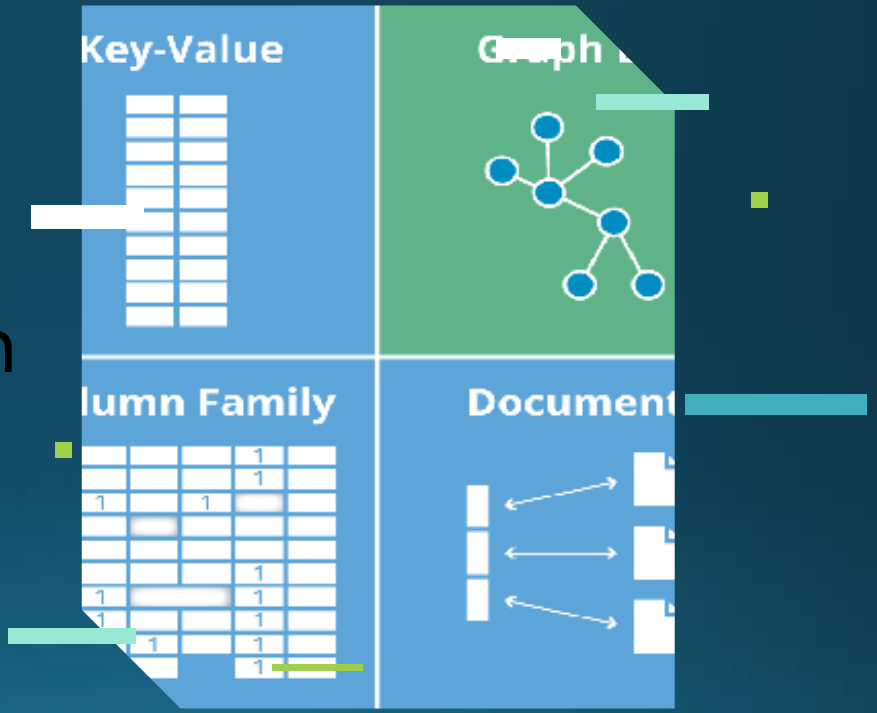
BASE DE DATOS II

Joel Reynaldo Condori Tumiri

Manejo de conceptos

1¿BASE DE DATOS RELACIONALES?

Base de datos relacionales: Es un conjunto de tablas formada por filas, registros, columnas y campos



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

Las bases de datos NoSQL están diseñadas específicamente para modelos de datos específicos y tienen esquemas flexibles para crear aplicaciones modernas. Las bases de datos NoSQL son ampliamente reconocidas porque son fáciles de desarrollar, por su funcionalidad y el rendimiento a escala.

¿Que es MySQL y MariaDB? Explique las diferencias o son iguales

MySQL es un sistema de gestión de bases de datos relacionales de código abierto respaldado por Oracle y basado en el lenguaje de consulta estructurado (SQL). MySQL funciona prácticamente en todas las plataformas, incluyendo Linux, UNIX y Windows. Aunque puede utilizarse en una amplia gama de aplicaciones, MySQL se asocia más a menudo con las aplicaciones web y la publicación en línea.

¿Que son las funciones de agregación?

MariaDB es un sistema de gestión de bases de datos que está muy relacionado con MySQL, ya que fue desarrollado por uno de los desarrolladores, Michael “Monty” Widenius. El objetivo de su desarrollo fue el de mantener el software de gestión de base de datos en un modelo de software libre.

FUNCIONES DE AGREGACIÓN

AVG

```
select avg(jug.edad)
from jugador AS JUG
```

La media de valores.

COUNT

```
select COUNT(jug.id_jugador)
from jugador AS JUG
```

El numero de filas.

MAX

```
select max(jug.edad)
from jugador AS JUG
```

El valor mas grande.

MIN

```
select min(jug.edad)
from jugador AS JUG
```

El valor mas pequeño.

SUM

```
select sum(jug.edad)
from jugador AS JUG
```

La suma de los valores

¿Qué llegaría ser XAMPP?

XAMPP es básicamente un paquete que ayuda a instalar todo lo necesario para poner en marcha un servidor web con todo lo que necesita para funcionar. En concreto, el software de servidor web Apache, el software de de base de datos MariaDB, el lenguaje de desarrollo web PHP y PERL, un lenguaje de programación dinámico.

WAMP SERVER

Se trata de una plataforma de desarrollo web que utiliza Linux como sistema operativo, Apache como servidor web, MySQL como sistema de gestión de bases de datos relacionales y PHP como lenguaje de script orientado a objetos.

LAMP

LAMP es el acrónimo usado para describir un sistema de infraestructura de internet que usa las siguientes herramientas:Linux, el sistema operativo; En algunos casos también se refiere a LDAP. Apache, el servidor web; MySQL/MariaDB, el gestor de bases de datos; PHP, el lenguaje de programación.

DIFERENCIA ENTRE FUNCIONES DE AGREGACION Y FUNCIONES POR EL USUARIOS

Las funciones de agregación en SQL nos permiten efectuar operaciones sobre un conjunto de resultados, pero devolviendo un único valor agregado para todos ellos. Es decir, nos permiten obtener medias, máximos, sobre un conjunto de valores. Funciones creados por el BDA se utiliza para definir una función de tabla, fila o escalar de SQL definida por el usuario. Una función escalar devuelve un solo valor cada vez que se invoca y en general es válida cuando una expresión SQL es válida.

USE

El comando USE DATABASE se utiliza para designar una base externa como base de datos actual, en otras palabras, la base a la cual se dirigirán las próximas consultas SQL en el proceso actual.

DML

Las sentencias DML se utilizan para controlar información contenida en la base de datos

-INSERT

```
insert into campeonato values ('camp-111','Campeonato Unifranz','El Alto')
insert into campeonato values ('camp-222','Campeonato Unifranz','Cochabamba')
```

Insertar registros a una tabla

-UPDATE

```
update campeonato set sede = 'El alto'
where id_campeonato = 'camp-111'
```

Modificación de la información de una tabla

-DELETE

```
delete from jugador
where id_jugador = 'jug-333'
```

Eliminar registros de una tabla

DDL

Esta formado por un conjunto de sentencias llamadas ddl que nos sirve para la creación de una base de datos y todos sus componentes

-CREATE

```
Create database Unifranzitos
use Unifranzitos
```

Crea base de datos y nos permite crear tablas

-DROP

```
drop table jugador
```

```
drop  database Unifranzitos
```

Drop table borra la tabla

-ALTER

```
alter table jugador
add
seleccion varchar (12)
```

Modifica la estructura de una tabla

-TRUNCATE

```
truncate table jugador
```

Truncate vacía la tabla

CARACTERISTICAS FUNCIONES

Parámetros de entrada

```
CREATE FUNCTION contar_productos(gama VARCHAR(50))
```

Parámetros de salida

```
DELIMITER $$
DROP FUNCTION IF EXISTS contar_productos$$
CREATE FUNCTION contar_productos(gama VARCHAR(50))
  RETURNS INT UNSIGNED
  ...
BEGIN
  ...

  RETURN total;
END
```

Variables

```
DECLARE var_name [, var_name] ... type [DEFAULT value]
```

COMANDOS FUNCIONES

Crear función

```
create function compara_materias(cod_mat varchar(20), nombre_mat varchar(20))
```

Modificar

```
create or replace function get_genero_edad(genero varchar(10), edad int)
    returns boolean
```

Eliminar función

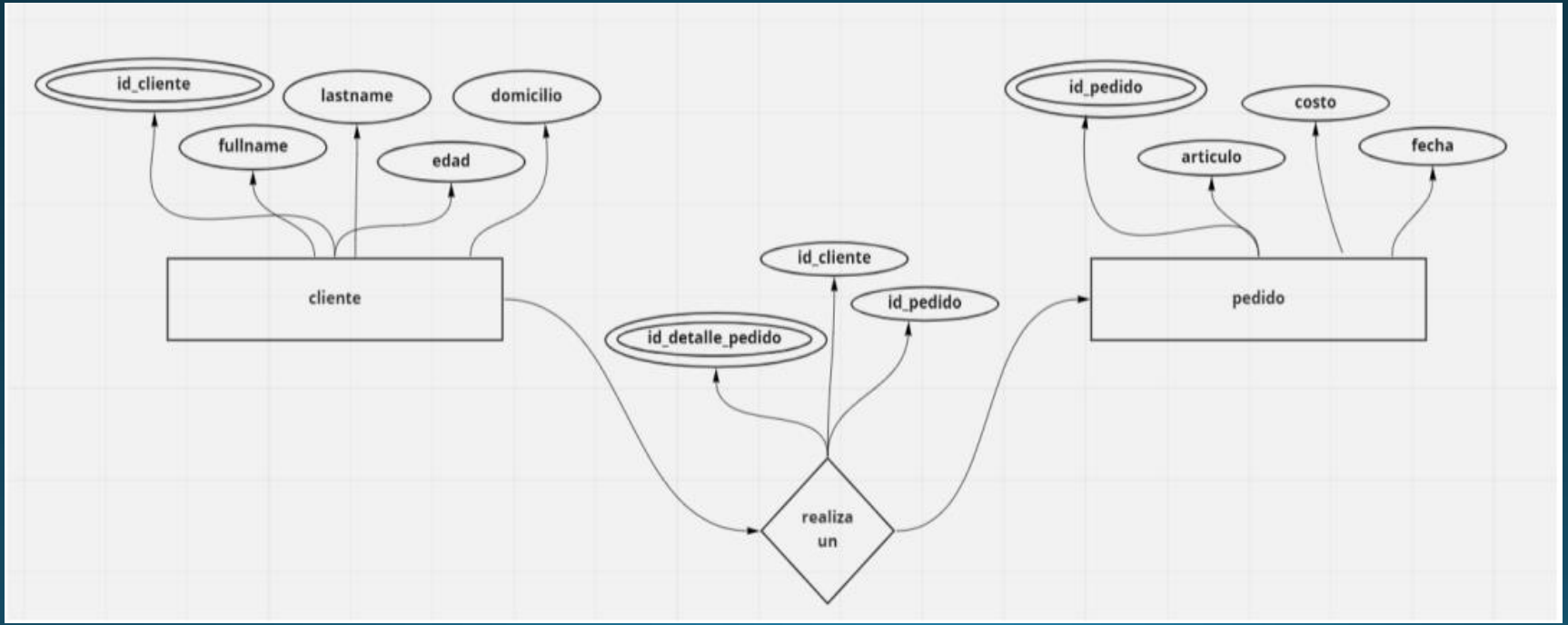
```
drop function comparaNombre;
```

PARTE PRACTICA

01

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

MODELO E-R



POLLOS COPA

```sql
create database pollos_copa;
use pollos_copa;

create table cliente
(
id_cliente int auto_increment primary key not null,
fullname varchar(50),
lastname varchar(50),
edad int,
domicilio varchar(50)
);

create table pedido
(
id_pedido int auto_increment primary key not null,
articulo varchar(50) not null,
costo float(50) not null,
fecha date
);

create table detalle_pedido
(
id_detalle_pedido int 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)
);
```

CREAR CONSULTAS SQL EN BASE AL EJERCICIO ANTERIOR

CLIENTE

```sql
insert into cliente(fullname, lastname, edad, domicilio)
values('joel','tumiri',23,'ceja'),
      ('roberto','mamani',23,'av buenos aires');
```

PEDIDO

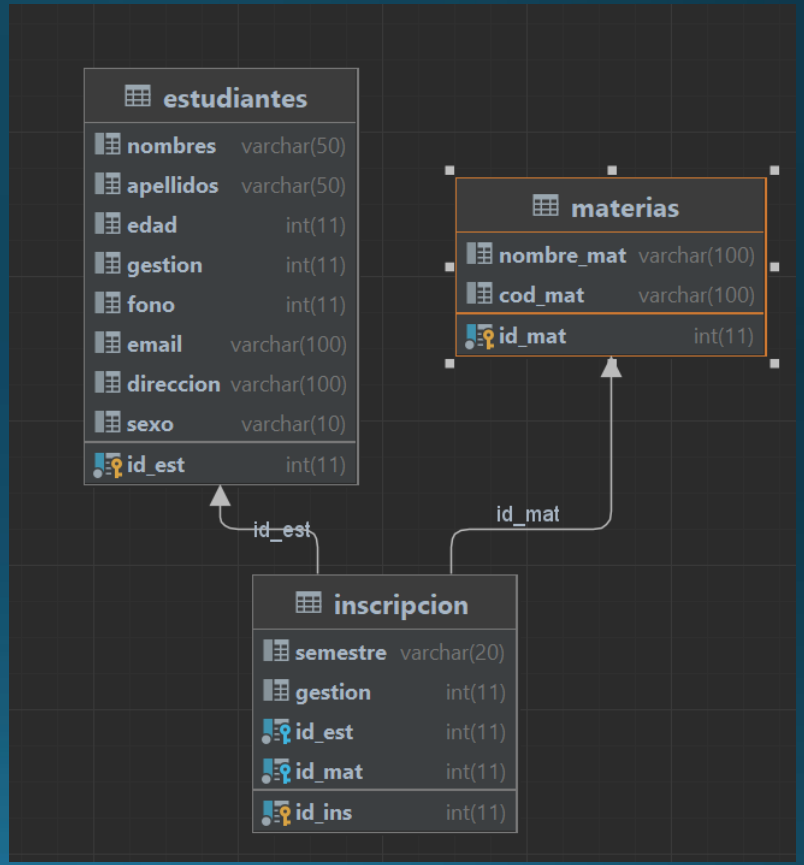
```sql
insert into pedido(articulo, costo, fecha)
values('broaster',22.0,'2023-05-02'),
      ('milanesa',23.0,'2023-05-02');
```

DETALLE PEDIDO

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

```sql
-- cliente que realizo un pedido el 2023
select cli.fullname,ped.costo
from cliente as cli
inner join detalle_pedido as de on cli.id_cliente = de.id_cliente
inner join pedido ped on de.id_cliente = ped.id_pedido
where ped.fecha like ('2023%');
```

MODELO E-R


estudiantes
nombres varchar(50)
apellidos varchar(50)
edad int(11)
gestion int(11)
fono int(11)
email varchar(100)
direccion varchar(100)
sexo varchar(10)
id_est int(11)
materias
nombre_mat varchar(100)
cod_mat varchar(100)
id_mat int(11)
id_est
id_mat
inscripcion
semestre varchar(20)
gestion int(11)
id_est int(11)
id_mat int(11)
id_ins int(11)

CODIGO SQL

TAREA HITO 2

```sql
create database tareaHito2;
use tareaHito2;

create table estudiantes
(
    id_est    int auto_increment primary key not null,
    nombres   varchar(50),
    apellidos varchar(50),
    edad      int(11),
    gestion   int(11),
    fono      int(11),
    email     varchar(100),
    direccion varchar(100),
    sexo      varchar(10)
);

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

create table inscripcion
(
    id_ins   int auto_increment primary key not null,
    semestre varchar(20),
    gestion  int(11),

    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)
);
```

REGISTROS

ESTUDIANTES

```
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');
```

MATERIA

```
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');
```

INSCRIPCION

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

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

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

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

```sql
-- creando funcion
create or replace function compara_materias(cod_mat varchar(20), nombre_mat varchar(20))
returns boolean
begin
    declare respuesta boolean;
    if cod_mat = nombre_mat
    then
        set respuesta = 50;
        end if;
    return respuesta;
end;

select est.nombres, est.apellidos, mat.cod_mat, mat.nombre_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 compara_materias( cod_mat: mat.cod_mat,  nombre_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 prom_edades(cod_mat varchar(20), sexo varchar(20))
returns int
begin
    declare avgEDAD int default 30;
    select avg(est.edad) into avgEDAD
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 est.sexo = sexo and mat.cod_mat = cod_mat;
    return avgEDAD;
end;

select prom_edades( cod_mat: 'ARQ-104',  sexo: 'femenino') AS promedio;
```

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 concaten(pep1 varchar(20), pep2 varchar(20), pep3 varchar(20))
returns varchar(60)
begin
    declare resultado varchar(60);
    set resultado = concat(pep1, ' ', pep2, ' ', pep3);
    return resultado;
end;

select concaten( pep1: 'pepito', pep2: 'pep', pep3: '50');
```

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(**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**).

```sql
create or replace function get_genero_edad(genero varchar(20), edad int)
returns boolean
begin
    declare resultado int default 0;
    declare ifRes boolean;

    select sum(est.edad) into resultado
    from estudiantes as est
    where est.sexo = genero;

    if resultado %2 = 0 and resultado > edad
        then
        set ifres = 1;
    end if;
    return ifRes;
end;

select est.nombres, est.apellidos, i.semestre
from  estudiantes as est
inner join inscripcion i on est.id_est = i.id_est
where get_genero_edad( genero: 'masculino',  edad: 22);
```

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.

- Ejemplo:

```
create function busca_nombres_apellidos(
    est.nombres,
    'William',
    est.apellidos,
    'Barra Paredes'
) RETURNS ....
```

- 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 comparaNombre(nombre varchar(50), apellido varchar(50), nombreEst varchar(50), apellidoEst varchar(50))
returns boolean
begin
    declare resultado boolean;
    if nombre = nombreEst and apellido = apellidoEst
        then
        set resultado =1;
    end if;
    return resultado;
end;

select est.*
from estudiantes as est
where comparaNombre( nombre: est.nombres,  apellido: est.apellidos,  nombreEst: 'Joel',  apellidoEst: 'Adubiri Mondar')
```