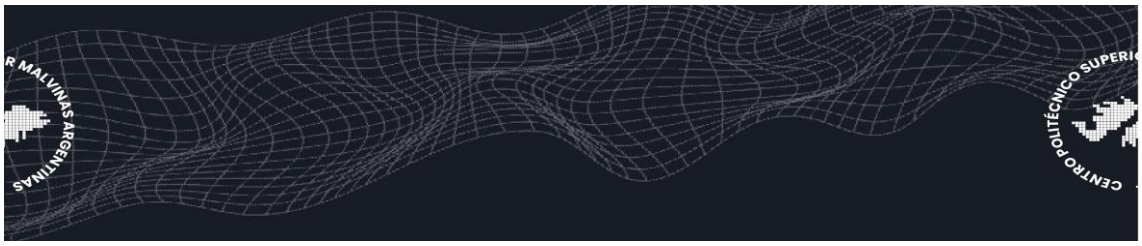


BASE DE DATOS

1° AÑO

Clase Nº 3: Introducción a lenguaje SQL

Contenido: Introducción a lenguaje sql, consultas básicas de datos, tipos de datos, funciones y operadores.

En la clase de hoy trabajaremos los siguientes temas:

- Qué es SQL y sus objetivos
- Donde se implementa el lenguaje SQL
- Funciones básicas de consultas
- Que es un tipo de datos y características del tipo de dato
- Qué son las funciones y operadores básicos en SQL

1. **Presentación:**

Bienvenidos a la **clase N° 3 del espacio BASE DE DATOS**

En esta clase desarrollaremos el tema **INTRODUCCIÓN AL LENGUAJE SQL** Este concepto es importante porque el SQL se usa para controlar todas las funciones que un sistema gestor de base de datos brinda a sus usuarios, proporcionando además un marco para crear la propia base de datos, gestionar su seguridad, actualizar sus contenidos, recuperar los datos y compartirlos entre diferentes usuarios. En general, el uso de SQL en una base de datos permite desarrollar habilidades importantes en el manejo y gestión de datos, así como en la toma de decisiones basada en datos y la optimización del rendimiento de una base de datos.

2.-Desarrollo y Actividades:

Los conceptos que estudiaremos en esta clase se relacionan con lo que vimos anteriormente en la clase N º2 Relaciones en base de datos que han estudiado anteriormente en conceptos básico de base de datos.

Si no recuerdas estos temas te invitamos a que visites estos Links/revises nuevamente la clase N° 2 conceptos básicos de Base de datos o veas este video de la clase 2 ,Si te quedan dudas sobre este tema siempre podes consultar en el foro o en las tutorías cuando nos encontremos.

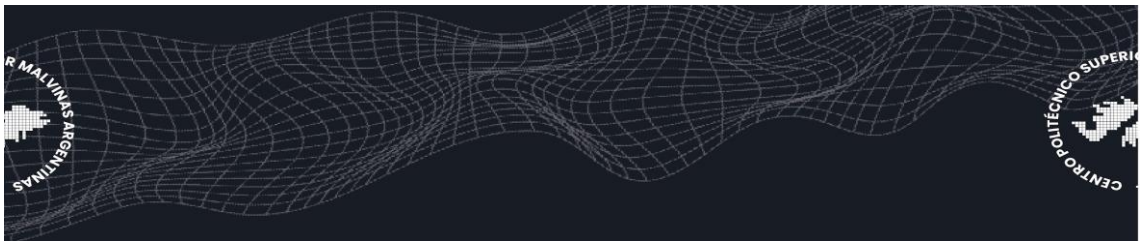
Instrucción a base de datos donde se desarrollaron los siguientes temas:

- Definiciones de llave primaria
- Qué son los atributos y tipos de atributos en la base de datos
- Qué son las entidades de la base de datos
- Que es una relación y tipo de relaciones

Lenguaje SQL

Introducción y objetivos

Las siglas SQL corresponden a *Structured Query Language*, un lenguaje estándar que permite manejar los datos de una base de datos relacional. La mayor parte de los SGBD relacionales implementan este lenguaje y mediante él se realizan todo tipo de accesos a la base de datos. En este capítulo se hace una presentación del lenguaje SQL, haciendo énfasis en la sentencia de



consulta de datos, la sentencia SELECT.
Al finalizar este capítulo, el estudiante debe ser capaz de:

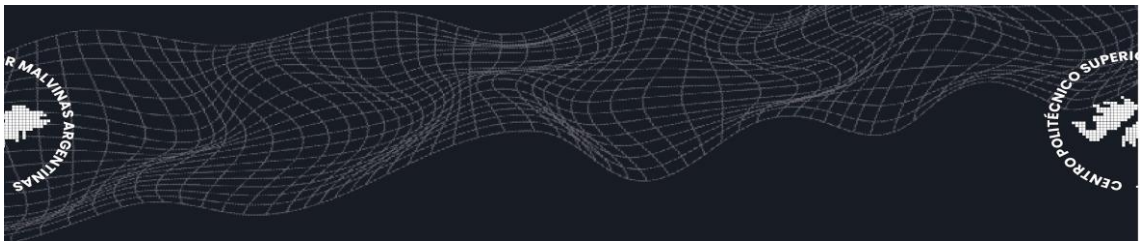
- Emplear la sentencia CREATE TABLE para crear tablas a partir de una especificación dada.
- Emplear las sentencias INSERT, UPDATE, DELETE para insertar, actualizar y borrar datos de tablas de una base de datos.
- Emplear la sentencia SELECT para responder a cualquier consulta de datos sobre una base de datos dada.
- Especificar una sentencia SELECT equivalente a otra dada que no haga uso de los operadores que se indiquen, con el objetivo de intentar acelerar el tiempo de respuesta.

Visión general del lenguaje

Normalmente, cuando un SGBD relacional implementa el lenguaje SQL, todas las acciones que se pueden llevar a cabo sobre el sistema se realizan mediante sentencias de este lenguaje. Dentro de SQL hay varios tipos de sentencias que se agrupan en tres conjuntos:

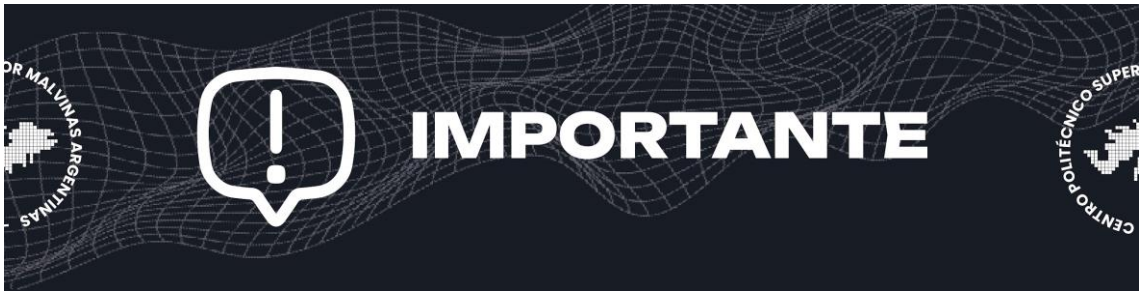
Sentencias de definición de datos: son las sentencias que permiten crear tablas, alterar su definición y eliminarlas. En una base de datos relacional existen otros tipos de objetos además de las tablas, como las vistas, los índices y los disparadores, que se estudiarán más adelante. Las sentencias para crear, alterar y eliminar vistas e índices también pertenecen a este conjunto.

Sentencias de manejo de datos: son las sentencias que permiten insertar datos en las tablas, consultarlos, modificarlos y borrarlos.



Sentencias de control: son las sentencias que utilizan los administradores de la base de datos para realizar sus tareas, como por ejemplo crear usuarios y concederles o revocar los privilegios.

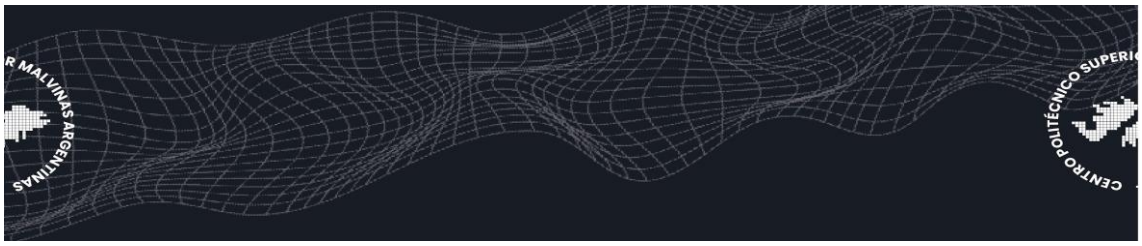
Las sentencias de SQL se pueden escribir tanto en mayúsculas como en minúsculas, y lo mismo sucede con los nombres de las tablas y de las columnas. Para facilitar la lectura de los ejemplos, se utilizará mayúsculas para las palabras clave del lenguaje y minúsculas para los nombres de tablas y de columnas. En los ejemplos se introducirán espacios en blanco para tabular las expresiones. Las sentencias de SQL terminan siempre con el carácter punto y coma (;).



Creación de tablas

Para crear una tabla en una base de datos se utiliza la sentencia CREATE TABLE. Su sintaxis es la siguiente:

```
CREATE TABLE nombre_tabla (
  columna1 tipo_dato1,
  columna2 tipo_dato2,
  columna3 tipo_dato3,
```



```
  ...
  columnaN tipo_datoN
);
```

Ejemplo de creación de una tabla:

Supongamos que queremos crear una tabla llamada "Usuarios" con las columnas "ID" (entero), "Nombre" (cadena de caracteres) y "Edad" (entero). La sentencia CREATE TABLE sería la siguiente:

```
CREATE TABLE Usuarios (
  ID INT,
  Nombre VARCHAR(50),
  Edad INT
);
```

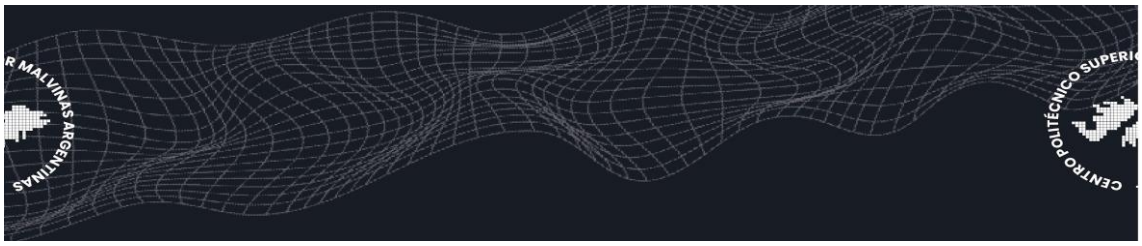
A continuación, se especifica el significado de cada identificador y de cada cláusula de la sentencia CREATE TABLE:

nombre_tabla: nombre de la nueva tabla.

nombre_columna: nombre de una columna de la tabla.

tipo datos: tipo de datos de la columna.

DEFAULT expr: asigna un valor por defecto a la columna junto a la que aparece; este valor se utilizará cuando en una inserción no se especifique valor para la columna.



CONSTRAINT nombre restricción: a las restricciones que se definen sobre columnas y sobre tablas se les puede dar un nombre (si no se hace, el sistema generará un nombre automáticamente).

NOT NULL: la columna no admite nulos.

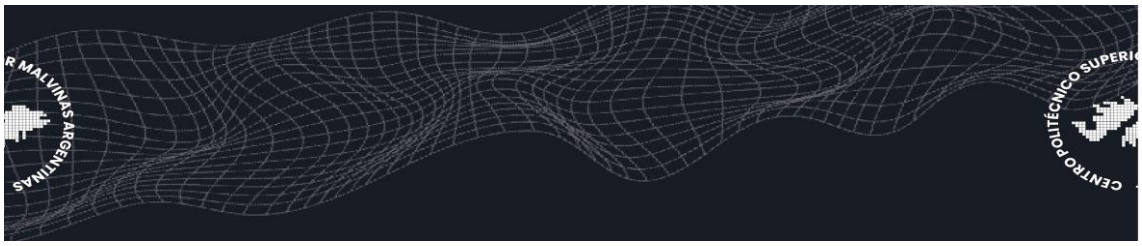
NULL: la columna admite nulos (se toma por defecto si no se especifica

NOT NULL).

UNIQUE especificada como restricción de columna indica que la columna sólo puede contener valores únicos. UNIQUE (nombre_columna [,...]) especificada como restricción de tabla indica que el grupo de columnas sólo pueden contener grupos de valores únicos. Mediante esta cláusula se especifican las claves alternativas.

PRIMARY KEY especificada como restricción de columna o bien PRIMARY KEY (nombre_columna[,...]) especificada como restricción de tabla indica la columna o el grupo de columnas que forman la clave primaria de la tabla. Los valores de la clave primaria, además de ser únicos, deberán ser no nulos.

Inserción de datos

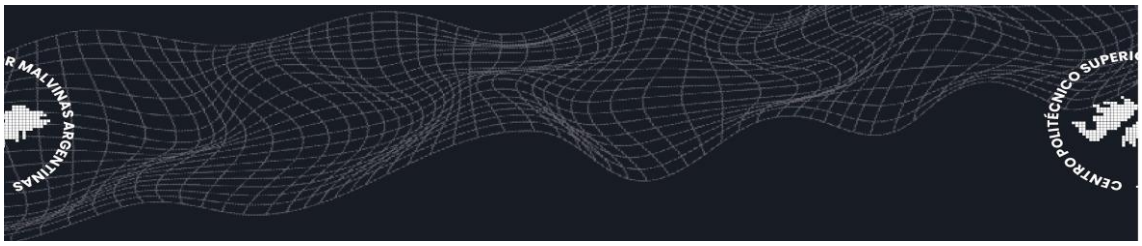


Una vez creada una tabla podemos introducir datos en ella mediante la sentencia INSERT, como se muestra en los siguientes ejemplos:

```
INSERT INTO facturas(codfac,fecha, codcli,codven,iva,dto ) VALUES(6600,
  '30/04/2007',111, 55, 0, NULL);

INSERT INTO lineas_fac(codfac,linea,cant,codart, precio,dto)
VALUES(6600, 1, 4, 'L76425',3.16, 25 );

INSERT INTO lineas_fac(codfac,linea,cant,codart, precio,dto)
VALUES(6600, 2, 5, 'B14017',2.44, 25 );

INSERT INTO lineas_fac(codfac,linea,cant,codart, precio,dto)
VALUES(6600, 3, 7, 'L92117',4.39, 25 );
```

Mediante estas sentencias se ha introducido la cabecera de una factura y tres de sus líneas. Nótese que tanto las cadenas de caracteres como las fechas, se introducen entre comillas simples. Para introducir nulos se utiliza la expresión NULL.Algunos SGBD relacionales permiten insertar varias filas en una mismatabla mediante una sola sentencia INSERT, y realizan las inserciones de un modo más eficiente que si se hace mediante varias sentencias independientes. Así, la tres inserciones que se han realizado en la tabla LINEAS_FAC también se pueden realizar mediante la siguiente sentencia:

```
INSERT INTO lineas_fac(codfac,linea,cant,codart, precio,dto)
VALUES(6600, 1, 4, 'L76425',3.16, 25 ), (6600, 2,
  5, 'B14017',2.44, 25 ), (6600, 3, 7, 'L92117',4.39,
  25 );
```



Consulta de datos

Una vez se ha visto cómo almacenar datos en la base de datos, interesa conocer cómo se puede acceder a dichos datos para consultarlos. Para ello se utiliza la sentencia SELECT. Por ejemplo:

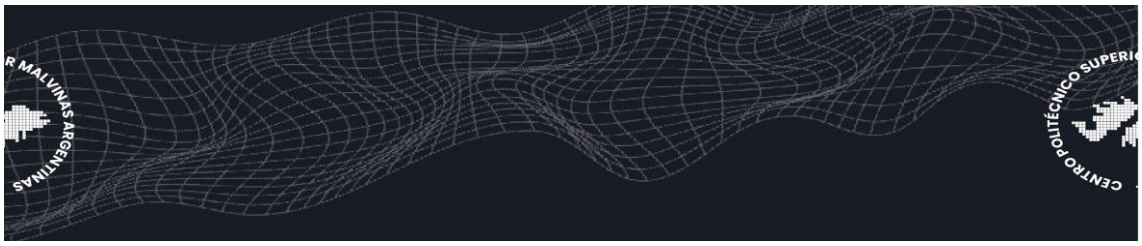
```
SELECT * FROM facturas;
```

En primer lugar aparece la palabra SELECT, que indica que se va a realizar una consulta. A continuación, el * indica que se desea ver el contenido de todas las columnas de la tabla consultada. El nombre de esta tabla es el que aparece tras la palabra FROM, en este caso, la tabla facturas.

Esta sentencia es, sin lugar a dudas, la más compleja del lenguaje de manejo de datos y es por ello que gran parte de este capítulo se centra en su estudio.

Actualización y eliminación de datos

Una vez insertados los datos es posible actualizarlos o eliminarlos mediante las sentencias UPDATE y DELETE, respectivamente. Para comprender el funcionamiento de estas dos sentencias es imprescindible conocer bien el funcionamiento de la sentencia SELECT. Esto es así porque para poder actualizar o eliminar datos que se han almacenado es preciso encontrarlos antes.



Y por lo tanto, la cláusula de estas sentencias que establece las condiciones de búsqueda de dichos datos (WHERE) se especifica del mismo modo que las condiciones de búsqueda cuando se hace una consulta.

Sin embargo, antes de pasar al estudio de la sentencia SELECT se muestran algunos ejemplos de estas dos sentencias.

UPDATE facturas:

UPDATE facturas SET dto = 0 SET codven = 105 WHERE dto IS NULL; WHERE codven IN (SELECT codven FROM vendedores WHERE codjefe = 105);

DELETE FROM facturas:

DELETE FROM facturas WHERE codcli = 333; WHERE iva = (SELECT MIN(iva) FROM facturas);

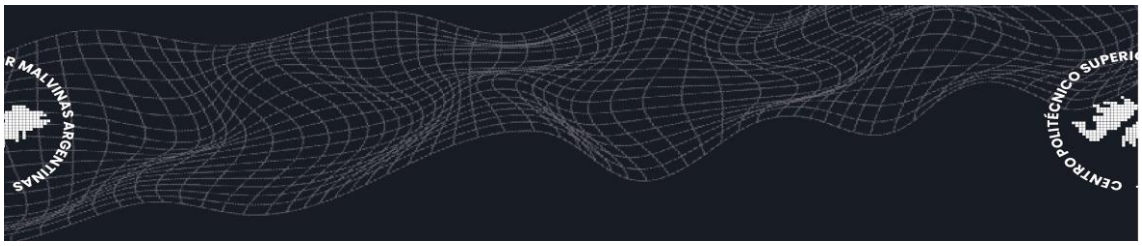
Estructura básica de la sentencia SELECT

La sentencia SELECT consta de varias cláusulas. A continuación se muestran algunas de ellas:**SELECT [DISTINCT] { * | columna [, columna] }**

FROM tabla [WHERE condición de búsqueda] [ORDER BY columna [ASC | DESC][,columna [ASC | DESC]];

El orden en que se tienen en cuenta las distintas cláusulas durante la ejecución y la función de cada una de ellas es la siguiente:

FROM: especifica la tabla sobre la que se va a realizar la consulta.



WHERE: si sólo se debe mostrar un subconjunto de las filas de la tabla, aquí se especifica la condición que deben cumplir las filas a mostrar; esta condición será un predicado booleano con comparaciones unidas por AND/OR.

SELECT: aquí se especifican las columnas a mostrar en el resultado; para mostrar todas las columnas se utiliza *.

DISTINCT: es un modificador que se utiliza tras la cláusula SELECT para que no se muestren filas repetidas en el resultado (esto puede ocurrir sólo cuando en la cláusula SELECT se prescinde de la clave primaria de la tabla o de parte de ella, si es compuesta).

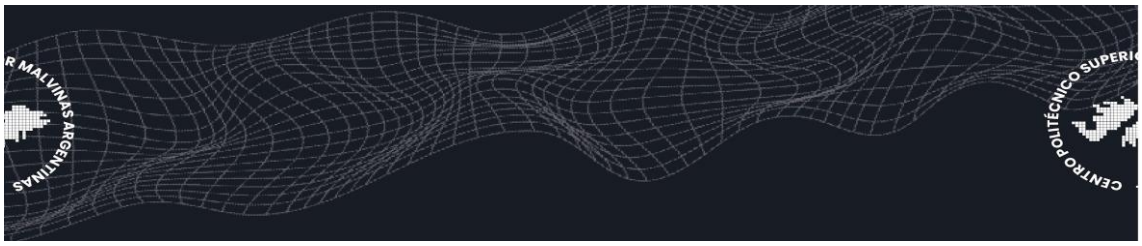
ORDER BY: se utiliza para ordenar el resultado de la consulta.

La cláusula ORDER BY, si se incluye, es siempre la última en la sentencia SELECT. La ordenación puede ser ascendente o descendente y puede basarse en una sola columna o en varias.La sentencia del siguiente ejemplo muestra los datos de todos los clientes, ordenados por el código postal, descendentemente. Además, todos los clientes de un mismo código postal aparecerán ordenados por el nombre, ascendentemente.

SELECT *FROM clientes ORDER BY codpostal DESC, nombre;

Expresiones en SELECT y WHERE

En las cláusulas SELECT y WHERE, además de columnas, también se pueden incluir expresiones que contengan columnas y constantes, así como funciones. Las columnas y expresiones especificadas en la cláusula SELECT se pueden renombrar al mostrarlas en el resultado mediante AS.Si el resultado de una consulta se debe mostrar ordenado según el valor de una expresión de la



cláusula SELECT, esta expresión se indica en la cláusula ORDER BY mediante el número de orden que ocupa en la cláusula SELECT.

```
SELECT precio, ROUND(precio * 0.8, 2) AS rebajado FROM artículos
ORDER BY 2;
```

Nulos

Cuando no se ha insertado un valor en una columna de una fila se dice que ésta es nula. Un nulo no es un valor: un nulo implica ausencia de valor. Para saber si una columna es nula se debe utilizar el operador de comparación IS NULL y para saber si no es nula, el operador es IS NOT NULL.

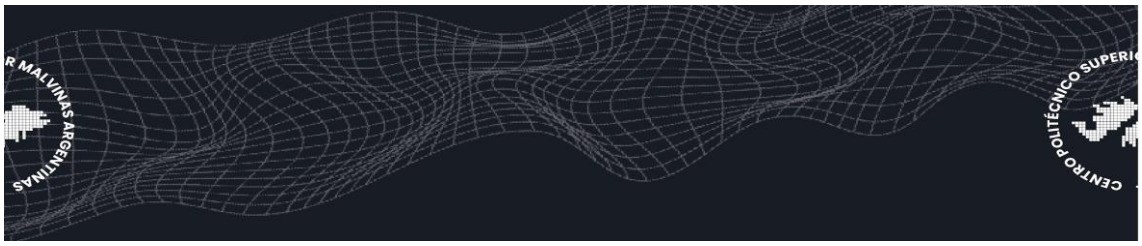
Tipos de datos

Los tipos de datos disponibles se deben consultar en el manual del SGBD relacional que se esté utilizando. Puesto que las prácticas de las asignaturas para las que se edita este libro se realizan bajo PostgreSQL, se presentan aquí los tipos de datos que se han usado en este SGBD para crear las tablas. Todos ellos pertenecen al estándar de SQL.

VARCHAR(n): Cadena de hasta n caracteres.

NUMERIC(n,m): Número con n dígitos, de los cuales m se encuentran a la derecha del punto decimal.

DATE: Fecha formada por día, mes y año. Para guardar fecha y hora se debe utilizar el tipo TIMESTAMP.



BOOLEAN: Aunque este tipo no se ha utilizado en la base de datos de prácticas, es interesante conocer su existencia. El valor verdadero se representa mediante TRUE y el falso mediante FALSE. Cuando se imprimen estos valores, se muestra el carácter ’t’ para verdadero y el carácter ’f’ para falso.Hay que tener siempre en cuenta que el nulo no es un valor, sino que implica ausencia de valor. El nulo se representa mediante NULL y cuando se imprime no se muestra nada

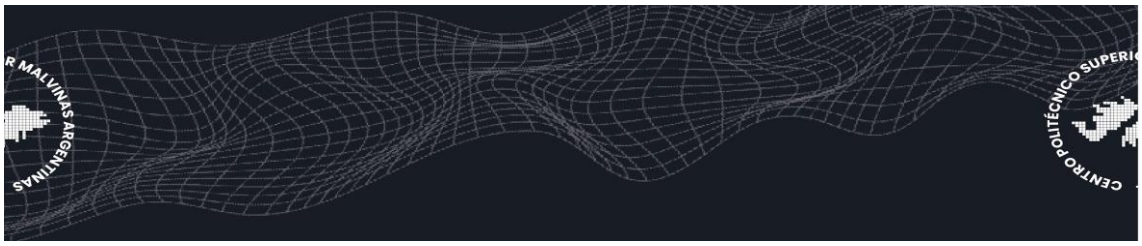
Funciones y operadores

Operadores lógicos

Los operadores lógicos son AND, OR y NOT. SQL utiliza una lógica booleana de tres valores y la evaluación de las expresiones con estos operadores es la que se muestra en la siguiente tabla:

Operadores de comparación

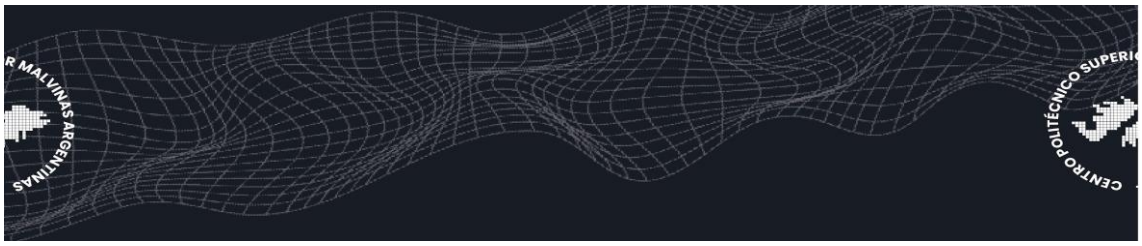
Operador	Descripción	Ejemplo	Resultado
`AND`	Ambos operandos deben ser verdaderos para que la expresión sea verdadera.	`SELECT * FROM clientes WHERE edad > 18 AND genero = 'M';`	Devuelve solo los clientes masculinos mayores de 18 años.
`OR`	Al menos uno de los operandos debe ser verdadero para que la expresión sea verdadera.	`SELECT * FROM pedidos WHERE estado = 'Pendiente' OR estado = 'En curso';`	Devuelve todos los pedidos con estado "Pendiente" o "En curso".
`NOT`	Invierte el valor lógico del operando.	`SELECT * FROM empleados WHERE NOT activo;`	Devuelve solo los empleados que no están activos.



`=`	Igual que	`SELECT * FROM empleados WHERE nombre = 'Ana';`	Devuelve solo la empleada llamada "Ana".
`<>`	Distinto de	`SELECT * FROM usuarios WHERE email <> 'info@empresa.com';`	Devuelve todos los usuarios con correo electrónico diferente a "info@empresa.com".
`a BETWEEN x AND y`	Entre un rango	`SELECT * FROM fechas WHERE fecha BETWEEN '2023-12-01' AND '2024-02-29';`	Devuelve las fechas entre el 1 de diciembre de 2023 y el 29 de febrero de 2024 (inclusive).
`a NOT BETWEEN x AND y`	No está en un rango	`SELECT * FROM numeros WHERE numero NOT BETWEEN 10 AND 20;`	Devuelve todos los números que no estén entre 10 y 20 (inclusive).

`a IS NULL`	Es nulo	`SELECT * FROM productos WHERE descripcion IS NULL;`	Devuelve solo los productos que no tienen descripción.
`a IS NOT NULL`	No es nulo	`SELECT * FROM clientes WHERE telefono IS NOT NULL;`	Devuelve solo los clientes que tienen un número de teléfono registrado.
`a IN (v1, v2, ...)`	Está en una lista	`SELECT * FROM usuarios WHERE pais IN ('España', 'Francia', 'Italia');`	Devuelve solo los usuarios que residen en España, Francia o Italia.

Actividad 1



Se proporcionarán diferentes consultas SQL y los alumnos deberán analizar si la consulta está escrita de manera correcta o si contiene errores. Deben indicar si la consulta es válida o inválida y, en caso de ser inválida, señalar el error presente. A continuación, se presentan ejemplos de consultas SQL para verificar.

1.-Consulta: **SELECT * FROM Customers WHERE Country = 'USA' AND City = 'New York';**

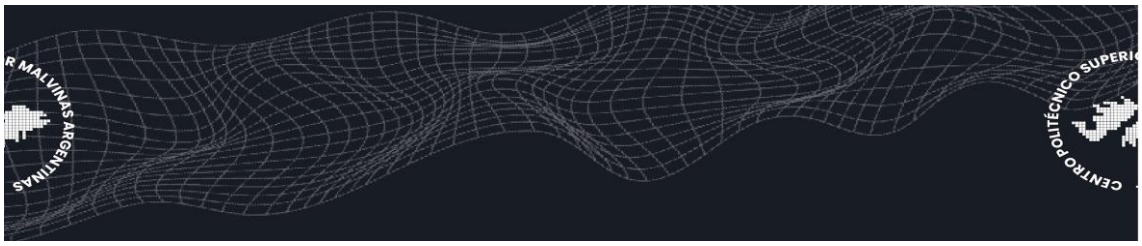
2.-Consulta: **SELECT ProductName, Price FROM Products WHERE Category = 'Electronics' ORDER BY Price DESC;**

3.-Consulta: **SELECT FirstName, LastName, Age FROM Customers WHERE Age > 30 AND Age < 50 AND City = 'London';**

4.-Consulta: **SELECT * FROM Orders WHERE OrderDate = '2021-05-15'**

5.-Consulta: **SELECT ProductName, Category, Price FROM Products WHERE Price > 100 AND StockAvailability = true;**

6.-Consulta: **SELECT CustomerName, OrderDate, TotalAmount FROM Customers WHERE OrderDate = 2022-03-10;**



7.-Consulta: **SELECT * FROM Products WHERE Category = 'Electronics' AND Price >;**

8.-Consulta: **SELECT ProductName, SUM(Quantity) FROM Sales GROUP BY ProductName;**

9.-Consulta: **SELECT * FROM Customers WHERE Country = 'Germany' OR Country = 'France' AND Age > 25;**

10.-Consulta: **SELECT AVG(Price) FROM Products WHERE Category = 'Clothing' ORDER BY Price ASC;**

Actividad 2

Un videoclub que alquila películas en video almacena la información de sus películas en alquiler en una tabla llamada "películas".

1- Elimine la tabla, si existe:

```
if object_id('peliculas') is not null  drop table peliculas;
```

2- Cree la tabla:

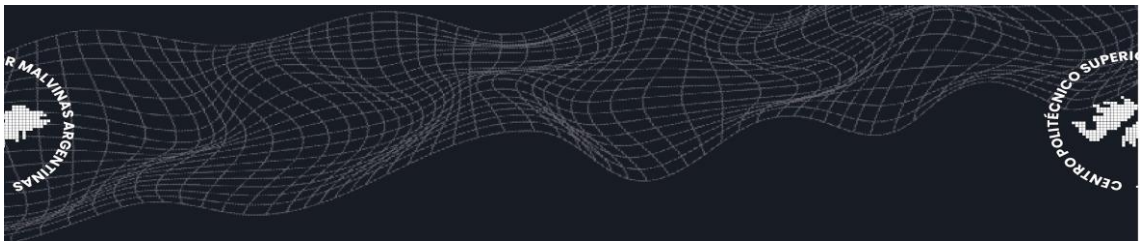
```
create table peliculas( titulo varchar(20),actor varchar(20),duracion integer, cantida:integer);
```

3- Vea la estructura de la tabla (sp_columns).

4- Ingrese a los siguientes registros:

```
insert into peliculas (titulo, actor, duracion, cantidad) values ('Mision imposible','Tom Cruise',180,3);
```

```
insert into peliculas (titulo, actor, duracion, cantidad) values ('Mision imposible 2','Tom Cruise',190,2);
```



insert into peliculas (titulo, actor, duracion, cantidad) values ('Mujer bonita','Julia Roberts',118,3);

insert into peliculas (titulo, actor, duracion, cantidad) values ('Elsa y Fred','China Zorrilla',110,2);

5- Realice un "select" mostrando solamente el título y actor de todas las películas

6- Muestre el título y duración de todas las películas

7- Muestre el título y la cantidad de copias

3. Actividad Integradora de Cierre:

Trabaje con la tabla "agenda" en la que registra los datos de sus amigos.

1- Elimine "agenda", si existe:

if object_id('agenda') is not null drop table agenda;

2- Cree la tabla, con los siguientes campos:

apellido (cadena de 30), nombre (cadena de 20), domicilio (cadena de 30) y teléfono (cadena de 11).

3- Visualice la estructura de la tabla "agenda".

4- Ingrese los siguientes registros:

Acosta, Ana, Colon 123, 4234567;

Bustamante, Betina, Avellaneda 135, 4458787;

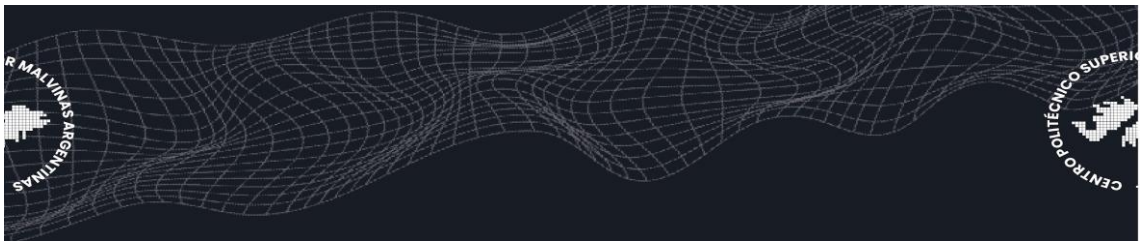
Lopez, Hector, Salta 545, 4887788;

Lopez, Luis, Urquiza 333, 4545454;

Lopez, Marisa, Urquiza 333, 4545454.

5- Seleccione todos los registros de la tabla

6- Seleccione el registro cuyo nombre sea "Marisa" (1 registro)



7- Seleccione los nombres y domicilios de quienes tengan apellido igual a "Lopez" (3 registros)

8- Muestre el nombre de quienes tengan el teléfono "4545454" (2 registros)

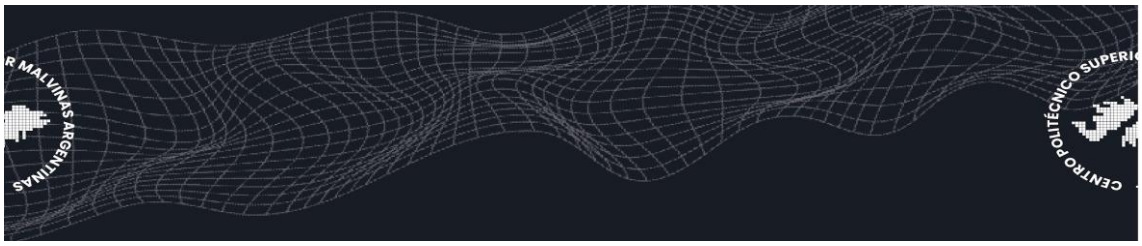
4. Cierre:

Al utilizar SQL en una base de datos, se pueden aprender diversas habilidades y conceptos importantes, entre ellos:

1.-Qué es SQL y sus objetivos: Aprenderás qué es SQL (Structured Query Language), un lenguaje de programación diseñado para administrar y manipular bases de datos relacionales. Comprenderás los objetivos principales de SQL, que incluyen la gestión de datos, la consulta y recuperación de información, la manipulación de datos y la administración de bases de datos.

2.-Donde se implementa el lenguaje SQL: Conocerás los diferentes sistemas y entornos en los que se implementa el lenguaje SQL. Esto puede incluir sistemas de gestión de bases de datos relacionales (RDBMS) como Oracle, MySQL, SQL Server, PostgreSQL, entre otros, así como otros programas y herramientas que admiten SQL para interactuar con bases de datos.

3.-Funciones básicas de consultas: Aprenderás las funciones básicas de consulta en SQL, que te permiten recuperar información específica de una base de datos. Esto incluirá conceptos como SELECT (selección de columnas), FROM (tabla de origen), WHERE (condiciones de filtrado),

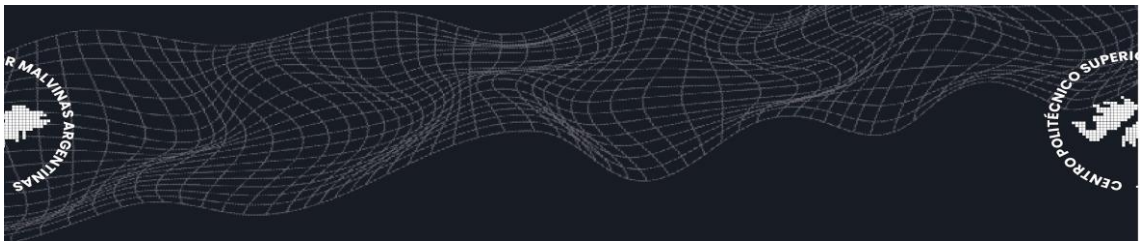


ORDER BY (ordenamiento) y otras cláusulas importantes en las consultas SQL.

4.-Qué es un tipo de datos y características del tipo de dato: Entenderás qué son los tipos de datos en SQL y cómo se definen. Aprenderás acerca de los tipos de datos más comunes, como VARCHAR (texto), INTEGER (números enteros), DATE (fechas), entre otros. Además, conocerás las características y restricciones asociadas a cada tipo de dato, como longitud, precisión, formato y limitaciones.

5.-Qué son las funciones y operadores básicos en SQL: Descubrirás las funciones y operadores básicos que están disponibles en SQL para realizar cálculos, manipular datos y realizar operaciones lógicas. Esto incluirá funciones como SUM, COUNT, AVG, MAX, MIN, así como operadores aritméticos, de comparación y lógicos que te permiten realizar tareas más avanzadas en tus consultas y manipulaciones de datos.

Estos temas te proporcionarán una base sólida para comprender y utilizar SQL como lenguaje de consulta y manipulación de bases de datos relacionales.



5.-Bibliografía Obligatoria:

- Marquez, M. (2011) : Base de datos. Editorial Universitat Jaume.
- Catheren M. R. (2009): Base de Datos. Edición McGraw Hill
- Cita de página Web: https://www.codigofuente.org/comandos-comunes-en-consultas-sql/
- Cita de página Web: https://tursos.com/10-comandos-sql-que-te-pueden-ser-de-utilidad/