos implementan para que el usuario pueda realizar el CRUD definiendo así la estructura de una base de datos donde se almacenarán los datos/información. También permite la implementación de procedimientos o funciones que permitan al usuario consultar dichos datos almacenados en la estructura anteriormente creada.

MANEJO DE CONCEPTOS

9. **¿Qué cosas características debe de tener una función? Explique sobre el nombre, el return, parámetros, etc.**
10. **¿Cómo crear, modificar y como eliminar una función?**

Para crear una función debemos de usar la sentencia CREATE FUNCTION. La sintaxis para crear una función es casi idéntica a la de crear un procedimiento, veamos:

CREATE FUNCTION nombre_función (parametro1,parametro2,...)
RETURNS tipoDato
[atributos de la rutina]
<bloque de instruccciones>

Para modificar una función usamos el comando ALTER FUNCTION. Con esta sentencia podemos cambiar los atributos de la función, pero no podremos cambiar el cuerpo. Veamos la sintaxis:

ALTER FUNCTION nombre_funcion
[SQL SECURITY {DEFINER|INVOKER}]
[COMMENT descripción]

MANEJO DE CONCEPTOS

Para eliminar una función usamos el comando DROP FUNCTION. Simplemente especificamos el nombre de la función y esta se borrará de la base de datos. Su sintaxis esta definida de la siguiente forma:
DROP FUNCTION nombre_funcion

PARTE PRACTICA

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

PARTE PRACTICA

12. Crear una consulta SQL en base al ejercicio anterior.

PARTE PRACTICA

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

```
CREATE DATABASE tareaHito2;
USE tareaHito2;

CREATE TABLE estudiantes
(
id_est INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
nombres VARCHAR(50),
apellidos VARCHAR(50),
edad INTEGER,
gestion INTEGER,
fono INTEGER,
email VARCHAR(100),
direccion VARCHAR(100),
sexo VARCHAR(10)
);
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');
```

PARTE PRACTICA

```
CREATE TABLE materias
(
id_mat INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
nombre_mat VARCHAR(100),
cod_mat VARCHAR(100)
);
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');

CREATE TABLE inscripcion
(
id_ins INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
id_est INT NOT NULL,
id_mat INT NOT NULL,
semestre VARCHAR(20),
gestion INTEGER,
FOREIGN KEY (id_est) REFERENCES estudiantes (id_est),
FOREIGN KEY (id_mat) REFERENCES materias (id_mat)
);
```

PARTE PRACTICA

```
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);

# Mostrar los nombres y apellidos de los estudiantes inscritos en la materia
# ARQ-105, adicionalmente mostrar el nombre de la materia.

SELECT est.id_est,
       est.nombres,
       est.apellidos,
       mat.nombre_mat,
       mat.cod_mat
FROM estudiantes AS est
   INNER JOIN inscripcion AS ins ON est.id_est = ins.id_est
   INNER JOIN materias AS mat ON ins.id_mat = mat.id_mat
WHERE mat.cod_mat = 'ARQ-105';
```

PARTE PRACTICA

```
#Deberá de crear una función que reciba dos parámetros y
#esta función deberá ser utilizada en la cláusula WHERE.

CREATE FUNCTION comparaMaterias(materia varchar(15), codMateria varchar(15))
  RETURNS BOOLEAN
  begin
      return materia = codMateria;
  end;
SELECT est.id_est,
       est.nombres,
       est.apellidos,
       mat.nombre_mat,
       mat.cod_mat
FROM estudiantes AS est
    INNER JOIN inscripcion AS ins ON est.id_est = ins.id_est
    INNER JOIN materias AS mat ON ins.id_mat = mat.id_mat
WHERE comparaMaterias(mat.cod_mat, 'ARQ-105');
```

PARTE PRACTICA

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.

```
CREATE FUNCTION avg_edad_estudiantes() RETURNS int
BEGIN
return
(
SELECT AVG(est.edad)
FROM estudiantes AS est
);
END;
SELECT avg_edad_estudiantes();
```

PARTE PRACTICA

15. Crear una función que permita concatenar 3 cadenas.

```
select CONCAT(est.nombres, ' ',est.apellidos, ' ', est.edad) as Persona
from estudiantes as est;
create function getNombreCompletoyEdad(nombre varchar(100),apellidos varchar(100),edad integer)
returns varchar(200)
begin
    declare nombreCompletoyEdad varchar(200);
    set nombreCompletoyEdad = CONCAT(nombre, ' ', apellidos, ' ',edad);
    return nombreCompletoyEdad;
end;
select getNombreCompletoyEdad(est.nombres, est.apellidos,est.edad) as persona
from estudiantes as est;
```

PARTE PRACTICA

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.

```
CREATE FUNCTION sumaEdades(genero varchar(10))
RETURNS INTEGER
begin
    return (
    select SUM(est.edad)
    from estudiantes AS est
    where est.sexo = genero
 );
end;

select est.nombres, est.apellidos
from estudiantes as est
where sumaEdades('femenino') % 2 = 0;

select sumaEdades('femenino');
```

PARTE PRACTICA

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).

```
create function myCompareFunction(nombres varchar(20), apellidos varchar(20), nombresAcomparar varchar(20), apellidosAcomparar varchar(20))
returns bool
begin
    declare respuesta bool default false;

    set respuesta = (nombres = nombresAcomparar AND apellidos=apellidosAcomparar);
    return respuesta;
end;

SELECT est.*
FROM estudiantes AS est
WHERE myCompareFunction(est.nombres, est.apellidos, 'Sandra', 'Mavir Uria')
```

GRACIAS