



/DEFENSA FINAL HITO 2

BASE DE DATOS II

EDSON IVER CONDORI CONDORI
SIS10929449



EL ALTO, MARZO DE 2023

UNIFRANZ
Internacionalízate

**INNOVACIÓN
EN EDUCACIÓN**



/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 los
requerimientos.





/START!

> /BDAII





/01 /PARTE TEORICA



/CONCEPTOS



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

Nos referimos a una base de datos relacional cuando se trabaja mediante un conjunto de **TABLAS**, las cuales están formadas por **FILAS (registros)** y **COLUMNAS (campos)**.



2. ¿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.



3. ¿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



4.¿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.



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

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.



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

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



/CONCEPTOS



7. ¿Para qué sirve el comando USE?

Es el posicionamiento a la BDA en la que se va a trabajar.

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



8. ¿Que es DML y DDL?

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

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



¿Qué cosas características debe de tener una función?

Las mas elementales:

```
CREATE OR REPLACE FUNCTION est_mat(cod_mat VARCHAR(50))  
RETURN INT  
BEGIN  
  DECLARE parametro INT DEFAULT 0;  
  SELECT est.in_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:



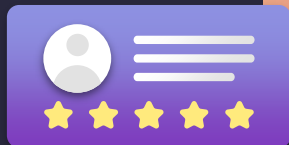
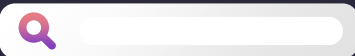
```
DROP FUNCTION comparacion;
```

ELIMINA SI EXISTE:



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

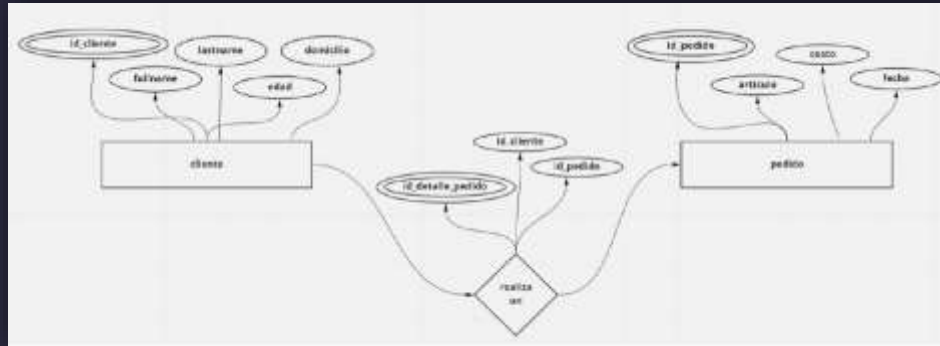




/02 /PARTE PRACTICA



/ 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; -- creacion de la base de datos
USE Pollos_Copa; -- posicionando en la base de datos
```

El comando CREATE DATABASE es el que genera la DBA.

Comando USE para posicionarnos en la BDA para trabajar en el mismo.



/CREACION DE LAS TABLAS.

-- CREACION DE TABLA CLIENTES

CREATE TABLE Clientes(

id_cliente INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL ,
fullname VARCHAR (30) NOT NULL,
lastname VARCHAR (30) NOT NULL,
edad INTEGER NOT NULL ,
domicilio VARCHAR (50) NOT NULL

);

-- CREANDO LA TABLA DETALLE PEDIDO CON RELACION DE LAS 2 TABLAS ANTERIORES

CREATE TABLE detalle_pedido (

id_detalle INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
id_cliente INTEGER NOT NULL,
id_pedido INTEGER NOT NULL,

FOREIGN KEY (id_cliente) REFERENCES Clientes (id_cliente),
FOREIGN KEY (id_pedido) REFERENCES pedido (id_pedido)

);

-- CREACION DE TABLA PEDIDO

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

);

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.



```
-- INGRESANDO REGISTROS A LA TABLA CLIENTES
```

```
INSERT INTO Clientes (fullname, lastname, edad, domicilio)
```

```
VALUES('Edson Iver', 'Condori Condori', 19, 'Faro Murillo'),  
      ('Jhonatan David', 'Alanoca Blanco', 21, '16 de Julio'),  
      ('Carlo Hernesto', 'Torres de la Cruz', 22, 'Obelisco'),  
      ('Juan Carlos', 'Vazques Ramos', 25, 'Sopocachi'),  
      ('Carmen Gabriela', 'Lopez Mendoza', 23, 'Costanera');
```

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

```
-- INGRESANDO REGISTROS A LA TABLA PEDIDO
```

```
INSERT INTO pedido (articulo, costo, fecha)
```

```
VALUES('Cesta de Pollo', 95, '2022-03-25'),  
      ('1/2 de Pollo', 85, '2022-03-22'),  
      ('Combo Fiesta', 60, '2022-03-18'),  
      ('Balde de Alitas', 70, '2022-03-15'),  
      ('Combo Doble', 45, '2022-03-10');
```

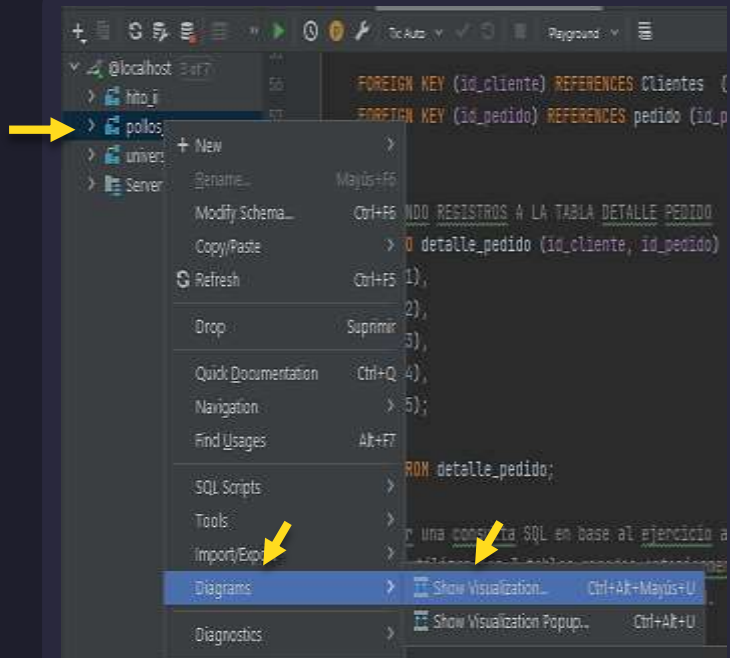
```
-- INGRESANDO REGISTROS A LA TABLA DETALLE PEDIDO
```

```
INSERT INTO detalle_pedido (id_cliente, id_pedido)
```

```
VALUES (1,1),  
      (2,2),  
      (3,3),  
      (4,4),  
      (5,5);
```

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

/DIAGRAMA LOGICO DEL PROYECTO.



Haciendo un clic derecho sobre la carpeta de principal, dirijase 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.

```
-- MOSTRAR LOS CLEINTES QUE ADQUIRIERON EL PRODUCTO COMBO FIESTA
SELECT cl.fullname,cl.lastname,ped.costo,ped.fecha,detped.id_pedido -- LO QUE SE MUESTRA
FROM clientes as cl
INNER JOIN pedido as ped on cl.id_cliente = ped.id_pedido
INNER JOIN detalle_pedido as detped on ped.id_pedido = detped.id_detalle
WHERE ped.articulo = 'Combo Fiesta'; -- LA CONDICION QUE SE DEBE CUMPLIR
```

CONSULTA: Se desea ver el nombre, apellido, costo, fecha y id del pedido de los clientes que adquirieron el "Combo Fiesta":

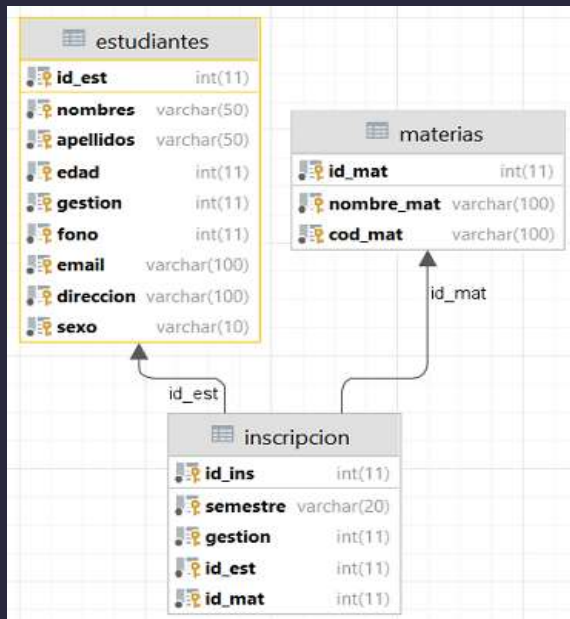
EJECUCION:

	fullname	lastname	costo	fecha	id_pedido
1	Carlo Hernesto	Torres de la Cruz	60	2022-03-18	3



/Crear un función que compare dos códigos de materia.

o Recrear la siguiente base de datos:





/CREACION DE LA DBA.



```
CREATE DATABASE tareaHito2; -- CREACION DE LA BASE DE DATOS
USE tareaHito2; -- POSICIONAMIENTO
```

El comando CREATE DATABASE es el que genera la DBA.

Comando USE para posicionarnos en la BDA para trabajar en el mismo.



/CREACION DE LAS TABLAS.

```
-- CREACION DE LA TABLA ESTUDIANTE
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,
    fono INTEGER NOT NULL ,
    email VARCHAR (100) NOT NULL ,
    direccion VARCHAR (100) NOT NULL ,
    genero VARCHAR (20)
);
```

```
-- CREACION DE LA TABLA MATERIAS
CREATE TABLE materias (
    id_mat INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
    nombre_mat VARCHAR(100) NOT NULL ,
    cod_mat VARCHAR(100) NOT NULL
);
```

```
-- CREACION DE LA TABLA INSCRIPCION (SE REALIZA LA REALCION CON LAS 2 TABLAS ANTERIORES)
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 ,

    -- RELACIONANDO LAS TABLAS
    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.



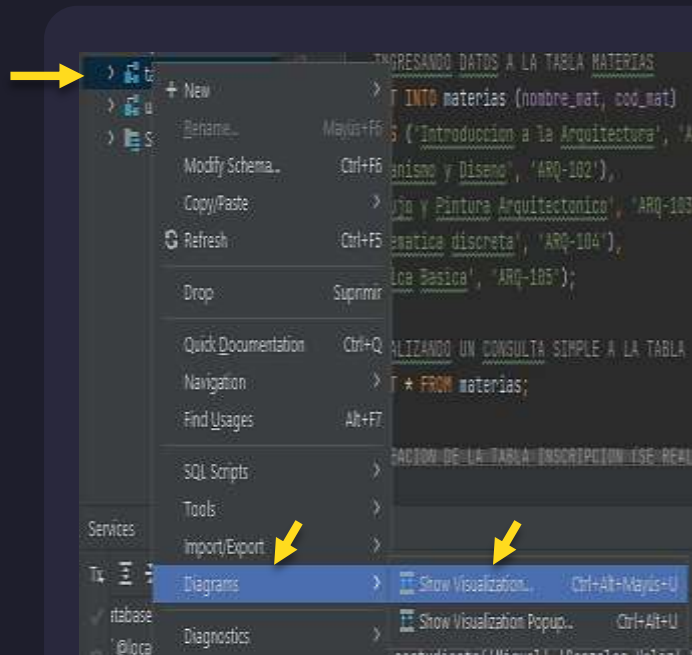
```
-- INGRESANDO DATOS A LA TABLA ESTUDIANTES
INSERT INTO estudiantes (nombres, apellidos, edad, fono, email, direccion, genero)
VALUES ('Miguel', 'Gonzales Veliz', 20, 2832115, 'miguel@gmail.com', 'Av. 6 de Agosto', 'masculino'),
('Sandra', 'Mavir Uria', 25, 2832116, 'sandra@gmail.com', 'Av. 6 de Agosto', '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');
```

```
-- INGRESANDO DATOS A LA TABLA MATERIAS
INSERT INTO materias (nombre_mat, cod_mat)
VALUES ('Introducción a la Arquitectura', 'ARQ-101'),
('Urbanismo y Diseño', 'ARQ-102'),
('Dibujo y Pintura Arquitectónico', 'ARQ-103'),
('Matemática discreta', 'ARQ-104'),
('Física Básica', 'ARQ-105');
```

```
-- INGRESANDO REGISTROS A LA TABLA INSCRIPCION
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 principal, dirijase 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.

```
-- creando la funcion que recibe dos parametros
CREATE OR REPLACE FUNCTION comparaMateria(cod_mat VARCHAR(100), edad INTEGER)
RETURNS VARCHAR (100) -- retornamos en tipo varchar
BEGIN
    DECLARE result VARCHAR(100); -- declaramos para obtener la respuesta

    -- Implementamos la consulta anterior con algunas modificaciones
    -- concatenamos lo que queremos mostrar
    SELECT CONCAT(est.id_est, ' ', est.nombres, ' ', est.apellidos, ' ', est.edad, ' ', mat.nombre_mat, ' ', mat.cod_mat)
    INTO result -- almacenamos la concatenacion en la variable declarada anteriormente
    FROM estudiantes AS est
    INNER JOIN materias AS mat ON est.id_est = mat.id_mat -- establecemos lo que se va relacionar entre tablas
    WHERE mat.cod_mat = cod_mat and est.edad = edad; -- se deben de cumplir las condiciones que se manden

    -- En caso de que no se encuentre nada se devolvera o retornara lo siguiente
    IF result IS NULL THEN
        SET result = 'No se encontraron coincidencias';
    END IF;

    RETURN result; -- Se devuelve la respuesta de la funcion
END; -- Fin de la Funcion

-- CONSULTA A LA FUNCION
SELECT comparaMateria(cod_mat 'ARQ-105', edad 24);
```

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

EJECUCION:

```
comparaMateria('ARQ-105',24)
1 5 Santos Montes Valenzuela 24 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.

o La función recibe como parámetro el género y el código de materia.

```
-- USANDO LA CONSULTA ANTERIOR EN UNA FUNCION
CREATE OR REPLACE FUNCTION promedioedadgenero (genero VARCHAR(50),cod_mat VARCHAR(50))
RETURNS VARCHAR(100)
BEGIN

    DECLARE response VARCHAR(100);

    SELECT AVG (est.edad)
    INTO response
    FROM estudiantes AS est
    INNER JOIN inscripcion ins on est.id_est = ins.id_est
    INNER JOIN materias mat on ins.id_mat = mat.id_mat
    WHERE mat.cod_mat = cod_mat AND est.genero = genero ;

    IF response IS NULL THEN
        SET response = 'No se encontraron coincidencias';
    END IF;

    RETURN response;
END;

SELECT promedioedadgenero( genero: 'femenino', cod_mat: 'ARQ-104');
```

```
-- REALIZANDO UN CONSULTA PARA LUEGO IMPLEMENTAR A UNA FUNCION
SELECT est.nombres, est.apellidos,AVG (est.edad)
FROM estudiantes AS est
INNER JOIN inscripcion ins on est.id_est = ins.id_est
INNER JOIN materias mat on ins.id_mat = mat.id_mat
WHERE mat.cod_mat = 'ARQ-104' AND est.genero = 'Femenino';
```

EJECUCION:

```
1 promedioedadgenero('femenino','ARQ-104')
1 23.0000000000
```





/Creación de consultas SQL en base al ejercicio anterior.

Crear una función que permita concatenar 3 cadenas.

```
-- CREACION DE LA FUNCION
CREATE OR REPLACE FUNCTION datosestudiante (nom VARCHAR(50), ape VARCHAR(50) , edad INTEGER )
RETURNS VARCHAR(100)
BEGIN
    DECLARE response VARCHAR(100); -- DECLARANDO TIPO DE VARIABLE
    |
    -- CONSULTA
    SELECT CONCAT(est.nombres,' ',est.apellidos,' ',est.edad)
    INTO response
    FROM estudiantes as est
    WHERE est.nombres = nom and est.apellidos = ape and est.edad = edad;

    -- RESPUESTA EN CASO DE NO ENCONTRAR NADA
    IF response IS NULL THEN
        SET response = 'No se encontraron coincidencias';
    END IF;
    RETURN response; -- DEVOLVIENDO LA RESPUESTA
END;

-- REALIZANDO LA CONSULTA A LA FUNCION
SELECT datosestudiante( nom: 'Miguel', ape: 'Gonzales Veliz', edad: 20);
```

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

EJECUCION:

```
SQL> datosestudiante('Miguel','Gonzales Veliz',20)
1 Miguel Gonzales Veliz 20
```





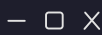
/Creación de consultas SQL en base al ejercicio anterior.



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 2018.
- 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





/Creación de consultas SQL en base al ejercicio anterior.

```
-- CONSULTA PREVIA ANTES DE LA IMPLEMENTACION DE VISTA Y EXPLICACION DE CODIGO
SELECT CONCAT(est.nombres, ' ', est.apellidos) as fullname, -- CONCATENANDO LO QUE SE DESEA MOSTRAR Y ASIGNADO ALIAS
       est.edad as edad,
       ins.gestion as gestion,

CASE -- REALIZAMOS LA EVALUACION

    WHEN -- WHEN ESPECIFICA LA CONDCION QUE SE DEBE CUMPLIR
    EXISTS -- EXISTS verificara SI existe algun registro en la tabla

    (SELECT 1 FROM inscripcion -- SELECT 1 buscara un registro q cumpla con la condicion
     -- si encuentra una lo devolvera como verdadero

    WHERE ins.id_ins = est.id_est -- condicionamos para que se cumpla
    AND ins.gestion = '2018') -- CONDCION Y ESTAMOS BUSCANDO
    -- QUE SE CUMPLA ESTA CONDCION (ACTUA DE RECEPTOR Y VERIFICADOR)

    THEN 'LIBRE' -- THEN DEVOLVERA EL VALOR/RESPUESTA CORRECTA SI CUMPLE CON LA CONDCION
    ELSE 'NO LIBRE' -- ELSE DEVOLVERA UNA RESPUESTA NEGATIVA SI NO CUMPLE CON LA CONDCION
    END AS DIA_LIBRE -- ESTABLECIENDO LA NUEVA COLUMNA

FROM estudiantes as est -- RELACIONANDO TABLAS
INNER JOIN materias as mat on est.id_est = mat.id_mat
INNER JOIN inscripcion as ins on mat.id_mat = ins.id_ins
WHERE ins.gestion = 2020; -- AQUI PREGUNTAMOS LO QUE SE BUSCA (ACTUA DE EMISOR "lo manda al where de arriba")
```



/Creación de consultas SQL en base al ejercicio anterior.

```
-- IMPLEMENTANDO EN UNA VISTA
CREATE OR REPLACE VIEW arquitectura_dia_libre AS
SELECT CONCAT(est.nombres, ' ', est.apellidos) as fullname,
       est.edad as edad,
       ins.gestion as gestion,
       CASE
         WHEN EXISTS (SELECT 1 FROM inscripcion
                      WHERE ins.id_ins = est.id_est
                      AND ins.gestion = '2018')
         THEN 'LIBRE'
         ELSE 'NO LIBRE'
       END AS DIA_LIBRE
FROM estudiantes as est
INNER JOIN materias as mat on est.id_est = mat.id_mat
INNER JOIN inscripcion as ins on mat.id_mat = ins.id_ins
WHERE ins.gestion = 2018;
```

EJECUCION:

	fullname	edad	gestion	DIA_LIBRE
1	Miguel Gonzales Veliz	20	2018	LIBRE
2	Sandra Mavir Uria	25	2018	LIBRE



/Creación de consultas SQL en base al ejercicio anterior. ●●●

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`





/Creación de consultas SQL en base al ejercicio anterior.

```
-- CREANDO UN TABLA
CREATE TABLE SEDE (
    id_sede INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL ,
    nombre_sede VARCHAR (100) NOT NULL ,
    id_ins INTEGER NOT NULL ,

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

```
-- INGRESANDO REGISTROS A LA TABLA
INSERT INTO SEDE (nombre_sede, id_ins)
VALUES ('El Alto',1),
       ('La Paz',2),
       ('Oruro',3),
       ('Cochabamba',4),
       ('Santa Cruz',5);
```

```
-- REALIZANDO LA CONSULTA
-- Mostrar los estudiantes de la gestion 2019 de la sede de Oruro
SELECT est.nombres,est.apellidos,mat.cod_mat,ins.semestre,ins.gestion,se.nombre_sede
FROM estudiantes as est
INNER JOIN materias as mat on est.id_est = mat.id_mat
INNER JOIN inscripcion as ins on mat.id_mat = ins.id_ins
INNER JOIN SEDE as se on ins.id_ins = se.id_sede
WHERE ins.gestion = 2019 and se.nombre_sede = 'Oruro';
```

EJECUCION DE LA CONSULTA:

	nombres	apellidos	cod_mat	semestre	gestion	nombre_sede
1	Joel	Adubiri Mondar	ARQ-103	1er Semestre	2019	Oruro





/Creación de consultas SQL en base al ejercicio anterior. ● ● ●

```
-- IMPLEMENTANDO LA VISTA
CREATE OR REPLACE VIEW PARALELO_DBA_I AS
SELECT est.nombres,est.apellidos,mat.cod_mat,ins.semestre,ins.gestion,se.nombre_sede
FROM estudiantes as est
INNER JOIN materias as mat on est.id_est = mat.id_mat
INNER JOIN inscripcion as ins on mat.id_mat = ins.id_ins
INNER JOIN SEDE as se on ins.id_ins = se.id_sede
WHERE ins.gestion = 2019 and se.nombre_sede = 'Oruro';

-- ===== REALIZADO POR: ===== --
-- ===== Edson Iver Condori Condori ===== --
```





¡GRACIAS POR SU ATENCION!



condoedsoniver@gmail.com
+591 72096981



BASE DE DATOS II



EL ALTO, MARZO DE 2023



INNOVACIÓN
EN EDUCACIÓN