

EVALUCION PROCESUAL HITO 2

BASE DE DATOS II

Start

PRESENTA:

RONALD URIEL CHOQUE PACO – SIS6972733

COMANDOS ELEMENTALES PARA SU CONPRENSION:

Para optimizar nuestro tiempo, explicaremos algunos comandos elementales:

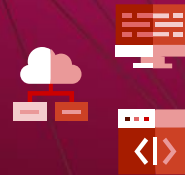
1. **CREATE DATABASE** crea nuestra base de datos, además de **USE** para ejecutar acciones en la misma.
2. **CREATE TABLE** genera nuestras tablas, cada tabla tendrá sus columnas y estas deberán ser definidas por un tipo de variable (**INT**, **VARCHAR**, **DATE**); además del uso de **NOT NULL** para no aceptar campos vacíos.
3. **PRIMARY KEY** define la llave primaria de nuestra tabla, **FOREING KEY** define nuestra llave foránea con objetivo de relacionar los campos entre tablas.
4. **SELECT** para seleccionar los campos a visualizar, **FROM** para seleccionar la tabla y **WHERE** para agregar algún tipo de condición.
5. **CREATE FUNCTION** para crear nuestras funciones, los cuales retornaran un tipo de para metro (ej. **RETURN BOOLEAN**).



START!

Go!

1. ¿A QUE SE REFIERE CUANDO HABLAMOS BASES DE DATOS RELACIONALES Y NO RELACIONALES?



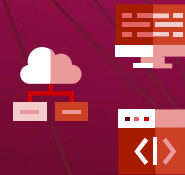
RELACIONALES

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

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.

2. ¿QUE ES MYSQL Y MARIADB?. EXPLIQUE.



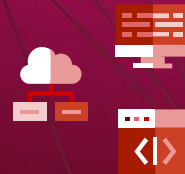
MYSQL

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

MARIADB

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

3. ¿QUE ES UNA FUNCION DE AGREGACION?. ADICIONALMETE MUESTRE UN EJ.



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

Los comandos mas usados son:

- SUM (suma de datos).
- AVG (promedio de datos).
- MIN Y MAX (menor y mayor respectivamente),
- LEN (conteo de caracteres).
- STRCMP (comparación de dos tipo de datos tipo booleano).
- COUNT (conteo de registros),

```
SELECT SUM(est.edad) INTO sumaEdad
FROM estudiantes AS est
```

```
SELECT STRCMP(est.nombres, est.apellidos)
FROM estudiante as est
```

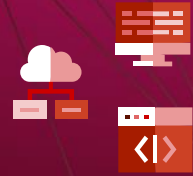
```
SELECT AVG(est.edad)
FROM estudiantes AS est
```

```
SELECT MAX(est.id_est)
FROM estudiantes AS est
```

```
SELECT MIN(est.edad)
FROM estudiantes AS est
```

CONTINUAR

4. ¿A QUE SE REFIERE CUANDO HABLAMOS BASES DE DATOS RELACIONALES Y NO RELACIONALES?



USE define la DBA en la que se va a trabajar.

DDL (DATA MANIPULATION LANGUAGE)

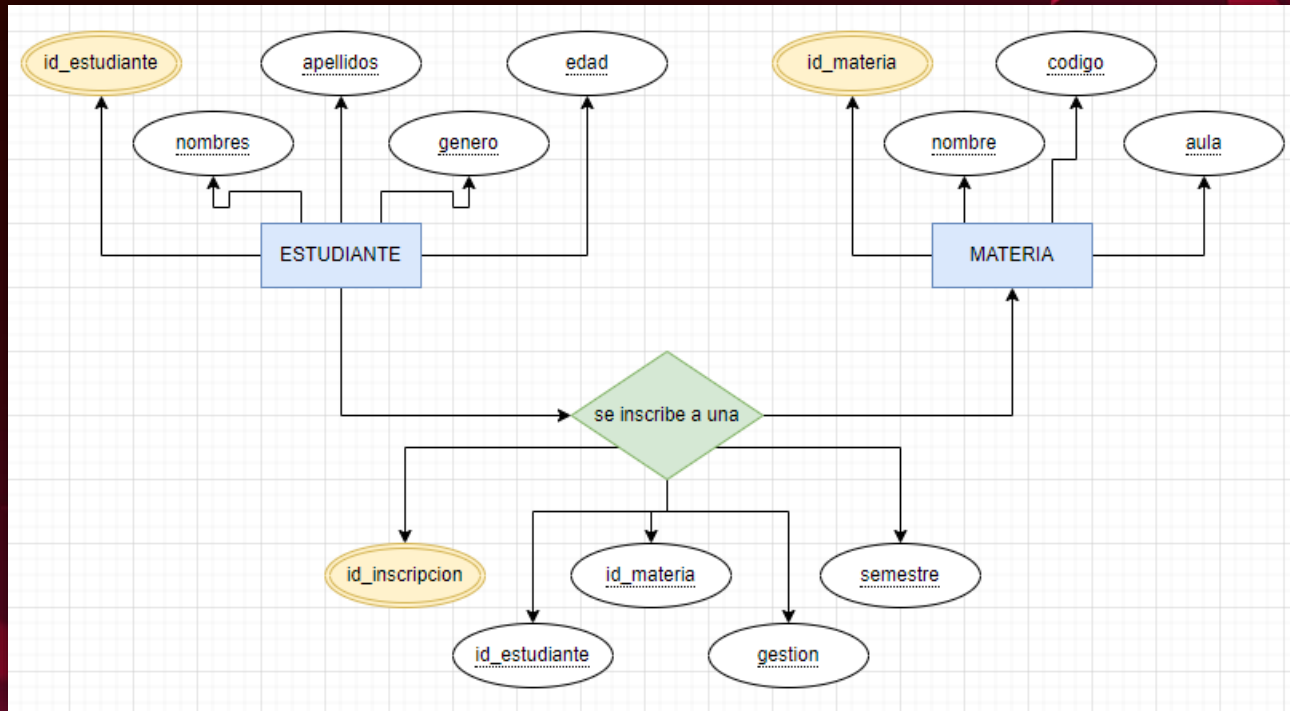
```
SELECT ins.id_est, AVG(ins.id_mat)
FROM inscripcion AS ins
GROUP BY ins.id_est;
```

DML (DATA DEFINITION LANGUAGE)

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

5. GENERAR LA BASE DE DATOS PARA EL SIGUIENTE MODELO E-R.

Crear la base de datos UNI_hito2



5. GENERAR LA BASE DE DATOS PARA EL SIGUIENTE MODELO E-R.

Crear la base de datos UNI_hito2

```
CREATE DATABASE UNI_Hito2;  
USE UNI_Hito2;
```

Crear la tabla estudiante

```
CREATE TABLE estudiante  
(  
    id_estudiante INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,  
    nombres VARCHAR(50) NOT NULL,  
    apellidos VARCHAR(50) NOT NULL,  
    edad INTEGER NOT NULL,  
    genero VARCHAR(10) NOT NULL  
);
```

Crear la tabla materia

```
CREATE TABLE materia  
(  
    id_materia INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,  
    nombre_mat VARCHAR(100) NOT NULL,  
    cod_mat VARCHAR(100) NOT NULL,  
    aula INTEGER NOT NULL  
);
```

Crear la tabla inscripción

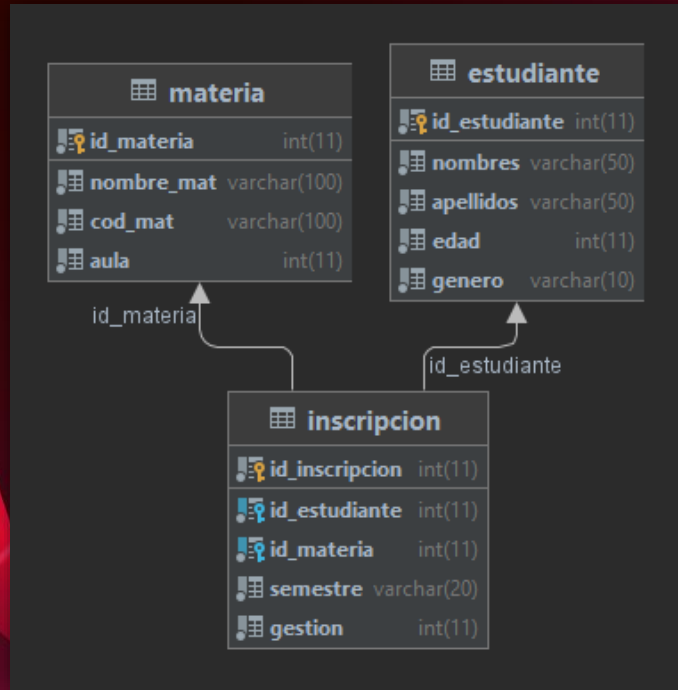
```
CREATE TABLE inscripcion  
(  
    id_inscripcion INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,  
    id_estudiante INTEGER NOT NULL,  
    id_materia INTEGER NOT NULL,  
    semestre VARCHAR(20) NOT NULL,  
    gestion INTEGER NOT NULL,  
    FOREIGN KEY (id_estudiante) REFERENCES estudiante (id_estudiante),  
    FOREIGN KEY (id_materia) REFERENCES materia (id_materia)  
);
```

```
INSERT INTO estudiante(nombres, apellidos, edad, genero) VALUES  
( 'nombre1', 'apellido1', 19, 'masculino'),  
( 'nombre2', 'apellido2', 21, 'masculino'),  
( 'nombre3', 'apellido3', 19, 'femenino');  
  
INSERT INTO materia(nombre_mat, cod_mat, aula) VALUES  
( 'Base de Datos I', 'BDA-211', 212),  
( 'Programacion I', 'PRO-231', 313),  
( 'Estructura de Datos I', 'EST-311', 414);  
  
INSERT INTO inscripcion(id_estudiante, id_materia, semestre, gestion) VALUES  
(3, 3, '3er Semestre', 2022),  
(1, 2, '1er Semestre', 2021),  
(2, 1, '2do Semestre', 2022);  
  
INSERT INTO estudiante(nombres, apellidos, edad, genero) VALUES  
( 'nombre4', 'apellido4', 18, 'masculino'),  
( 'nombre5', 'apellido5', 22, 'femenino');  
  
INSERT INTO materia(nombre_mat, cod_mat, aula) VALUES  
( 'Sistemas Informaticos', 'SIS-121', 515);  
  
INSERT INTO inscripcion(id_estudiante, id_materia, semestre, gestion) VALUES  
(4, 4, '3er Semestre', 2022),  
(5, 4, '3er Semestre', 2022);
```

5. GENERAR LA BASE DE DATOS PARA EL SIGUIENTE MODELO E-R.



Diagrama lógico de la base de datos UN1_hito2



6. GENERAR UNA FUNCION PARA MOSTRAR LA MENOR EDAD.



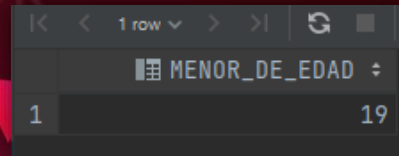
CODIGO:

```
CREATE OR REPLACE FUNCTION menor_edad(parametro1 VARCHAR(50))
RETURNS INT
BEGIN
    DECLARE menEdad INTEGER DEFAULT 0;

    SELECT MIN(est.edad) INTO menEdad
    FROM estudiante AS est
    WHERE est.genero = parametro1;
    RETURN menEdad;
end;

SELECT menor_edad( parametro1: 'femenino') AS MENOR_DE_EDAD;
```

EJECUCION:

A screenshot of a database query results window. It shows a single row of data. The column header is 'MENOR_DE_EDAD' and the value in the row is '19'. The window has a dark theme and includes navigation buttons at the top.

	MENOR_DE_EDAD
1	19



7. MOSTRAR LOS ESTUDIANTES (NOMBRES Y APELLIDOS) QUE ESTEN INSCRITOS EN LA GESTION 2022.

CODIGO:

```
SELECT est.id_estudiante, est.nombres, est.apellidos, ma.nombre_mat, ins.gestion
FROM estudiante AS est
INNER JOIN inscripcion AS ins ON est.id_estudiante = ins.id_estudiante
INNER JOIN materia AS ma ON ins.id_materia = ma.id_materia
WHERE ins.gestion = 2022;
```

EJECUCION:

	id_estudiante	nombres	apellidos	nombre_mat	gestion
1	2	nombre2	apellido2	Base de Datos I	2022
2	3	nombre3	apellido3	Estructura de Datos I	2022
3	4	nombre4	apellido4	Sistemas Informaticos	2022
4	5	nombre5	apellido5	Sistemas Informaticos	2022



8. MOSTRAR LOS NOMBRES Y APELLIDOS DE LOS ESTUDIANTES INSCRITOS EN LA MATERIA SIS-121, ADICIONALMENTE MOSTRAR EL NOMBRE DE LA MATERIA.

CODIGO:

```
CREATE OR REPLACE FUNCTION inscritos(id_est INT, codigo_mat varchar(100))
    RETURNS BOOLEAN
BEGIN
    DECLARE parametro BOOLEAN DEFAULT 0;
    SELECT est.id_estudiante into parametro
    FROM estudiante AS est
    INNER JOIN inscripcion AS ins ON est.id_estudiante = ins.id_estudiante
    INNER JOIN materia AS ma ON ins.id_materia = ma.id_materia
    WHERE est.id_estudiante = id_est AND ma.cod_mat = codigo_mat;
    RETURN parametro;
end;

SELECT est.nombres, est.apellidos, ma.nombre_mat, ma.cod_mat
FROM estudiante AS est
INNER JOIN inscripcion AS ins ON est.id_estudiante = ins.id_estudiante
INNER JOIN materia AS ma ON ins.id_materia = ma.id_materia
WHERE inscritos( id_est: 4, codigo_mat: 'SIS-121') = est.id_estudiante;
```

EJECUCION:

	nombres	apellidos	nombre_mat	cod_mat
1	nombre4	apellido4	Sistemas Informaticos	SIS-121



9. MOSTRAR NOMBRE, APELLIDO Y SEMESTRE DE TODOS LOS ESTUDIANTES INSCRITOS SIEMPRE Y CUANDO LA SUMA DE LAS EDADES DEL SEXO “M O F” SEA PAR Y MAYORES A CIERTA EDAD.

CODIGO:

```
CREATE OR REPLACE FUNCTION sum_edades(sexo VARCHAR(50), edad INT)
RETURNS INT
BEGIN
    DECLARE sumaEdad INTEGER DEFAULT 0;
    SELECT SUM(est.edad) INTO sumaEdad
    FROM estudiante AS est
    INNER JOIN inscripcion AS ins ON est.id_estudiante = ins.id_estudiante
    WHERE est.genero = sexo AND est.edad >= edad;
    RETURN sumaEdad;
end;

SELECT est.nombres, est.apellidos, ins.semestre
FROM estudiante AS est
INNER JOIN inscripcion AS ins ON est.id_estudiante = ins.id_estudiante
WHERE sum_edades( sexo: 'femenino', edad: 22) % 2 = 0;
```

EJECUCION:

	nombres	apellidos	semestre
1	nombre3	apellido3	3er Semestre
2	nombre1	apellido1	1er Semestre
3	nombre2	apellido2	2do Semestre
4	nombre4	apellido4	3er Semestre
5	nombre5	apellido5	3er Semestre



10. CREE UNA FUNCION SOBRE LA TABLA ESTUDIANTES QUE COMPARA EL NOMBRE Y APELLIDO (EN CASO DE QUE EXISTAN AMBOS MOSTRAR TODOS LOS DATOS DEL ESTUDIANTE).

CODIGO CON EL USO DE STRCMP:

```
CREATE OR REPLACE FUNCTION compara(nombre VARCHAR(100), apellido VARCHAR(100))
RETURNS BOOLEAN
BEGIN
    DECLARE parametro BOOLEAN;
    SELECT STRCMP(est.nombres, est.apellidos) INTO parametro
    FROM estudiante as est
    WHERE est.nombres = nombre AND est.apellidos = apellido;
    RETURN parametro;
end;

SELECT compara( nombre: 'nombre1', apellido: 'apellido1') AS DIFERENCIA;
```

EJECUCION:

DIFERENCIA	
1	1

DIFERENCIA	
1	<null>



10. CREE UNA FUNCION SOBRE LA TABLA ESTUDIANTES QUE COMPARA EL NOMBRE Y APELLIDO (EN CASO DE QUE EXISTAN AMBOS MOSTRAR TODOS LOS DATOS DEL ESTUDIANTE).

CODIGO MEDIANTE UNA CONSULTA TRADICIONAL:

```
CREATE OR REPLACE FUNCTION comparacion(nombre VARCHAR(100), apellido VARCHAR(100))
RETURNS BOOLEAN
BEGIN
    DECLARE validar BOOLEAN;
    SELECT est.id_estudiante INTO validar
    FROM estudiante as est
    WHERE est.nombres = nombre AND est.apellidos = apellido;
    RETURN validar;
end;

SELECT est.*
FROM estudiante AS est
WHERE comparacion( nombre: 'nombre2', apellido: 'apellido2') = est.id_estudiante;
```

EJECUCION:

1 row					Txc Auto	DDL
	id_estudiante	nombres	apellidos	edad	genero	
1	2	nombre2	apellido2	21	masculino	

¡GRACIAS POR SU ATENCION!

BASE DE DATOS II

¿TIENE ALGUNA CONSULTA?



RONALD.CHOQUE2111@GMAIL.COM



+591 65648933