




# BASE DE DATOS II

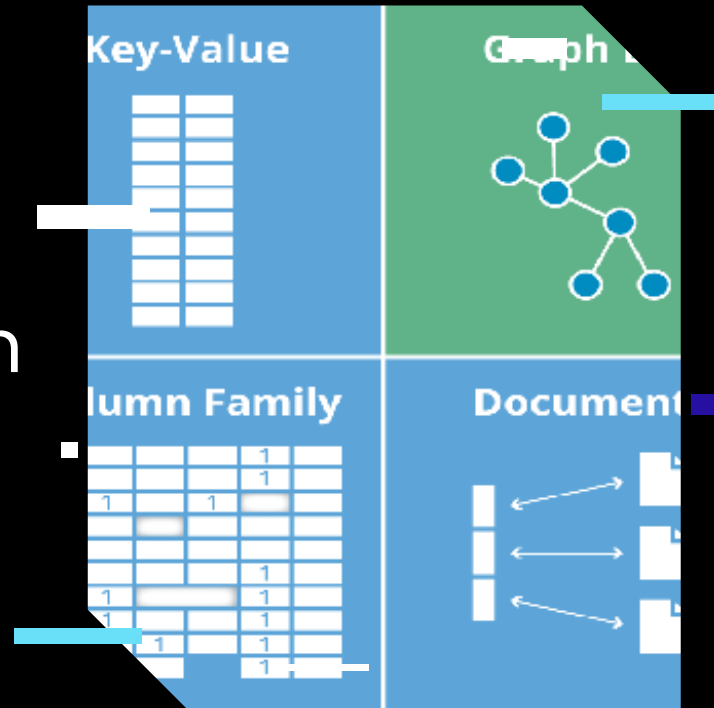
FREDDY MACHACA MAMANI

The background is black with several white lines of varying lengths and orientations. A small cyan square is located in the upper right quadrant. The text '01' is rendered in a large, stylized, cyan font with a blocky, digital appearance. The '0' has a vertical line through its center, and the '1' is a solid block. The text 'PARTE TEORICA' is in a bold, white, sans-serif font, positioned to the right of the '01'.

# PARTE TEORICA

# BASE DE DATOS RELACIONALES

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



# NOSQL

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.



# MYSQL

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.



# MARIADB

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

# 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 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.



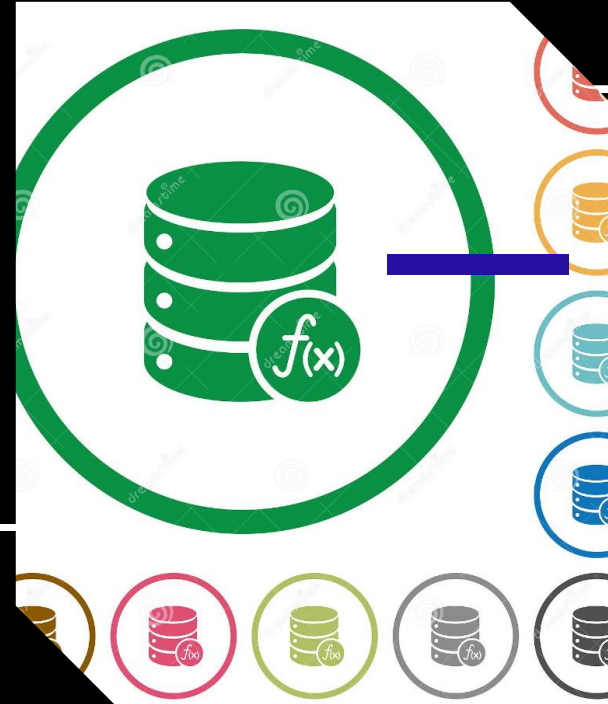
# LAMP

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.



# 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;
```





02

# PARTE PRACTICA

# CREANDO UNA BASE DE DATOS

01

BASE DE DATOS POLLOS COPA

ANALISE DE COSAS QUE DEBERIA TENER

02

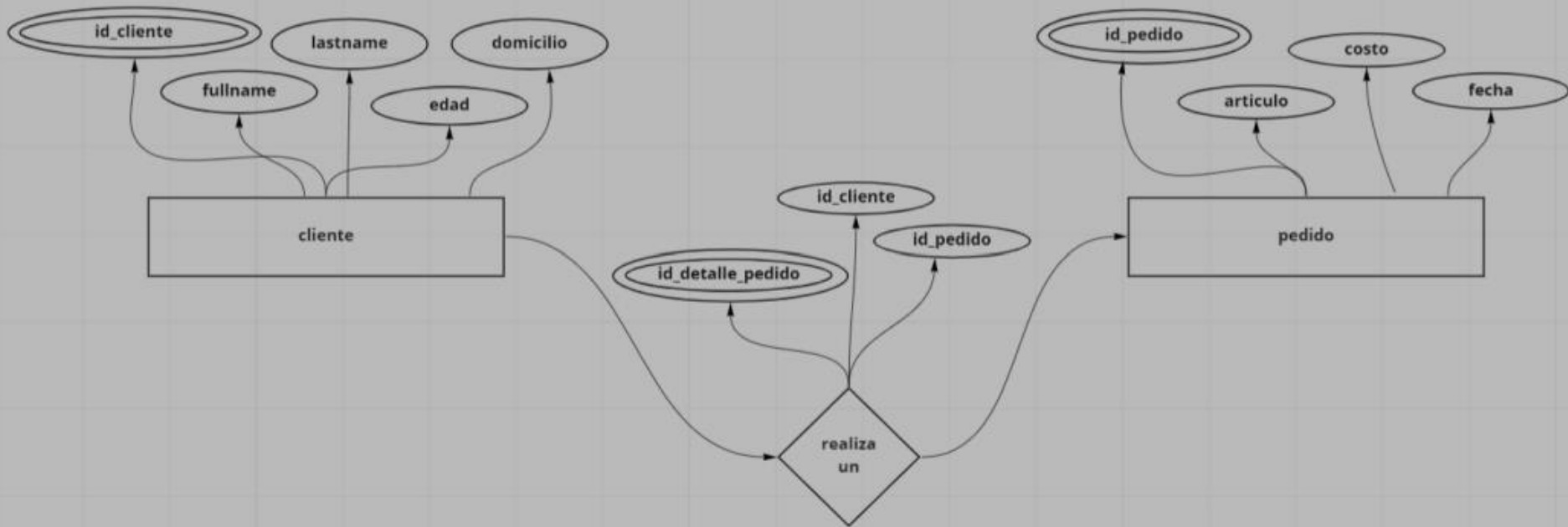
BASE DE DATOS TAREA HITO 2

SOLUCION DE CONSULTAS

01

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

# MODELO E-R



01

# POLLOS COPA

```
create database POLLOS_COPA;  
use POLLOS_COPA;
```

```
CREATE TABLE cliente  
(  
  id_cliente int auto_increment primary key not null,  
  fullname VARCHAR(30),  
  lastname VARCHAR(30),  
  edad INTEGER,  
  domicilio VARCHAR(40)  
);
```

```
CREATE or replace TABLE pedido  
(  
  id_pedido int auto_increment primary key not null,  
  articulo VARCHAR(30),  
  costo FLOAT,  
  fecha DATE  
);
```

```
CREATE TABLE detallePedido  
(  
  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)  
);
```

# 01

## REGISTROS

CLIENTE

```
insert into cliente (fullname, lastname, edad, domicilio)
values ('Daniel', 'Orivaldo', 19, 'Rua tiradentes'),
       ('Gustavo', 'Zepulvida', 20, 'Rua maranaho');
```

PEDIDO

```
insert into pedido (articulo, costo, fecha)
values ('Celular', 1920, '2020-04-02'),
       ('Grafica', 3000, '2019-06-10');
```

DETALLE PEDIDO

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

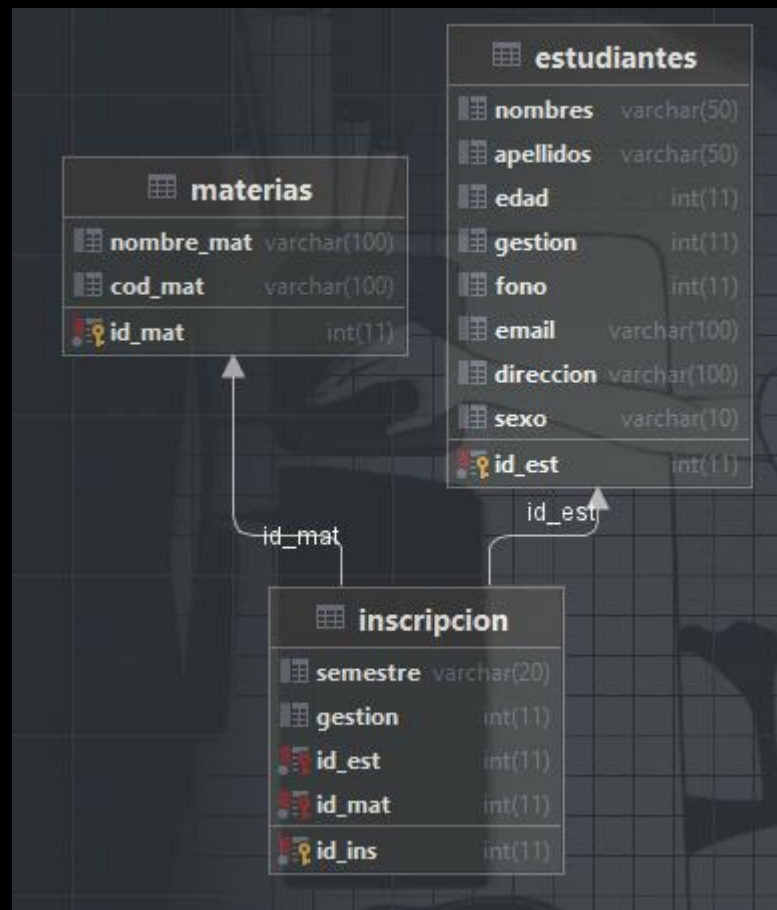
## CONSULTA

```
#Que cliente realizo un pedido en 2020
select cle.fullname, p.fecha
from cliente as cle
inner join detallepedido as d on cle.id_cliente = d.id_cliente
inner join pedido p on d.id_pedido = p.id_pedido
where p.fecha like ('2020%')
```

02

# MODELO E-R

## TAREA HITO 2



02

# CODIGO SQL

## TAREA HITO 2

```
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)
);
```

# 02

## ESTUDIANTES

# REGISTROS MATERIA TAREA HITO 2

## INSCRIPCION

```
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', 'Aubiri 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 Arquitectonica', 'ARQ-103'),
('Matematica discreta', 'ARQ-104'),
('Fisica Basica', 'ARQ-105');
```

```
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),
```



# 02

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

```
62 ✓ select est.nombres, est.apellidos, mat.cod_mat, mat.nombre_mat
63 from estudiantes as est
64     inner join inscripcion as ins on est.id_est = ins.id_est
65     inner join materias as mat on ins.id_mat = mat.id_mat
66 where mat.cod_mat = 'ARQ-105';
```

Output Result 1 ×

< < 1 row > > |

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

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

```
71 create function compara_materias(cod_mat varchar(20), nombre_mat varchar(20))
72 returns boolean
73 begin
74     declare respuesta boolean;
75     if cod_mat = nombre_mat
76     then
77         set respuesta=1;
78     end if;
79     return respuesta;
80 end;
81
82 #clausula where en la consulta
83 ✓ select est.nombres, est.apellidos, mat.cod_mat, mat.nombre_mat
84 from estudiantes as est
85     inner join inscripcion as ins on est.id_est = ins.id_est
86     inner join materias as mat on ins.id_mat = mat.id_mat
87 where compara_materias( cod_mat: mat.cod_mat, nombre_mat: 'ARQ-105');
```

Output Result 2 ×

< < 1 row > > |

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

#### 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**.

```
102 CREATE OR REPLACE FUNCTION get_avg_est(genero varchar(10), codMateria varchar(10))
103 RETURNS INTEGER
104 BEGIN
105     declare avgEdad int default 0;
106     SELECT avg(est.edad) into avgEdad
107     FROM estudiantes AS est
108         inner join inscripcion ins on est.id_est = ins.id_est
109         inner join materias mat on ins.id_mat = mat.id_mat
110     where est.sexo = genero and mat.cod_mat = codMateria;
111     return avgEdad;
112 END;
113
114
115 ✓ select get_avg_est( genero: 'femenino', codMateria: 'ARQ-104') as promedio;
116
```

Output promedio:int(11) ×

1 row

| promedio |
|----------|
| 23       |

# 02

## 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.

```
125 create function getParametros(par1 varchar(20), par2 varchar(20), par3 varchar(20))
126     returns varchar(60)
127 begin
128     declare resultado varchar(60);
129     set resultado = CONCAT(par1, ' ', par2, ' ', par3);
130     return resultado;
131 end;
132
133 ✓ select getParametros( par1: 'pepito', par2: 'pep', par3: '50');
```

getParametros()

Output getParametros('pepit...p', '50'):varchar(60) ×

1 row

``getParametros('pepito', 'pep', '50')``

1 pepito pep 50

# 02

## 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**).

```
149 create or replace function get_genero_edad(genero varchar(10), edad int)
150     returns boolean
151 begin
152     declare resultado int default 0;
153     declare ifRes boolean;
154
155     select sum(est.edad) into resultado
156     from estudiantes as est
157     where est.sexo=genero;
158
159     if resultado%2=0 and resultado>edad
160     then
161         set ifRes=1;
162     end if;
163     return ifRes;
164 end;
165
166 ✓ select est.nombres, est.apellidos, i.semestre
167 from estudiantes as est
168 inner join inscripcion i on est.id_est = i.id_est
169 where get_genero_edad( genero: 'masculino', edad: 22);
170
```

Output Result 5

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 |

# 02

## 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.

```
179 create or replace function comparaNombre(nombre varchar(50),apellido varchar(50),nombreEst varchar(50), apellidoEst varchar(50))  
180 returns boolean  
181 begin  
182     declare resultado boolean;  
183     if nombre=nombreEst and apellido=apellidoEst  
184     then  
185         set resultado=1;  
186     end if;  
187     return resultado;  
188 end;  
189  
190 ✓ select est.*  
191 from estudiantes as est  
192 where comparaNombre( nombre: est.nombres, apellido: est.apellidos, nombreEst: 'Joel', apellidoEst: 'Adubiri Mondar')  
193
```

Output tareahito2.estudiantes x

< 1 row > Txc Auto DDL

|   | id_est | nombres | apellidos      | edad | gestion | fono    | email          | direccion       | sexo      |
|---|--------|---------|----------------|------|---------|---------|----------------|-----------------|-----------|
| 1 | 3      | Joel    | Adubiri Mondar | 30   | <null>  | 2832117 | joel@gmail.com | Av. 6 de Agosto | masculino |