Clase 04.
Vistas, Procedimientos almacenados, Funciones, Triggers. GRANT-REVOKE

Recordando lo aprendido hasta el momento

¿Qué es base de datos?

Una base de datos es un conjunto de datos estructurados que pertenecen a un mismo contexto y, en cuanto a su función, se utiliza para administrar de forma física y/o electrónica grandes cantidades de información.

Bases de datos transaccionales (OLTP)
Son bases de datos que soportan procesos transaccionales en una empresa, estos pueden ser sistemas de ventas, caja, logística,etc.

Bases de datos analíticas (OLAP)

Son sistemas que almacenan grandes volúmenes de datos. Su objetivo en concreto es servir a sistemas de soporte para la toma de decisiones.

Tablas
Las tablas son objetos de base de datos que contienen todos tus datos.

Qué es SQL
SQL, es un lenguaje de consulta a motores de bases de datos estructurados.

Sub lenguajes en SQL

- **DDL**
  Este es el conjunto de sentencias que se encargan de la definición de la base de datos y sus objetos.
- **DML**
  Este es el conjunto de sentencias que se encargan de la manipulación de datos.
- **DCL**
  Este es el conjunto de sentencias que se encargan de controlar la seguridad de los datos en una BD.
- **DQL:**
  Este es el conjunto de sentencias que se encargan de la consulta de datos.

DDL

1. BASES DE DATOS

Acciones	Ejemplo
Crear base de datos	CREATE DATABASE dbventas_datapath;
Eliminar Base de datos	DROP DATABASE dbventas_datapath;

2. TABLAS

Acciones		Ejemplo
CREAR	Crear tabla	/* CREAMOS LA TABLA VENDEDOR */ CREATE TABLE vendedor (idvendedor INTEGER PRIMARY KEY NOT NULL, nombre VARCHAR(100) NOT NULL, sexo CHAR(1) NOT NULL, dirección VARCHAR(100) DEFAULT NULL);
ELIMINAR	Eliminar Tabla **más** su estructura	DROP TABLE vendedor;
	Eliminar todos los registros de una tabla, **menos** su estructura	TRUNCATE TABLE vendedor;
MODIFICAR	Para añadir una nueva columna a una tabla	ALTER TABLE vendedor ADD edad INT;
	Para borrar una columna de una tabla	ALTER TABLE vendedor DROP COLUMN edad;
	Para modificar el tipo de dato de una columna de una tabla	ALTER TABLE vendedor ALTER COLUMN date DATATIME;

3. CREAR VISTAS

En SQL, una vista es una tabla virtual basada en el conjunto de resultados de una instrucción SQL.

¿Por qué necesitamos las vistas?
Una vista es virtual, los datos de una vista no se almacenan físicamente. Es un conjunto de consultas que, cuando se aplica a una o más tablas, se almacena en la base de datos como un objeto.

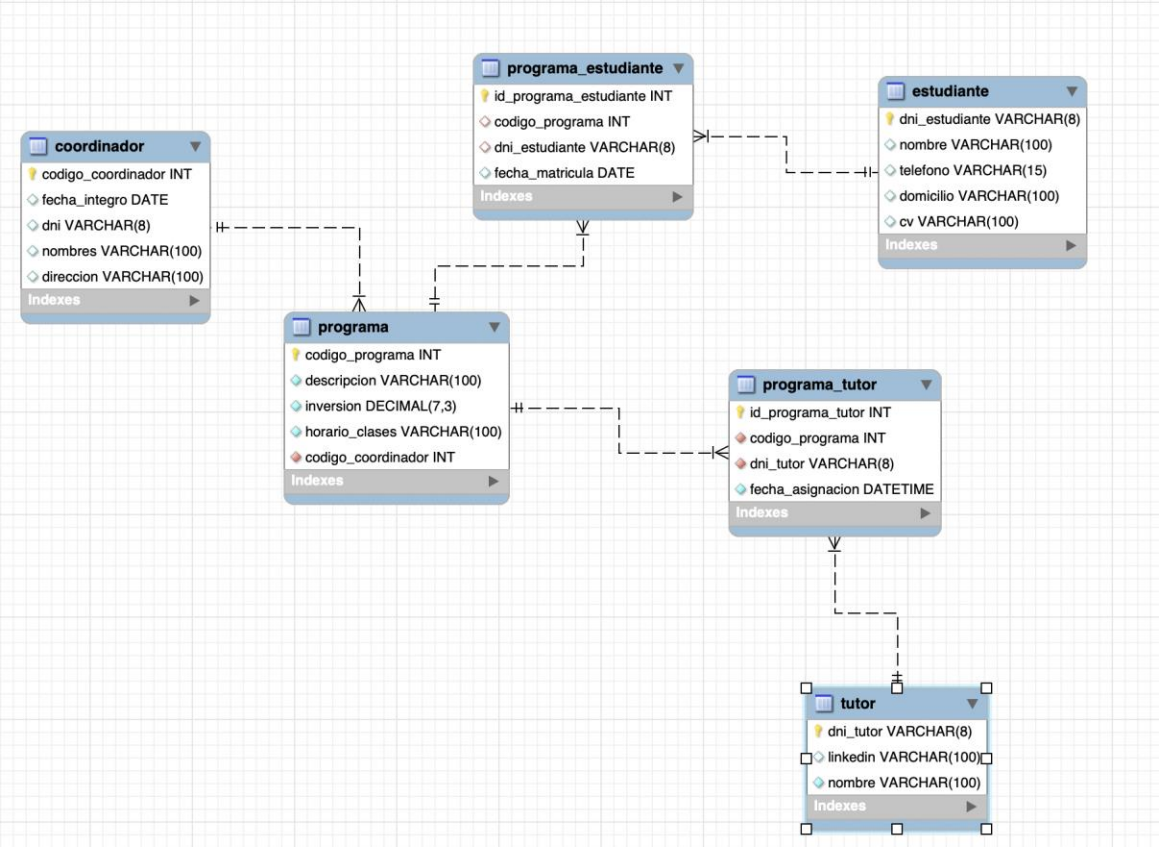
¿Cómo crear vistas?

Ejemplo CREATE VIEW

```
CREATE VIEW name_view AS SELECT * FROM test;
```

Ejemplo

Importante: La base de datos a utilizar es el académico_datapath. Este tiene la siguiente estructura:



Para crearla debemos utilizar este script:
https://drive.google.com/file/d/1S2oBU1SW4Eh9LLn8SliNm-0Co45ijPYq/view?usp=drive_link
Y también el de db_ventas_datapath:
https://drive.google.com/file/d/1Gn1tQVm6stcfZTdQxkpNsw9Upuc5il1e/view?usp=drive_link

En base a ello, desarrollaremos los siguientes ejercicios:

Creación de una vista para el programa de data analyst.

```
DROP VIEW IF EXISTS view_data_analyst;
CREATE VIEW  view_data_analyst AS
SELECT descripcion, inversion FROM programa
WHERE LOWER(descripcion) LIKE "%data analyst%";
```

Creación de una vista para el programa de data engineering

```
CREATE VIEW view_data_engineering AS
SELECT descripcion, inversion FROM programa
WHERE LOWER(descripcion) LIKE "%data engineer%";
```

¿Cómo consultar una vista?

```
SELECT * FROM name_view
```

¿Cómo elimino una vista?

```
DROP VIEW IF EXISTS name_view;
```

4. PROCEDIMIENTOS ALMACENADOS

Un procedimiento almacenado es un conjunto de instrucciones SQL que se almacena asociado a una base de datos.
Los procedimientos se asemejan a las construcciones de otros lenguajes de programación, porque pueden:
Aceptar parámetros de entrada y devolver varios valores en forma de parámetros de salida al programa que realiza la llamada.

¿Por qué necesitamos los procedimientos almacenados?

Encapsular consultas largas y repetitivas en una sola función. Optimizar el proceso de ejecución de las consultas.

¿Cómo crear procedimientos almacenados en MySql?

```
CREATE PROCEDURE name_procedure(p_1,p_2,...)
```

```
BEGIN
    QUERY
END
```

Ejemplo de un procedimiento almacenado

```
DELIMITER $$
CREATE PROCEDURE sp_programa_dia (IN dia_semana VARCHAR (50))
BEGIN
        SELECT * FROM programa
        WHERE LOWER(horario_clases) LIKE CONCAT('%' ,LOWER(dia_semana) , '%');
END
$$
```

Llamar un procedimiento almacenado

```
CALL sp_programa_dia("domingo");
```

5. FUNCIONES

Una función almacenada es un programa almacenado de tipo especial que devuelve un valor único.

¿Por qué necesitamos las funciones?

Por lo general, utiliza funciones almacenadas para encapsular fórmulas comunes o reglas comerciales que se pueden reutilizar entre sentencias SQL o programas almacenados.

Existen 2 tipo de funciones, Implícitas y explícitas

Funciones Implícitas: son las que **SÍ** están definidas por defecto en Mysql

Tipo	**Sintaxis**	**Ejemplo**
Texto	LOWER (*texto*)	[poner el ejemplo aquí]
	UPPER (*texto*)	[poner el ejemplo aquí]
	LENGTH (*texto*)	[poner el ejemplo aquí]

	CONCAT (*texto*)	[poner el ejemplo aquí]
Número	ROUND (*número*)	[poner el ejemplo aquí]
	ABS (*número*)	[poner el ejemplo aquí]
	LOG (*número*)	[poner el ejemplo aquí]
	PI (*número*)	[poner el ejemplo aquí]
	SQRT (*número*)	SELCT SQRT(9) AS reaiz_nueve;
Fecha	CURRENT_DATE ()	[poner el ejemplo aquí]
	CURRENT_TIME ()	[poner el ejemplo aquí]
	ADDDATE (*fecha, INTERVAL n DAY*)	[poner el ejemplo aquí]
	DATEDIFF (*fecha1,fecha2*)	[poner el ejemplo aquí]
	DAY (*fecha*)	SELECT DAY('202-09-28') AS dia;
	MONTH (*fecha*)	SELECT MONTH('202-09-28') AS mes;
	YEAR (*fecha*)	SELECT YAR('202-09-28') AS anio;
	EXTRACT (*fecha*)	[poner el ejemplo aquí]

Funciones explícitas: son las que **NO** están definidas por defecto en Mysql

¿Cómo crear funciones en MySql?

```
CREATE FUNCTION  function_name (p_1 type, p_2 type, ...)
RETURNS type
DETERMINISTIC
```

```
BEGIN
END
```

Ejemplo de función

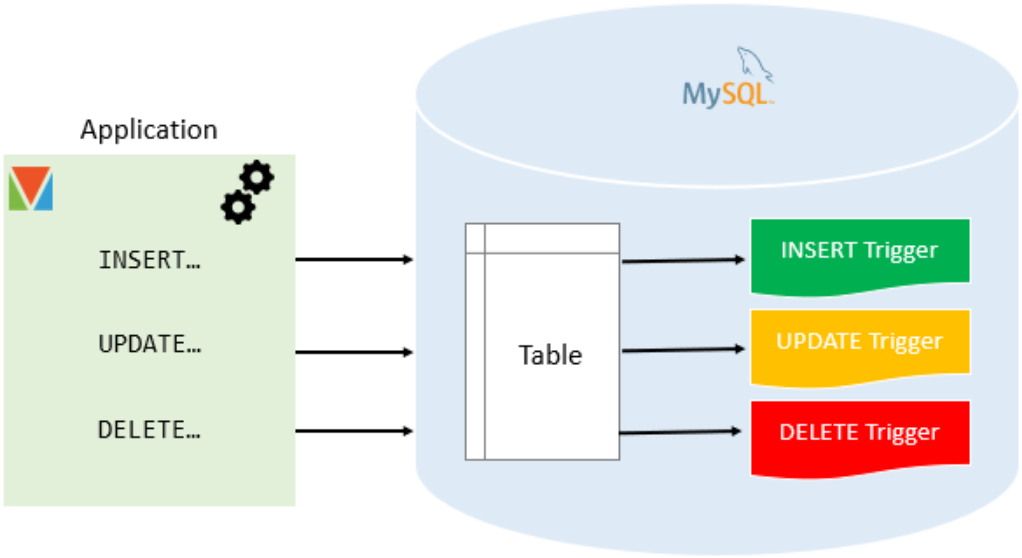
```
DELIMITER $$
CREATE FUNCTION precio_promedio( programa VARCHAR (30))
RETURNS DOUBLE
DETERMINISTIC
BEGIN
        IF programa="DAP"
                        THEN
                                SET  @precio_promedio:= (SELECT AVG(inversion) FROM
programa
           WHERE LOWER(descripcion) LIKE  "%data analyst%");
        ELSEIF (programa="DEP")
                        THEN
                                SET @precio_promedio:= (SELECT AVG(inversion) FROM
programa
           WHERE LOWER(descripcion) LIKE  "%data engineer%");
        ELSEIF (programa="DSP")
                        THEN
                                SET @precio_promedio:= (SELECT AVG(inversion) FROM
programa
           WHERE LOWER(descripcion) LIKE "%data scientist%");
         END IF;
       RETURN (@precio_promedio);
 END;
$$
```

como llamar a una función

```
SELECT precio_promedio("DSP");
```

6. TRIGGERS

En MySQL, un disparador es un programa almacenado que se invoca automáticamente en respuesta a un evento como insertar , actualizar o eliminar que ocurre en la tabla asociada.



¿Por qué necesitamos los triggers?

Los TRIGGERS ofrecen una forma de ejecutar tareas programadas automáticamente.

¿Cómo crear triggers en MySql?

```
CREATE TRIGGER trigger_name
   [AFTER, BEFORE ] [INSERT,UPDATE,DELETE]
   ON table_name FOR EACH ROW
      trigger_body
```

Ejemplo de trigger

El caso de uso es guardar un registro de los eventos que ocurren en la tabla estudiante de mi base de datos.

creamos la tabla

```
CREATE TABLE log_estudiante (
          id INTEGER PRIMARY KEY NOT NULL AUTO_INCREMENT,
   accion VARCHAR (500) NOT NULL,
   fecha DATETIME DEFAULT CURRENT_TIMESTAMP
   )
```

Crear un trigger DESPUÉS DE INSERTAR un registro de la tabla estudiante

```
DELIMITER $$
CREATE TRIGGER log_estudiante AFTER INSERT ON estudiante

FOR EACH ROW

BEGIN

        INSERT INTO log_estudiante(accion) VALUES (CONCAT("Se insertó un nuevo estudiante a la base de datos, y se llama : ", NEW.nombre));

END

$$
```

Crear un trigger DESPUÉS DE ELIMINAR un registro de la tabla estudiante

```
DELIMITER $$
CREATE TRIGGER log_estudiante_delete AFTER DELETE ON estudiante
FOR EACH ROW

BEGIN

INSERT INTO  log_estudiante(accion) VALUES (CONCAT("Se eliminó un estudiante de la base de datos, y se llama : ", OLD.nombre));

END

$$
```

```

```

Cómo eliminar un trigger

```
DROP TRIGGER IF EXISTS log_estudiante;
```

DCL

7. GRANT

Permite dar permisos a un usuario nuevo a las bases de datos

Ejemplo:
Crear un usuario “dataanalystJuanito” con contraseña “juanito12345”

```
USE academico_datapath;
GRANT select
ON tutor
TO "dataanalystJuanito";
```

listar los permisos a un usuario

```
SHOW GRANTS FOR "dataanalystJuanito"
```

8. REVOKE

Eliminar permisos

```
REVOKE insert
ON tutor
FROM "dataanalystJuanito"
```