



/TAREA HITO 2

BASE DE DATOS II

RONALD URIEL CHOQUE PACO
SIS6972733



EL ALTO, ABRIL DE 2022





/CONTENIDO (DBAII).



/01

/PARTE TEORICA



Se explicara
conceptos
elementales de
introduccion a
DBAII.



/02

/PARTE PRACTICA



Se presentara la
aplicacion de la
parte teorica para
la resolucion de
requerimientos.





/START!

> DBAII





/01 /PARTE TEORICA





/CONCEPTOS



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

Las BDA relacionales funcionan a base de SQL manejando tablas bidimensionales (con filas y columnas), sus campos de relación deben ser del mismo tipo.



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

Las DBA no relacionales refieren a que SQL no es su principal lenguaje, este tipo de base esta dedicada al rendimiento gracias a que puede almacenar grandes cantidades de datos, los cuales se van expandiendo.



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

MySQL es un gestor de BDA relacionales de doble licencia (código abierto/licencia ORACLE).

MariaDB es un gestor de DBA casi idéntico a MySQL, con la diferencia que es software libre puro.





/CONCEPTOS



**¿A que se refiere cuando
¿Qué son las funciones de
agregación?**

Son aquellas que funcionan bajo la clausula SELECT, aplicado a un grupo de registros y que devuelven un único valor.



**¿Qué llegaría a ser
XAMPP?**

Es una distribución de APACHE (HTTPS web gratuito) que concatena varios softwares libres (LINUX, APACHE, MYSQL/MARIADB, PHP, PERL); dedicado a administrar bases de datos.



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

(vea la pregunta 4)
Una función CUSTOM esta definida por el comando CREATE FUNCTION, su sintaxis y uso esta completamente definida por el USER.





/CONCEPTOS



¿Para qué sirve el comando USE?

Define la DBA en la que se va a trabajar.

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



¿Que es DML y DDL?

DDL es DATA DEFINITION LANGUAGE, todo lo relacionado con el diseño y creaciones de tablas.

DML es DATA MANIPULATION LANGUAGE, todo lo relacionado a la manipulación de tablas.



¿Qué cosas características debe de tener una función?

Las mas elementales:

```
CREATE OR REPLACE FUNCTION est_mat(cod_mat VARCHAR(50))  
RETURNS INT  
BEGIN  
    DECLARE parametro INT DEFAULT 0;  
    SELECT est.id_est INTO parametro  
    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 = cod_mat;  
    RETURN parametro;  
end;
```





/CONCEPTOS



→ CREA:

```
CREATE FUNCTION nombre()
```

→ MODIFICA:

```
CREATE OR REPLACE FUNCTION nombre()
```



¿Cómo crear, modificar y cómo eliminar una función?

Requerimos de los siguientes comandos:

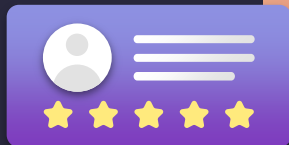
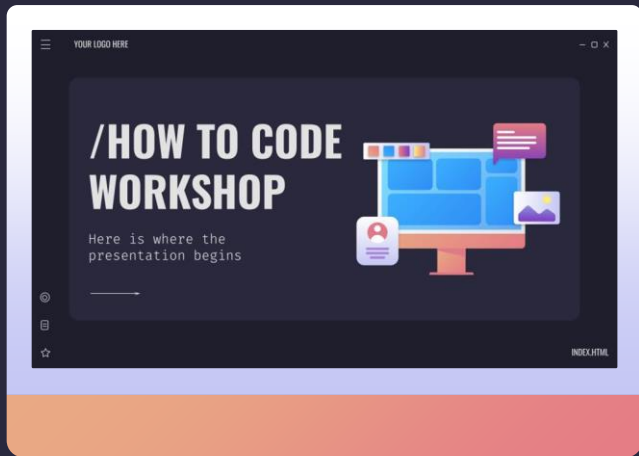
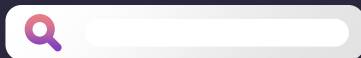
→ ELIMINA:

```
DROP FUNCTION comparacion;
```

→ ELIMINA SI EXISTE:

```
DROP FUNCTION IF EXISTS comparacion;
```

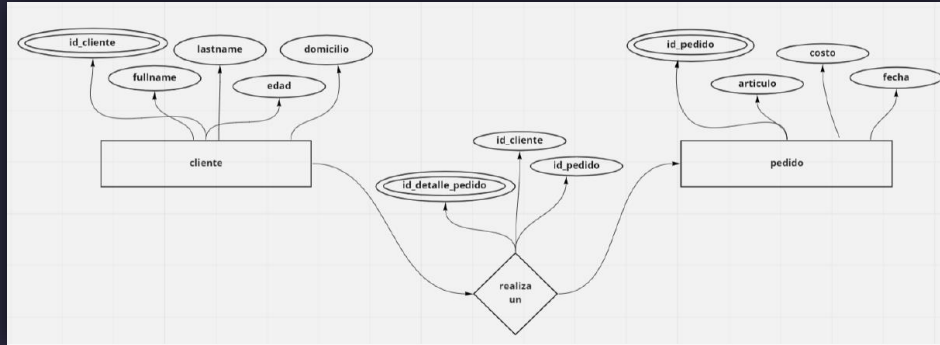




/02 /PARTE PRACTICA



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




/SUGERENCIAS

Se sugiere crear una base de datos de nombre POLLOS_COPA y en ella crear las tablas:


- cliente
- detalle_pedido
- pedido



/CREACION DE LA DBA.



```
CREATE DATABASE pollos_copa;
```



```
USE pollos_copa;
```

El comando CREATE TABLE es el que genera la DBA.

Comando USE para trabajar en el mismo.





/CREACION DE LAS TABLAS.



```
CREATE TABLE cliente
(
    id_cliente INTEGER PRIMARY KEY NOT NULL,
    fullname VARCHAR(100) NOT NULL,
    lastname VARCHAR(100) NOT NULL,
    edad INTEGER NOT NULL,
    domicilio VARCHAR(100) NOT NULL
);
```



```
CREATE TABLE pedido
(
    id_pedido INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
    articulo VARCHAR(100) NOT NULL,
    costo INTEGER NOT NULL,
    fecha DATE NOT NULL
);
```



```
CREATE TABLE detalle_pedido
(
    id_detalle_pedido INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
    id_pedido INTEGER NOT NULL,
    id_cliente INTEGER NOT NULL,
    FOREIGN KEY (id_cliente) REFERENCES cliente (id_cliente),
    FOREIGN KEY (id_pedido) REFERENCES pedido (id_pedido)
);
```



Definimos a los PRIMARY KEY.

El comando FOREIGN KEY es utilizado para relacionar las tablas mediante los PRIMARY KEY.

NOT NULL para no tener columnas sin registro.





/LLENADO DE LAS TABLAS.



```
INSERT INTO cliente (id_cliente, fullname, lastname, edad, domicilio) VALUES  
(6972733, 'Ronald', 'Choque', 19, 'Av. Illimani #3004'),  
(1_234_567, 'Daniel', 'Ruiz', 21, 'Av. Estructuradamente #123');
```

```
INSERT INTO pedido (articulo, costo, fecha) VALUES  
( 'Balde 8 presas', 90, '2022-3-29'),  
( 'Combo Aniversario', 45, '2022-3-30');
```

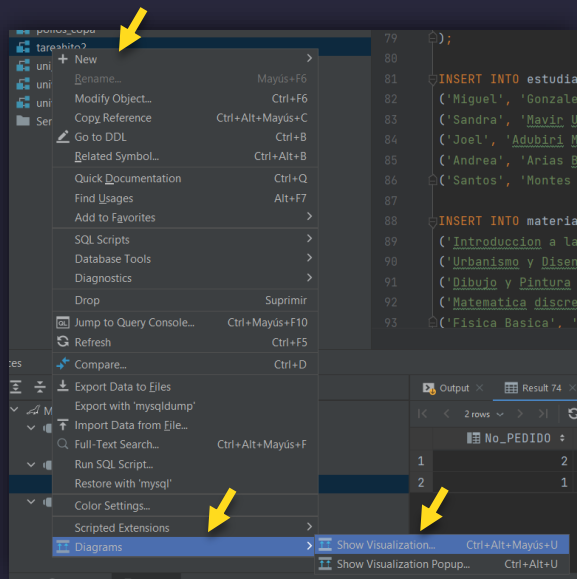
```
INSERT INTO detalle_pedido (id_pedido, id_cliente) VALUES  
(1, 6972733),  
(2, 1234567);
```

El llenado de la tabla puede acortarse mediante el uso de “,” al final de una cadena de valores, cortando el proceso con “;”

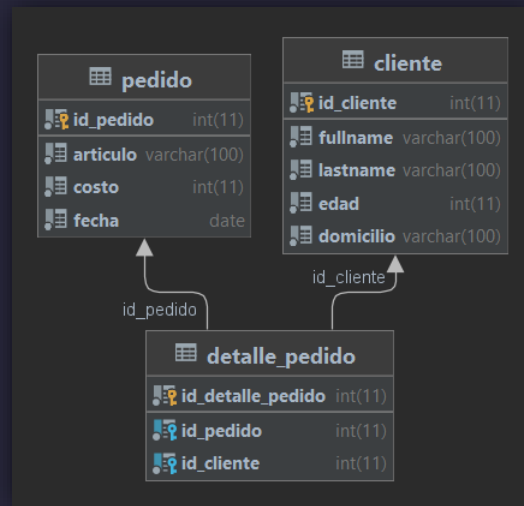
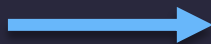
Los INSERT fueron realizados de forma autónoma (no especificado en el doc.).



/DIAGRAMA LOGICO DEL PROYECTO.



Haciendo un clic derecho sobre la carpeta de tablas, vaya y seleccione DIAGRAMS y la opción SHOW VISUALIZATION para poder ver el diagrama.





/Crear una consulta SQL en base al ejercicio anterior.

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

```
SELECT det.id_pedido AS No_PEDIDO, det.id_cliente AS CI, cli.domicilio  
FROM detalle_pedido AS det  
INNER JOIN cliente AS cli ON det.id_cliente = cli.id_cliente  
INNER JOIN pedido AS pe ON det.id_pedido = pe.id_pedido;
```



CONSULTA: Se desea ver el n° de pedido, CI del cliente y su dirección para que se haga llegar su pedido:

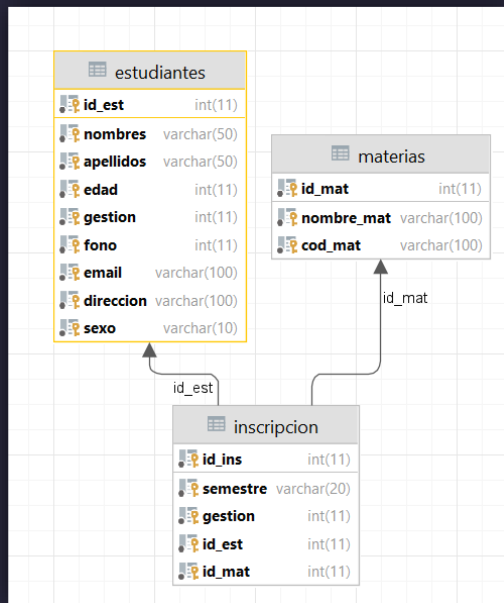
EJECUCION:

	No_PEDIDO	CI	domicilio
1	2	1234567	Av. Estructuramente #123
2	1	6972733	Av. Illimani #3004




/Crear un función que compare dos códigos de materia.

o Recrear la siguiente base de datos:






/CREACION DE LA DBA.



```
CREATE DATABASE tareaHito2;
```



```
USE tareaHito2;
```

El comando CREATE TABLE es el que genera la DBA.

Comando USE para trabajar en el mismo.





/CREACION DE LAS TABLAS.



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



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



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



Definimos a los PRIMARY KEY.

El comando FOREIGN KEY es utilizado para relacionar las tablas mediante los PRIMARY KEY.

NOT NULL para no tener columnas sin registro.

AUTO_INCREMENT para generar automáticamente el registro de la columna.





/LLENADO DE LAS TABLAS.



→

```
INSERT INTO estudiantes (nombres, apellidos, edad, fono, email, direccion, sexo) VALUES
('Miguel', 'Gonzales Veliz', 20, 2_832_115, 'miguel@gmail.com', 'Av. 6 de Agosto', 'masculino'),
('Sandra', 'Mavir Uria', 25, 2_832_116, 'sandra@gmail.com', 'Av. 6 de Agosto', 'femenino'),
('Joel', 'Adubiri Mondar', 30, 2_832_117, 'joel@gmail.com', 'Av. 6 de Agosto', 'masculino'),
('Andrea', 'Arias Ballesteros', 21, 2_832_118, 'andrea@gmail.com', 'Av. 6 de Agosto', 'femenino'),
('Santos', 'Montes Valenzuela', 24, 2_832_119, 'santos@gmail.com', 'Av. 6 de Agosto', 'masculino');
```



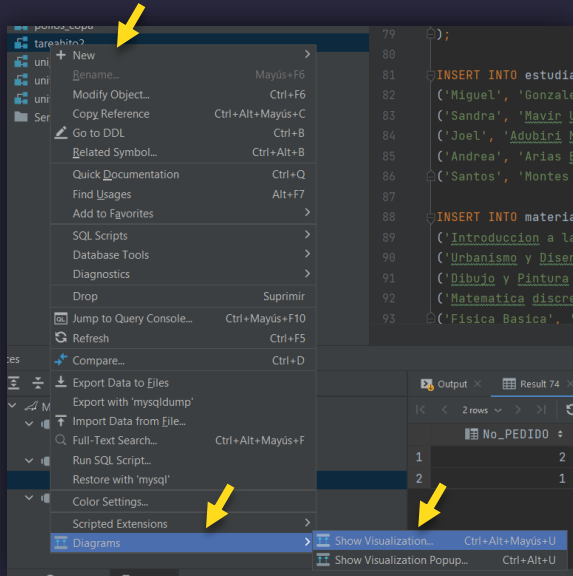
```
INSERT INTO materias (nombre_mat, cod_mat) VALUES
('Introduccion a la Arquitectura', 'ARQ-101'),
('Urbanismo y Diseno', 'ARQ-102'),
('Dibujo y Pintura Arquitectonico', 'ARQ-103'),
('Matematica discreta', 'ARQ-104'),
('Fisica Basica', 'ARQ-105');
```

```
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);
```

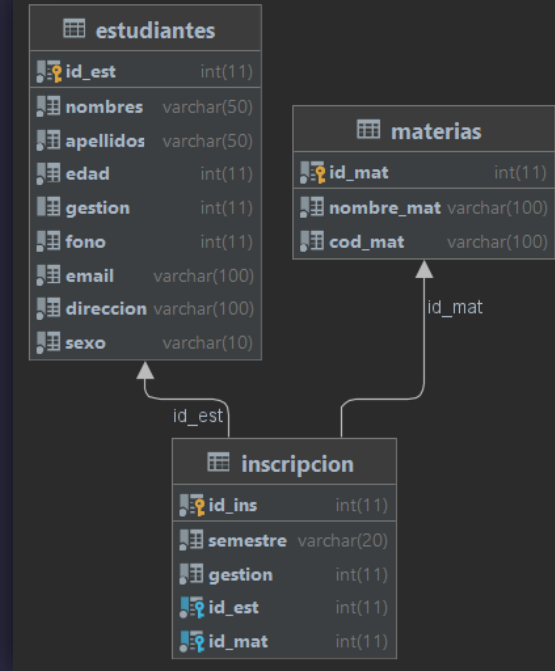
→



/DIAGRAMA LOGICO DEL PROYECTO.



Haciendo un clic derecho sobre la carpeta de tablas, vaya y seleccione DIAGRAMS y la opción SHOW VISUALIZATION para poder ver el diagrama.





/Creación de consultas SQL en base al ejercicio anterior.

Mostrar los nombres y apellidos de los estudiantes inscritos en la materia ARQ-105, adicionalmente mostrar el nombre de la materia. Deberá ser utilizada en la cláusula WHERE.

```
CREATE OR REPLACE FUNCTION est_mat(cod_mat VARCHAR(50))
RETURNS INT
BEGIN
    DECLARE parametro INT DEFAULT 0;
    SELECT est.id_est INTO parametro
    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 = cod_mat;
    RETURN parametro;
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 est_mat(cod_mat 'ARQ-105') = est.id_est;
```

EJECUCION:

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





/Creación de consultas SQL en base al ejercicio anterior.

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 OR REPLACE FUNCTION promedio_edades(genero VARCHAR(50))
RETURNS INT
BEGIN
    DECLARE parametro INTEGER DEFAULT 0;
    SELECT AVG(est.edad) INTO parametro
    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 = genero AND mat.cod_mat = 'ARQ-104';

    RETURN parametro;
end;

SELECT promedio_edades(genero: 'femenino');
```

EJECUCION:

PROMEDIO_EDADES ▾	
1	23





/Creación de consultas SQL en base al ejercicio anterior.

Crear una función que permita concatenar 3 cadenas.

- o La función recibe 3 parámetros.
- o Si la cadenas fuesen:
 - Pepito
 - Perez
 - 50
- o La salida debería ser: Pepito - Perez - 50

```
CREATE OR REPLACE FUNCTION concatenar(parametro1 VARCHAR(50), parametro2 VARCHAR(50), parametro3 VARCHAR(50))  
RETURNS TEXT  
BEGIN  
    DECLARE parametro TEXT;  
    SELECT CONCAT(parametro1,' - ', parametro2,' - ', parametro3) INTO parametro;  
    RETURN parametro;  
end;  
  
SELECT concatenar( parametro1: 'Pepito', parametro2: 'Perez', parametro3: '50') AS FUNCION_CONCATENAR;
```

EJECUCION:

FUNCION_CONCATENAR

1 Pepito - Perez - 50



/Creación de consultas SQL en base al ejercicio anterior.

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. Deberá ser utilizada en la cláusula WHERE.

```
CREATE OR REPLACE FUNCTION est_inscritos(sexo VARCHAR(50), edad INT)
RETURNS INT
BEGIN
    RETURN(
        SELECT SUM(est.edad)
        FROM estudiantes AS est
        WHERE est.sexo = sexo AND est.edad >= edad
    );
end;

SELECT est.id_est, est.nombres, est.apellidos, ins.semestre
FROM estudiantes AS est
INNER JOIN inscripcion AS ins ON est.id_est = ins.id_est
WHERE est_inscritos( sexo: 'masculino', edad: 22) % 2 = 0;
```

EJECUCION:

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



/Creación de consultas SQL en base al ejercicio anterior.

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). Deberá ser utilizada en la cláusula WHERE.

- La función devuelve un boolean.
- La función debe recibir el nombre y sus apellidos.

```
CREATE OR REPLACE FUNCTION comparacion(nombre VARCHAR(100), apellido VARCHAR(100))  
RETURNS BOOLEAN  
BEGIN  
    DECLARE validar BOOLEAN;  
    SELECT est.id_est INTO validar  
    FROM estudiantes as est  
    WHERE est.nombres = nombre AND est.apellidos = apellido;  
    RETURN validar;  
end;  
  
SELECT est.*  
FROM estudiantes AS est  
WHERE comparacion( nombre: 'Miguel', apellido: 'Gonzales Veliz') = est.id_est;
```

EJECUCION:

	id_est	nombres	apellidos	edad	gestion	fono	email	direccion	sexo
1	1	Miguel	Gonzales Veliz	20	<null>	2832115	miguel@gmail.com	Av. 6 de Agosto	masculino





¡GRACIAS POR SU ATENCION!



ronald.choque2111@gmail.com

+591 65648933



BASE DE DATOS II



EL ALTO, ABRIL DE 2022



INNOVACIÓN
EN EDUCACIÓN