

HENRY

Data Science



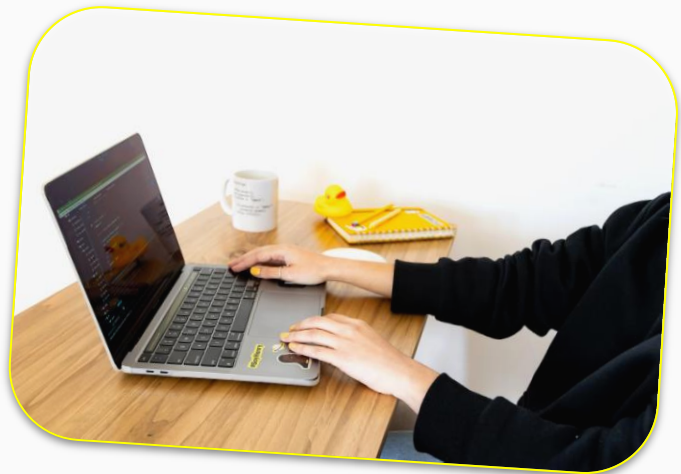
Clase 6

Introducción a DML





Agenda



- Introducción a DML y ORM.
- DML (INSERT, UPDATE, DROP, SELECT, WHERE).
- Operadores ariméticos.



OBJETIVOS DE LA CLASE

Al finalizar esta lecture estarás en la capacidad de...

- **Aplicar** las instrucciones básicas para realizar consultas en una base de datos relacional

DML



SQL no solo permite interactuar con las bases de datos para crear sus objetos, sino que además nos permite escribir y recuperar datos. Esto es posible gracias al **DML**.





**DML (INSERT, UPDATE, DROP,
SELECT, WHERE)**



Sentencias

INSERTAR

ELIMINAR

MODIFICAR

CONSULTAR





Insertar datos



Insertar datos (1)



Se detallan en orden los campos y los registros a ingresar en cada uno de ellos. Por cada sentencia **INSERT INTO se** puede declarar un **VALUES**. Es la opción menos eficiente para insertar datos en una tabla.



```
INSERT INTO alumnos(nombre,apellido,fecha_nacimiento,ciudad,pais,cedulaIdentidad)
VALUES('Maria ','Becerra','2000-4-1','Rosario',,'Argentina',38564122)
```

```
INSERT INTO alumnos(nombre,apellido,fecha_nacimiento,ciudad,pais,cedulaIdentidad)
VALUES('El ','Duki','1998-9-8','Santa Fé','Colombia',39874156)
```



Insertar datos (2)



Al igual que en la opción anterior, se detallan en orden los campos y los datos a ingresar en cada uno de ellos. Solo se declara una sentencia **INSERT INTO** y luego se listan en **VALUES** cada uno de los registros separados por coma.



```
INSERT INTO alumnos(nombre,apellido,fecha_nacimiento,ciudad,pais,cedulaIdentidad)
VALUES('Jeronimo','Gonzales','1988-8-9','Córdoba','Argentina',33687456),
('Ricardo','Lorenzo','1975-2-10','La Paz','Bolivia',20856147),
('Carlos','Principe','1985-6-11','Montevideo','Montevideo','Uruguay',25478369)
```



Insertar datos (3)

Se declara la sentencia **INSERT INTO** y el nombre de la tabla, luego se repite el **VALUES** de la opción 2.

La diferencia con las anteriores es que se deben insertar datos para todos los campos que forman parte de la tabla.



```
INSERT INTO alumnos
VALUES('Ernesto','Corvalan','1993-12-12','Caleta Olivia','Argentina',35879145),
('Roberto','Carlos','1997-1-13','Cuidad de México','México',37854698),
('Luis','Rodriguez','1976-3-14','Montevideo','Uruguay',20896369),
('Hernan','Crespo','1999-9-15','Santiago de Chile','Chile',39546178)
```



Modificar datos



Modificar datos (1)



Se modifica un **solo campo del registro.**



Es importante establecer cuidadosamente el registro a modificar, si no lo hacemos corremos el riesgo de modificar varios registros. Ese campo u otros pueden ser “filtrados” mediante la sentencia **WHERE.**



Modificar datos (2)



Se modifica **más de un campo del registro.**

```
UPDATE alumnos  
SET nombre = 'Juan', ciudad='Santa Fé'  
WHERE cedulaIdentidad = 33687456;
```



Eliminar datos



Eliminar datos



DELETE se utiliza para borrar registros, esto quiere decir que se eliminaran todos los campos de ese registro. Si verificamos un error en un solo campo no se debería eliminar el registro, sino realizar un **UPDATE**.

```
DELETE FROM alumnos WHERE idPerfil=1
```

```
DELETE FROM alumnos WHERE fechaInscripcion>'2021-01-08'
```



consultar datos



Consultar datos



Para consultar los datos en una tabla, se utiliza la sentencia SELECT, esta sentencia debe estar acompañada de manera obligatoria por FROM.



SELECT es una sentencia de proyección, donde puedes “solicitar” los campos a consultar.



En **FROM** se debe especificar cuál será la tabla a consultar.



La cláusula **WHERE** permite establecer criterios de filtrado o segmentación.



Operadores



Operadores aritméticos



Al igual que en python, dentro de SQL se pueden utilizar operadores para realizar cálculos en la sentencia

SELECT.



Es necesario que los campos sean de **tipos enteros** o decimales.



Operadores aritméticos

```
SELECT nombreProducto, subtotal + impuestos AS Total  
FROM productos
```

```
SELECT nombreColaborador, sueldo - retenciones AS SueldoNeto  
FROM staff
```

```
SELECT nombreProducto, precio * cantidad AS Total  
FROM productos
```




Operadores relacionales

```
SELECT *  
FROM productos  
WHERE fechaVenta = '2022-03-28'
```

```
SELECT *  
FROM productos  
WHERE fechaVenta != '2021-12-25'
```

```
SELECT *  
FROM productos  
WHERE fechaVenta > '2021-12-31'
```

```
SELECT *  
FROM productos  
WHERE fechaVenta < '2022-1-1'
```

```
SELECT *  
FROM alumnos  
WHERE fechaIngreso BETWEEN '2022-01-01' AND '2022-05-31'
```

```
SELECT *  
FROM alumnos  
WHERE pais IN ('Colombia','Mexico')
```

```
SELECT *  
FROM alumnos  
WHERE nombreModulo LIKE '%datos%'
```



Operadores lógicos

```
SELECT *  
FROM alumnos  
WHERE carrera IS NOT 'Full Stack'
```

```
SELECT *  
FROM alumnos  
WHERE pais = 'Colombia' AND pais = 'Mexico'
```

```
SELECT *  
FROM alumnos  
WHERE pais = 'Colombia' OR pais = 'Mexico'
```



¿Qué es ORM?



ORM

Un ORM es un **modelo de programación** que permite interactuar con las estructuras de una base de datos relacional (SQL Server, MySQL, PostgreSQL, etc.), lo que ayuda a simplificar y acelerar el desarrollo de aplicaciones.

Es a través de las aplicaciones como habitualmente se realizan los procesos de inserción, actualización, eliminación y consulta en una base de datos, el ORM permite a los **desarrolladores simplificar estos procesos.**



ORM

ORM en python, creando una tabla:

```
class Producto(db.Base):
    __tablename__ = 'producto'
    id = Column(Integer, primary_key=True)
    nombre = Column(String, nullable=False)
    precio = Column(Float)
    def __init__(self, nombre, precio):
        self.nombre = nombre
        self.precio = precio
    def __repr__(self):
        return f'Producto({self.nombre}, {self.precio})'
    def __str__(self):
        return self.nombre
```

HENRY

