



*Prof. Flavio Bevilacqua*

*Mail: [flabevy88@gmail.com](mailto:flabevy88@gmail.com)*

*Linkedin: [www.linkedin.com/in/flabevy](http://www.linkedin.com/in/flabevy)*



# ***MODULOS***

# ***MODULOS DEL LENGUAJE***

➤ Al ser un lenguaje declarativo orientado a Base de Datos, es necesario contar con una amplia gama de expresiones que permitan combinar la manipulación y la definición de los datos, así como también generar operaciones con los datos almacenados.

➤ En consecuencia, se derivan desde SQL 4 sublenguajes con diferentes finalidades, como expresiones definidas.





# MODULOS Y SUBLINGUAJES

**ESTRUCTURA DE:**  
Base de datos  
Tablas  
Campos  
Indíces  
Restricciones  
Código

DDL

**REGISTRO DE DATOS:**  
Añadir registros  
Modificar registros  
Borrar registros  
Consultarlos

DML

DCL

TCL

**PERMISOS**  
GRANT  
REVOKE

**TRANSACCIONES**  
TRANSACTION  
COMMIT  
ROLLBACK



# **DATA MANIPULATION LANGUAGE**



# DATA MANIPULATION LANGUAGE

- Las Instrucciones del Lenguaje de Datos DML hace referencia a la Manipulación de los datos, siendo su principal funcionalidad operar sobre los registros.
- El Modulo DML esta orientado a la consulta de los datos, como así también la insercion, actualizacion y eliminacion de filas dentro de las tablas.
- Veamos sus comandos principales:



# ***DATA MANIPULATION LANGUAGE***

## ***QUERIES***

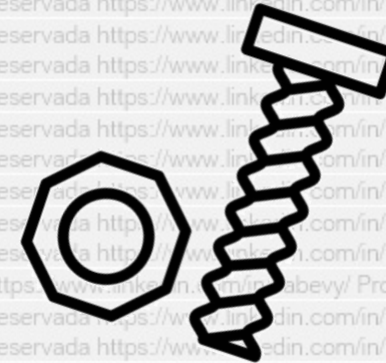
➤ Una Query es una sentencia DML orientada a la consulta sobre una base de datos.

➤ **COMANDOS PRINCIPALES:**


➤ **SELECT**

➤ **FROM**

➤ **WHERE**





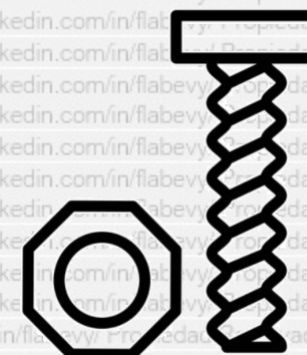


# ***SENTENCIAS DE CONSULTAS***



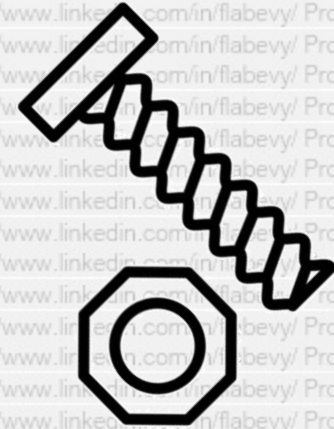
# ***SELECT***

- Una cláusula **SELECT** se usa para especificar los nombres de los campos que contienen los datos que quiere usar en una consulta. También puede usar operadores para hacer referencias a toda la tabla en su conjunto o expresiones capaces de generar campos calculados.
- Sintácticamente siempre precede a los Atributos que necesitamos como resultados.
- La aplicación de estos comandos no tiene incidencia en la base de datos o tablas reales. (No generan ninguna modificación)



# FROM

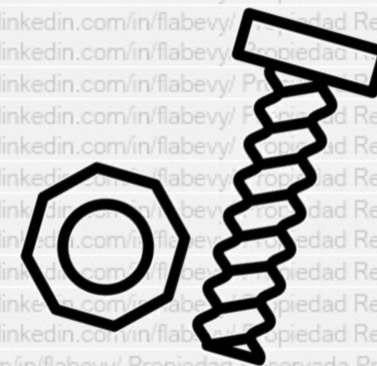
- Este comando es utilizado para especificar la tabla de la cual se van a seleccionar los registros que necesitamos como consulta.
- Siempre que haremos referencia a una tabla en el FROM, la misma será la tabla principal de nuestra consulta.
- El comando **FROM** siempre procede al comando SELECT, pero a diferencia de su lectura en el editor, esta sentencia es la primera en ejecutarse implícitamente por el motor de SQL.





# WHERE

- El comando **WHERE** es utilizado para determinar filtros en nuestras consultas.
- Los criterios que serán indicados en el **WHERE** mayormente tienden a ir acompañados por Operadores Comparativos, como Operadores Lógicos. Aun así, también es posible aplicar Funciones como Operadores Aritméticos.
- El comando **WHERE** es común utilizarlo en Sentencias de Consulta, como así también en Sentencias de Actualización y Eliminación de Registros.





# ***OPERADORES COMPARATIVOS***



# ***OPERADORES COMPARATIVOS***

- Los Operadores de Comparacion validan si dos expresiones son iguales. Se pueden utilizar en todas las expresiones, excepto en las de los tipos TEXT. En la siguiente tabla se presenta un glosario de sus diferentes variantes.

<i><b>OPERADOR</b></i>	<i><b>DESCRIPCION</b></i>
=	Igual a..
>	Mayor que..
<	Menor que..
>=	Mayor o Igual que..
<=	Menor o Igual que..
<>	No es igual a..



# ***OPERADORES ARITMETICOS***



# ***OPERADORES ARITMETICOS***

- Los Operadores Aritmeticos nos permiten realizar operaciones matemáticas básicas como sumas, restas, multiplicación o divisiones.
- Estos Operadores pueden ser utilizados tanto en sentencias DML de cualquier índole como así también en Funciones, StoreProcedure, Triggers y otros Objetos.

<i><b>OPERADOR</b></i>	<i><b>DESCRIPCION</b></i>
+	Suma
-	Menos
*	Multiplicacion
/	Division



# ***OPERADORES LOGICOS***



# OPERADORES LOGICOS

- *Los Operadores Logicos nos permiten evaluar expresiones, el resultado varia en función al operador lógico utilizado.*
- *Muchas de estos operados admiten la variante **NOT** lo que invierte el valor lógico en el resultado.*

## AND

*Opera únicamente sobre aquellos registros que cumplen con todas las condiciones a la vez.*

## OR

*Opera sobre aquellos registros que cumplen al menos con una condición de las mencionadas en la sintaxis.*

## IS NULL – IS NOT NULL

*IS NULL es un operador que nos permite retornar resultados NULOS.  
Mayormente se utiliza con la variante NOT NULL para no tener en cuenta los Valores Nulos en el resultado.*



# OPERADORES LOGICOS

➤ *Los Operadores Logicos nos permiten evaluar expresiones, el resultado varia en función al operador lógico utilizado.*

## **BETWEEN**

*Dicho operador nos permite recuperar los registros según el intervalo de valores de un campo. Se tiende a utilizar con valores numéricos como de tiempo.*

## **LIKE - LIKE NOT**

*El operador LIKE nos permite comparar una expresión de cadenas, por lo que siempre acompañados de comillas simples. A su vez, el símbolo de % acompaña en caso de no tener con exactitud algunos caracteres adicionales del valor. Permite el uso de comodines como "%" o "\_"*

## **IN - IN NOT**

*Este operador devuelve aquellos registros cuyo campo indicado coincide con alguno de los indicados en una lista o conjunto de valores. Permite la Variante IN NOT el cual invierte el valor lógico.*



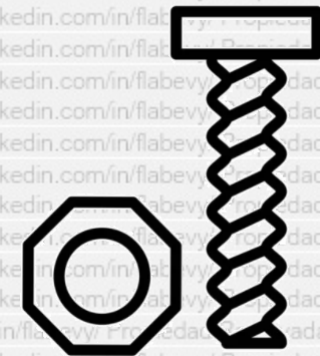




# ***OTROS COMANDOS***

# AS

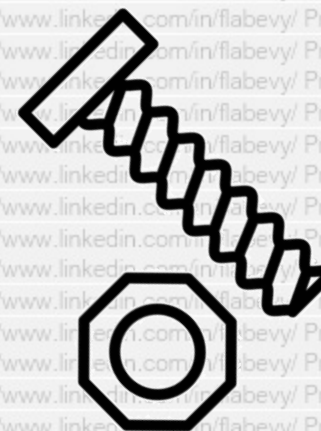
- Esta sentencia nos permite establecer un **ALIAS**, tanto a una tabla como a una columna, y se utiliza para simplificar las sentencias SQL cuando los nombres de tablas o columnas son largos o complicados.
- Es utilizado prácticamente de manera obligada cuando se trabajan con Joins o multiples columnas.
- A su vez, cuando realizamos operaciones matemáticas en alguno de nuestras campos, con el objetivo de aplicar una Columna Calculada, podemos asignarle un nombre a este nuevo campo que obtendremos en nuestra consulta.





# TOP (X)

- Este comando es utilizado para limitar la cantidad de registros que vendrán como resultado de nuestra consulta.
- Sintácticamente se indica luego del SELECT debe ser acompañado por un valor (x) que haga referencia a la cantidad de registros esperados como resultado.
- Al utilizarlo en compañía de un ORDER BY podremos especificar nuestra consulta mas en profundidad, incluso trayendo un único resultado como respuesta.



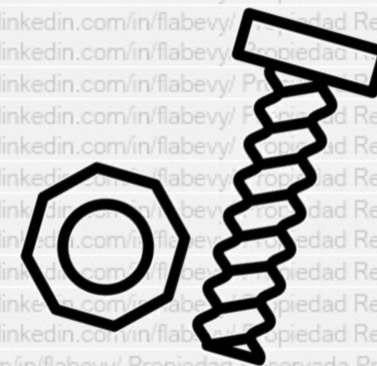
# ***DISTINCT***

- El comando DISTINCT nos permite omitir los registros duplicados en el resultado que obtenemos en base a nuestra consulta.
- El DISTINCT se posiciona posterior al SELECT y antes que todos los campos que pretendemos obtener en nuestra tabla de respuesta.
- Esta sentencia evalúa cada uno de los registros en su completitud y cuando reconoce que hay filas exactamente iguales procede a no duplicarlas.
- Dado a que solo se atañe al resultado, el DISTINCT no opera sobre aquellas columnas que no fueron mencionadas en el SELECT.



# ORDER BY

- El comando **ORDER BY** es utilizado para ordenar los registros seleccionados de acuerdo con un orden específico.
- Dicha sentencia siempre se indica al final de todo el bloque de consulta, mencionando el campo (o los campos) por los cuales se desea ordenar el resultado
- Dicho comando tiende a ir acompañado por las palabras reservadas **ASC** o **DESC** para invertir el criterio de organización en el orden de los registros.



FIN