

DEFENSA HITO 2

BASE DE DATOS II

NOMBRE: Mijail Oliver Choque Amaro

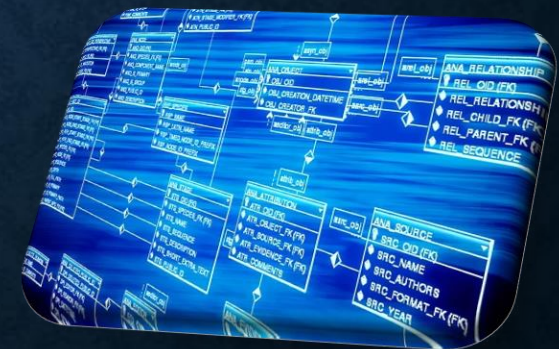
CARRERA: INGENIERIA DE SISTEMAS

CODIGO ESTUDANTIL: SIS12955851

UNIVERSIDAD: UNIFRANZ

DOCENTE: William Roddy Barra Paredes

FECHA DE ENTREGA: 24/10/22



MANEJO DE CONCEPTOS

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

R.- Es un conjunto de tablas ordenadas en filas(Registros) y columnas(campos)(, básicamente en tablas bidimensionales.

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

R.- Es una amplia clase de gestión de sistemas, caracterizado por no usar SQL, es capaz de almacenar grandes cantidades de datos, se enfoca en rendimiento mas que en consistencia.

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

R.- María DB 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.

4. ¿Qué son las funciones de agregación?

R.-Un función de agregación **realiza un cálculo sobre un conjunto de valores y devuelve un solo valor.**

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

R.-

- ❑ XAMPP es un paquete de software que incluye Apache (un servidor web), MySQL (un sistema de gestión de bases de datos), PHP (un lenguaje de programación utilizado para crear aplicaciones web) y Perl (un lenguaje de programación utilizado para la creación de scripts). XAMPP está disponible para Windows, Linux y MacOS.
- ❑ WAMP SERVER es un paquete de software similar a XAMPP, pero está diseñado específicamente para usuarios de Windows. WAMP SERVER incluye Apache, MySQL y PHP.
- ❑ LAMP es otro paquete de software similar a XAMPP y WAMP SERVER, pero está diseñado para usuarios de Linux. LAMP incluye Linux (un sistema operativo de código abierto), Apache, MySQL y PHP.

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

R.- Las funciones de agresión son funciones predefinidas y optimizadas por el DBMS, mientras que las funciones creadas por el DBA son personalizadas y específicas de la base de datos en particular.

7. ¿Para qué sirve el comando USE?

R.- Nos sirve para posicionarnos en una base de datos.

8. Que es DML y DDL?

R.- DML es Lenguaje de Manipulación de Datos y DDL es Lenguaje de Definición de Datos

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

R.-

- ❑ Nombre: nombre descriptivo y único para identificarla en el código.
- ❑ Parámetros: Las funciones deben aceptar uno o más parámetros. Los parámetros pueden ser opcionales o requeridos, y deben estar definidos en la declaración de la función.

- ❑ Return: Una función debe devolver un valor o un conjunto de valores.
- ❑ Retruns: En este apartado se indica el tipo de variable que debe retornar la función.
- ❑ Begin: Indica el inicio de una función.
- ❑ End: Indica el final de la función.
- ❑ Procedimiento: se indica lo que debe realizar una función, este se indica entre el Begin y el End

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

R.-

❑ CREAR:

```
CREATE OR REPLACE FUNCTION nombre_de_la_funcion( Parametros)
RETURNS tipo_de__variable_que_retorna
BEGIN
    DECLARE variable_a_retornar BOOLEAN DEFAULT FALSE;
    Indicaciones_de_lo_que_realiza_la_funcion
    RETURN variable_a_retornar ;
END;
```

❑ MODIFICAR:

```
ALTER FUNCTION nombre_de_la_funcion( Parametros)
RETURNS tipo_de__variable_que_retorna
BEGIN
    DECLARE variable_a_retornar BOOLEAN DEFAULT FALSE;
    Indicaciones_de_lo_que_se_va_a_modificar
    RETURN variable_a_retornar ;
END;
```

❑ ELIMINAR:

```
DROP FUNCTION nombre_de_la_funcion
```


PORTE PRACTICA

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

```
CREATE DATABASE POLLOS_COPA;  
USE POLLOS_COPA;
```

```
CREATE TABLE Cliente  
(  
    id_cliente INT AUTO_INCREMENT PRIMARY KEY,  
    fullname VARCHAR(100),  
    lastname VARCHAR(100),  
    edad INT,  
    domicilio VARCHAR(100)  
);
```

```
CREATE TABLE Pedido  
(  
    id_pedido INT AUTO_INCREMENT PRIMARY KEY,  
    articulo VARCHAR(100),  
    costo DOUBLE,  
    fecha DATE  
);
```

```
CREATE TABLE detalle_pedido  
(  
    id_detalle_pedido INT AUTO_INCREMENT PRIMARY KEY,  
    id_cliente INT,  
    id_pedido INT,  
  
    FOREIGN KEY (id_pedido) REFERENCES Pedido(id_pedido),  
    FOREIGN KEY (id_cliente) REFERENCES Cliente(id_cliente)  
);
```

```
INSERT INTO Cliente(fullname, lastname, edad, domicilio)  
VALUES('Juan Pérez', 'García', 30, 'Calle 1, Colonia A, Ciudad A, Estado A, 12345, País A'),  
('María González', 'Sánchez', 25, 'Calle 2, Colonia B, Ciudad B, Estado B, 67890, País B');
```

```
INSERT INTO Pedido(articulo, costo, fecha)  
VALUES ('Mouse inalámbrico', 25.99, '2023-03-28'),  
('Teclado USB', 39.99, '2023-03-29');
```

```
INSERT INTO detalle_pedido(id_cliente, id_pedido)  
VALUES (1,1),  
(2,2);
```


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

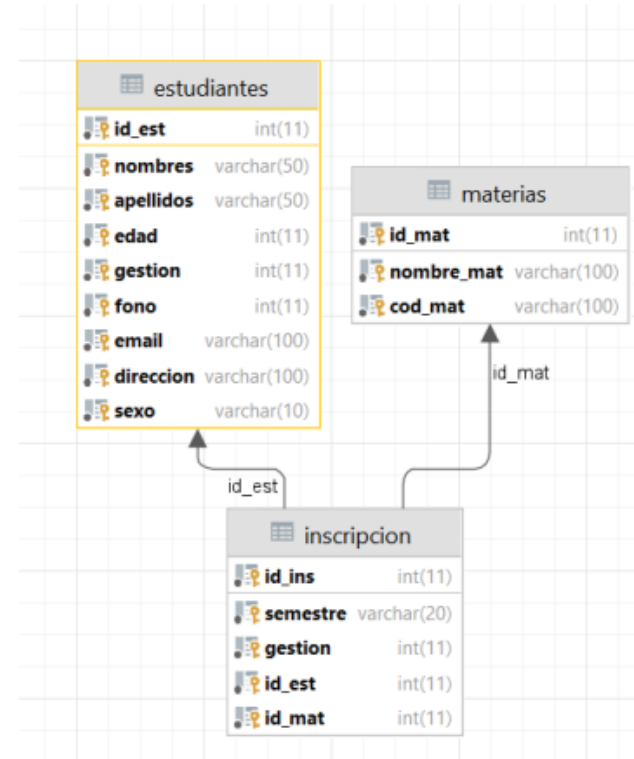
- Debe de utilizar las 3 tablas creadas anteriormente.
- Para relacionar las tablas utilizar JOINS.
- Adjuntar el código SQL generado.

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

- Recrear la siguiente base de datos:

```
SELECT P.*  
FROM Cliente AS C  
  INNER JOIN detalle_pedido AS dp on C.id_cliente = dp.id_cliente  
  INNER JOIN Pedido P on dp.id_pedido = P.id_pedido  
WHERE C.fullname LIKE '%Juan%';
```

```
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');
```

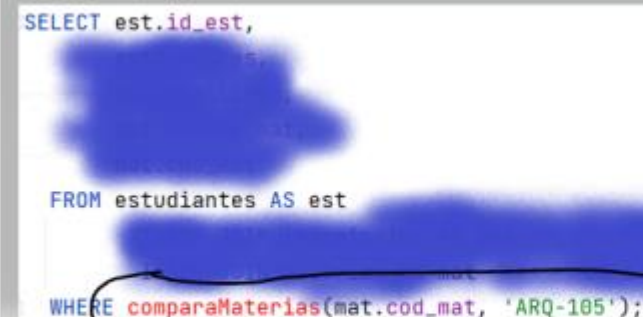
```
INSERT INTO materias (nombre_mat, cod_mat)
VALUES ('Introduccion a la Arquitectura', 'ARQ-101'),
('Urbanismo y Diseno', 'ARQ-102'),
('Dibujo y Pintura Arquitectonico', 'ARQ-103'),
('Matematica discreta', 'ARQ-104'),
('Fisica Basica', 'ARQ-105');
```

```
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.

○ Ejemplo:



```
SELECT est.id_est,
FROM estudiantes AS est
WHERE comparaMaterias(mat.cod_mat, 'ARQ-105');
```

○ El resultado al **ejecutar la consulta SQL** debería ser el siguiente:

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

```
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);
```

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

```
CREATE TABLE estudiantes
(
  id_est INT AUTO_INCREMENT PRIMARY KEY,
  nombres VARCHAR(50),
  apellidos VARCHAR(50),
  edad INT,
  gestion INT,
  fono INT,
  email VARCHAR(100),
  direccion VARCHAR(100),
  sexo VARCHAR(10)
);

CREATE TABLE materias
(
  id_mat INT AUTO_INCREMENT PRIMARY KEY,
  nombre_mat VARCHAR(100),
  cod_mat VARCHAR(100)
);

CREATE TABLE inscripcion
(
  id_ins INT AUTO_INCREMENT PRIMARY KEY ,
  semestre VARCHAR(20),
  gestion INT,
  id_est INT,
  id_mat INT,

  FOREIGN KEY (id_est) REFERENCES estudiantes(id_est),
  FOREIGN KEY (id_mat) REFERENCES materias(id_mat)
);
```

```
CREATE OR REPLACE FUNCTION comparaMaterias(cod_mat VARCHAR(100), cod_mat_buscar VARCHAR(100))
RETURNS BOOLEAN
BEGIN
    DECLARE A BOOLEAN DEFAULT FALSE;

    IF (cod_mat = cod_mat_buscar) THEN
        SET A = TRUE;
    END IF;
    RETURN A;
END;
```

```
SELECT E.id_est,
       E.nombres,
       E.apellidos,
       M.nombre_mat,
       M.cod_mat
FROM estudiantes AS E
     INNER JOIN inscripcion AS I on E.id_est = I.id_est
     INNER JOIN materias AS M on I.id_mat = M.id_mat
WHERE comparaMaterias(M.cod_mat,'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.

```
CREATE OR REPLACE FUNCTION promedioEdades(genero_buscar VARCHAR(10), cod_mat_buscar VARCHAR(100))
RETURNS INT
BEGIN
    DECLARE A INT DEFAULT 0;

    SELECT AVG(E.edad)
    FROM estudiantes AS E
        INNER JOIN inscripcion AS I on E.id_est = I.id_est
        INNER JOIN materias AS M on I.id_mat = M.id_mat
    WHERE E.sexo = genero_buscar AND M.cod_mat = cod_mat_buscar
    INTO A;

    RETURN A;
END;

SELECT promedioEdades('femenino','ARQ-104');

SELECT promedioEdades('masculino','ARQ-104');
```

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.

- Es decir podría mostrar el nombre, apellidos y la edad de los estudiantes.

```
CREATE OR REPLACE FUNCTION concatena3parametros(a VARCHAR(100), b VARCHAR(100), c INT)
  RETURNS VARCHAR(100)
  BEGIN
    DECLARE D VARCHAR(100);

    SELECT CONCAT('(',a,') ('',b,') ('',c,')')
    INTO D;

    RETURN D;
  END;

SELECT concatena3parametros('Pepito','pep',50);

SELECT concatena3parametros(E.nombres, E.apellidos,E.edad)
FROM estudiantes AS E;
```


16. Crear la siguiente VISTA:

- La vista deberá llamarse ARQUITECTURA_DIA_LIBRE
- El día viernes tendrán libre los estudiantes de la carrera de ARQUITECTURA debido a su aniversario
 - Este permiso es solo para aquellos estudiantes inscritos en el año 2021.
 - La vista deberá tener los siguientes campos.
 1. Nombres y apellidos concatenados = FULLNAME
 2. La edad del estudiante = EDAD
 3. El año de inscripción = GESTION
 4. Generar una columna de nombre DIA_LIBRE
 - a. Si tiene libre mostrar LIBRE
 - b. Caso contrario mostrar NO LIBRE.

```
CREATE VIEW ARQUITECTURA_DIA_LIBRE AS
SELECT CONCAT(E.nombres,' ',E.apellidos) AS FULLNAME,
       E.edad AS EDAD,
       I.gestion AS GESTION,
       (
        CASE
          WHEN I.gestion = 2021 AND M.cod_mat = 'ARQ-101' THEN 'LIBRE'
          ELSE 'NO LIBRE'
        END
      ) AS DIA_LIBRE
FROM estudiantes AS E
  INNER JOIN inscripcion AS I on E.id_est = I.id_est
  INNER JOIN materias AS M on I.id_mat = M.id_mat;

SELECT *
FROM ARQUITECTURA_DIA_LIBRE;
```

17. Crear la siguiente VISTA:

- Agregar una tabla cualquiera al modelo de base de datos.
- Después generar una vista que maneje las 4 tablas ■ La vista deberá llamarse PARALELO_DBA_I

```
CREATE TABLE Universidad
(
  id_universidad INT AUTO_INCREMENT PRIMARY KEY,
  nombre_uni VARCHAR(100),
  id_ins INT,

  FOREIGN KEY (id_ins) REFERENCES inscripcion(id_ins)
);

INSERT INTO Universidad(nombre_uni, id_ins)
VALUES ('UNIFRANZ',1),
       ('UNIFRANZ',2),
       ('UNIFRANZ',3),
       ('UNIFRANZ',4),
       ('UNIFRANZ',5),
       ('UNIFRANZ',6),
       ('UNIVALLE',7),
       ('UNIVALLE',8);

CREATE OR REPLACE VIEW PARALELO_DBA_I AS
SELECT CONCAT(E.nombres,' ',E.apellidos) AS FULLNAME,
       E.edad AS EDAD,
       I.gestion AS GESTION,
       U.nombre_uni AS NOMBRE_UNIVESIDAD
FROM estudiantes AS E
  INNER JOIN inscripcion AS I on E.id_est = I.id_est
  INNER JOIN materias AS M on I.id_mat = M.id_mat
  INNER JOIN Universidad U on I.id_ins = U.id_ins;

SELECT *
FROM PARALELO_DBA_I
WHERE NOMBRE_UNIVESIDAD = 'UNIVALLE';
```


THANKS!