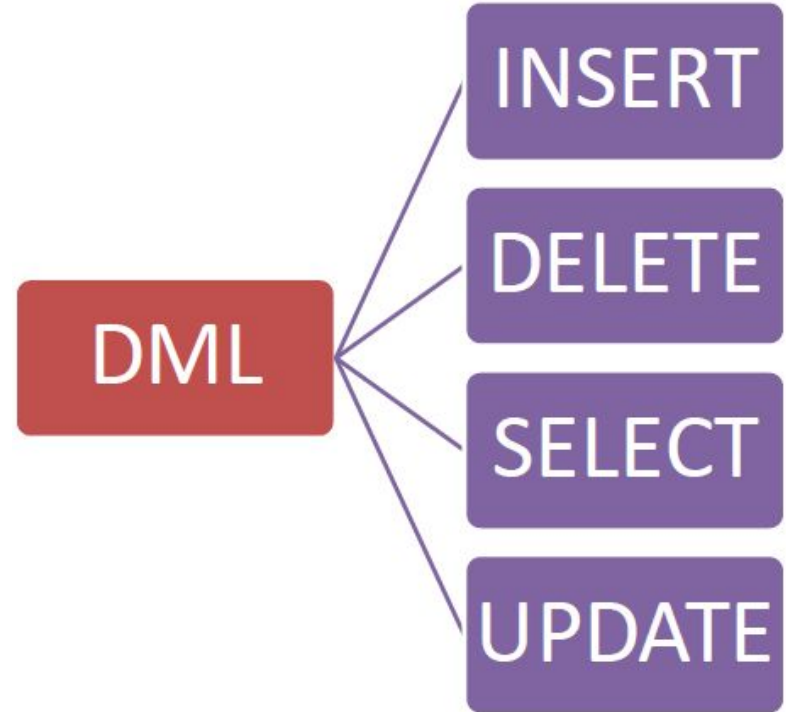


# SQL - DML

## Apunte teórico complementario

# DML

Este es el conjunto de sentencias que se encargan de la **inserción**, **actualización y eliminación** de los **datos** en las tablas de una base de datos.



La cláusula *insert* nos permitirá **insertar datos** en una tabla.



*INSERT*

Además necesitamos otras 2 sentencias ***into*** y ***values***:

*into* indica el destino de la inserción (puede ser una tabla o determinadas columnas de la misma)

*values* señala los valores a insertar.

# SINTAXIS

```
insert into nombre_tabla (campo1, campo2, campo3 .... campoN)  
values (valor1, valor2, valor3.... valorN);
```

```
insert into clientes(N_Cliente, Nombre, Rubro, Telefono)  
values (264, 'Juan', 'Textil', 1178372360);
```

```
insert into clientes  
values (235, 'Maximo', 'Textil ', 1178372360,'max.grindewald@gmail.com');
```

# UPDATE

La cláusula *update* permite **modificar** el **contenido** de un registro o fila.

Para hacer la actualización de un registro se necesita completar la instrucción con las sentencias **set** y **where**, que especifican el nuevo valor y el registro a modificar.

# SINTAXIS

```
update nombre_tabla  
SET nombre_campo_a_reemplazar = valor_nuevo  
WHERE nombre_campo_condicional = valor_condicion;
```

**Update** Clientes

```
set Nombre = 'Carlos'  
where N_Cliente = 562;
```

# DELETE

La cláusula *delete* se utiliza para eliminar determinadas filas de una tabla o todas las filas de la tabla

# SINTAXIS

## Eliminación de un registro:

```
delete from nombre_tabla  
where nombre_campo = valor_de_condicion;
```

```
delete from clientes where N_cliente = 562;
```

## Eliminación de varios registros:

```
delete from nombre_tabla where nombre_campo = valor_de_condicion1 or nombre_campo =  
valor_de_condicion2;
```

```
delete from clientes  
where N_cliente = 562 or N_Cliente = 398;
```



Eliminación de todos los registros:

```
delete from nombre_tabla;
```

```
delete from clientes;
```



# EJEMPLOS

**Para consultar todos los campos de una tabla:**

```
SELECT *  
FROM Clientes
```

**Para consultar determinados campos de una tabla:**

```
SELECT N_Cliente, Nombre, Telefono  
FROM Clientes
```

**Para consultar sólo los campos que cumplen determinada condición:**

```
SELECT N_Cliente, Nombre, Telefono  
FROM Clientes WHERE Nombre = 'Maria'
```

WHERE  
AS / DISTINCT

# WHERE

Permite **agregar condiciones** para **filtrar** los resultados. Obtendremos únicamente los registros que cumplan con esas condiciones.

```
SELECT  
N_cliente as id, nombre  
FROM Clientes  
WHERE N_cliente > 400;
```

id	Nombre
562	Fernando
817	Silvana

# AS

Permite **renombrar** el resultado (de forma temporal) de cualquier objeto dentro de la base de datos: campos, tablas, vistas, entre otros.

# DISTINCT

Permite distinguir registros repetidos

Si incluimos idCliente:

```
SELECT Distinct  
N_Cliente, Nombre, sexo  
FROM Clientes;
```



N. Cliente	Nombre	Sexo
398	Cristian	M
600	Cristian	M
601	Cristian	M
602	Cristian	M
603	Cristian	M

Si sólo tomamos nombre y sexo:

```
SELECT Distinct N  
ombre, Sexo  
FROM Clientes;
```



Nombre	Sexo
Cristian	M

# OPERADORES

# RELACIONALES

Operador	Condición	Ejemplo
=	Verdadero si es igual	Where campo = “valor”
!=	Verdadero si es diferente	Where campo != “valor”
>	Verdadero si es mayor que	Where campo > “valor”
<	Verdadero si es menor que	Where campo < “valor”
>=	Verdadero si es mayor e igual que	Where campo >= “valor”
<=	Verdadero si es menor e igual que	Where campo <= “valor”

# LÓGICOS

Operador	Condición	Ejemplo
AND	Verdadero si se cumplen dos o más condiciones	WHERE condicion1 AND condicion2 AND condicion3...
OR	Verdadero si se cumple alguna de las condiciones	WHERE condicion1 OR condicion2 OR condicion3...
NOT	Verdadero si no se cumple la condición	WHERE NOT condición
LIKE	Verdadero si cumple un patrón	WHERE columna1 LIKE patrón
IN	Verdadero si encuentra alguno de los valores otorgados	WHERE columna1 IN (valor1, valor2, valor3...)
BETWEEN	Verdadero si el valor se encuentra en el rango	WHERE columna1 BETWEEN valor1 AND valor2



# ARITMÉTICOS

Operador	Operación	Ejemplo
+	Permite sumar los valores de dos columnas, una columna y un valor, o dos valores.	Select campo1 + campo2 as campo3 ...
-	Permite restar.	Select campo1 - campo2 as campo3 ...
*	Permite multiplicar.	Select campo1 * campo2 as campo3
/	Permite dividir.	Select campo1 / campo2 as campo3

# CARACTERES

Función	Operación	Ejemplo
LEFT	Extrae caracteres iniciando desde el lado izquierdo	LEFT(Campo1, cantidad de caracteres)
RIGHT	Extrae caracteres iniciando desde el lado derecho	RIGHT(Campo1, cantidad de caracteres)
CONCAT	Permite unir campos o caracteres	CONCAT(Campo1, Campo2, "texto")
REPLACE	Permite reemplazar caracteres en un campo	REPLACE(Campo1, caracter a reemplazar, caracter nuevo)
UPPER/LOWER	Permite convertir todos los valores a mayúsculas o minúsculas, respectivamente	UPPER(Campo1)
TRIM	Elimina espacios al inicio y al final del valor.	Trim(Campo1)

# FECHA

Función	Operación	Ejemplo
YEAR / MONTH / DAY	Devuelve el periodo de tiempo del campo fecha	YEAR(Campo1)
DATEPART / DATENAME	Devuelven el número y el texto respectivamente, del periodo de tiempo correspondiente	DATEPART(Month, Campo1) DATENAME(Month, Campo1) DATEPART(Day, Campo1) DATENAME(Day, Campo1)
DATEADD	Permite agregar la cantidad de periodos que se especifique a una fecha.	DATEADD(periodo, cantidad, Campo1 )
DATEDIFF	Permite calcular la diferencia entre una fecha y otra.	DATEDIFF(periodo, FechaInicial, FechaFinal)

# CONVERSIÓN

Función	Operación	Ejemplo
FLOOR	Devuelve la parte entera del campo decimal	FLOOR(Campo1)
ABS	Devuelve el valor absoluto	ABS(Campo2)

# CAMBIO DE TIPO DE DATO

Función	Operación	Ejemplo
CAST	Convierte el campo de un tipo de datos a otro	CAST(Campo1 AS nuevo tipo de datos)

GROUP BY  
ORDER BY / LIMIT

# ORDER BY

La sentencia *Order By* permite **ordenar el resultado de la consulta** de forma **ascendente o descendente** con respecto a una o varias columnas.

```
SELECT campo1, campo2, ...  
FROM tabla  
ORDER BY campo1, campo2, ... ASC|DESC;
```

```
SELECT id_cliente, nombre  
FROM clientes  
ORDER BY fecha_nacimiento ASC;
```

# LIMIT

La sentencia Limit **delimita la cantidad de filas** devueltas en una consulta

```
SELECT campo  
FROM tabla  
LIMIT número;
```

```
SELECT id_cliente, nombre  
FROM clientes  
LIMIT 3;
```

# GROUP BY

La sentencia Group by sirve para **las funciones de agregación**: permiten efectuar operaciones sobre un conjunto de resultados, pero devolviendo un único valor agregado para todos ellos.

Es decir, nos permiten obtener **medias, máximos, etc...** sobre un conjunto de valores.

N. Cliente	Nombre	Sexo
345	Juan	M
390	Mariela	F
398	Cristian	M
562	Fernando	M
610	Fernando	M
817	Silvana	F



Sexo	Cant_Clientes
F	2
M	4



# GROUP BY

N. Cliente	Nombre	Sexo
345	Juan	M
390	Mariela	F
398	Cristian	M
562	Fernando	M
610	Fernando	M
817	Silvana	F



Sexo	Cant_Clientes
F	2
M	4

```
SELECT Sexo, COUNT(N_Cliente) as Cant_Clientes
FROM CLIENTES
GROUP BY Sexo
```

```
SELECT columna(s), función(columna) FROM tabla GROUP BY
columna(s)
```

***NOTA:** también se puede usar funciones de agregación sin usar la sentencia **group by***

# FUNCIONES DE AGREGACIÓN

Función	Operación	Ejemplo
AVG	Cálculo de Promedio	AVG(Campo) as Promedio
SUM	Cálculo de Sumatoria	SUM(Campo) as Sumatoria
MAX	Obtiene valor máximo	MAX(Campo) as Primero
MIN	Obtiene valor mínimo	MIN(Campo) as Primero
COUNT	Cálculo de cantidad de registros	COUNT(Campo) as Conteo

HAVING

# HAVING

La sentencia Having, como el where, permite establecer **condiciones para filtrar** los resultados **pero** únicamente sobre campos que fueron generados a partir de una **función de agregación**

N. Cliente	Nombre	Sexo
345	Juan	M
390	Mariela	F
398	Cristian	M
562	Fernando	M
610	Fernando	M
817	Silvana	F

Sexo	Cant_Clientes
F	2
M	4

Sexo	Cant_Clientes
M	4

# HAVING

```
SELECT
    Sexo,
    COUNT(N_Cliente) as Cant_Clientes
FROM CLIENTES
GROUP BY Sexo
HAVING COUNT(N_Cliente) > 3
```

N. Cliente	Nombre	Sexo
345	Juan	M
390	Mariela	F
398	Cristian	M
562	Fernando	M
610	Fernando	M
817	Silvana	F

Sexo	Cant_Clientes
F	2
M	4

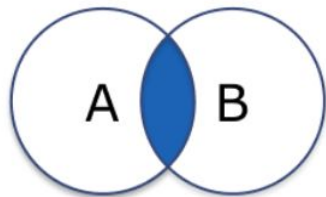
Sexo	Cant_Clientes
M	4

# JOINS

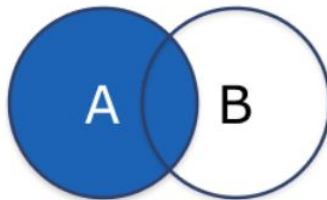
# JOINS

La cláusula **JOIN** permite **combinar registros** de diferentes tablas.

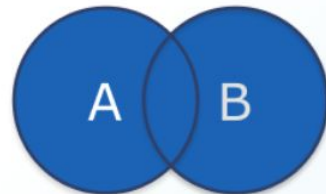
Se complementa con la cláusula **ON** que nos permite establecer la **condición por la cual queremos unir la tabla**, que generalmente son campos que tienen en común las tablas.



Inner Join

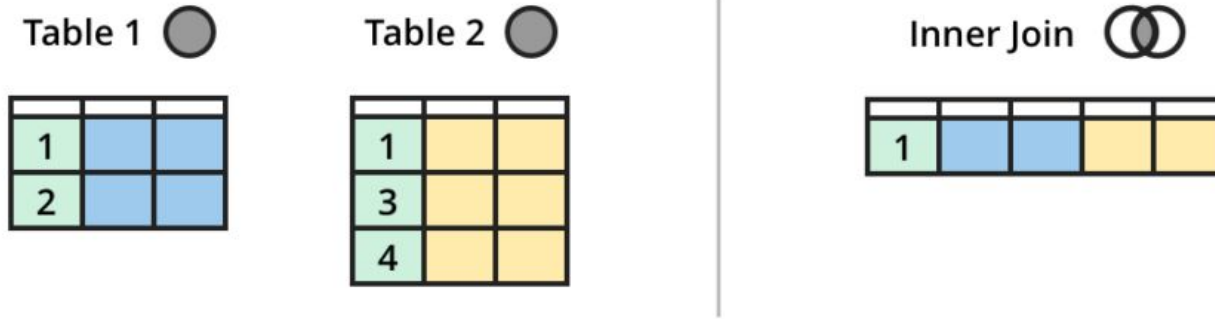


Left Join



Outer Join

# INNER JOIN



El INNER JOIN o simplemente JOIN, retorna **todas las filas de las dos tablas** siempre y cuando haya **coincidencia** por el campo declarado en el ON.



# INNER JOIN

## SINTAXIS

```
SELECT nombres_columnas  
FROM tabla1  
INNER JOIN tabla2 ON  
tabla1.columna_relacion=tabla2.columna_relacion;
```

## EJEMPLO

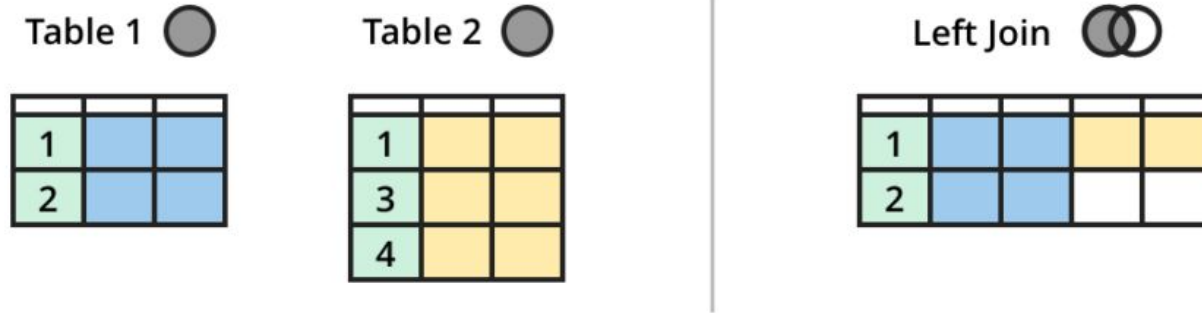
```
SELECT Personas.Apellido, Logros.Logro  
FROM Personas  
INNER JOIN Logros ON Personas.PersonaID =  
Logros.PersonaID
```

PersonaID	Apellido
1	Fleming
2	Eratóstenes
3	Newton
4	Fernandez

Apellido	Logro
Fleming	Descubrimiento de la Penicilina
Eratóstenes	Cálculo del perímetro terrestre
Newton	Ley de gravitación universal
Newton	Desarrollo del Cálculo

LogroID	PersonaID	Logro
1	1	Descubrimiento de la Penicilina
2	2	Cálculo del perímetro terrestre
3	3	Ley de gravitación universal
4	3	Desarrollo del Cálculo
5		Cura del cancer

# LEFT JOIN



El LEFT JOIN retorna **todas las filas de la tabla izquierda** (tabla1), con las filas **coincidentes** en la tabla derecha (tabla2).

El resultado es **NULL** en el lado derecho cuando no hay coincidencia.

# LEFT JOIN

## SINTAXIS

```
SELECT nombres_columnas
FROM tabla1
LEFT JOIN tabla2
ON tabla1.columna_relacion=tabla2.columna_relacion
```

## EJEMPLO

```
SELECT Personas.Apellido, Logros.Logro
FROM Personas
LEFT JOIN Logros
ON Personas.PersonaID = Logros.PersonaID;
```

PersonaID	Apellido
1	Fleming
2	Eratóstenes
3	Newton
4	Fernandez


Apellido	Logro
Fleming	Descubrimiento de la Penicilina
Eratóstenes	Cálculo del perímetro terrestre
Newton	Ley de gravitación universal
Newton	Desarrollo del Cálculo
Fernandez	

LogroID	PersonaID	Logro
1	1	Descubrimiento de la Penicilina
2	2	Cálculo del perímetro terrestre
3	3	Ley de gravitación universal
4	3	Desarrollo del Cálculo
5		Cura del cancer

# OUTER JOIN

Table 1 

1		
2		

Table 2 

1		
3		
4		

Outer Join 

1				
2				
3				
4				

El OUTER JOIN retorna **todas las filas de la tabla de la izquierda** (tabla1) y de la tabla de la derecha (tabla2).

En este caso podemos tener valores **NULL** de ambos lados

# OUTER JOIN

## SINTAXIS

```
SELECT nombres_columnas
FROM tabla1
FULL OUTER JOIN tabla2
ON tabla1.columna_relacion=tabla2.columna_relacion;
```

## EJEMPLO

```
SELECT Personas.Apellido, Logros.Logro
FROM Personas
FULL OUTER JOIN Logros ON Personas.PersonaID =
Logros.PersonaID;
```

PersonaID	Apellido
1	Fleming
2	Eratóstenes
3	Newton
4	Fernandez

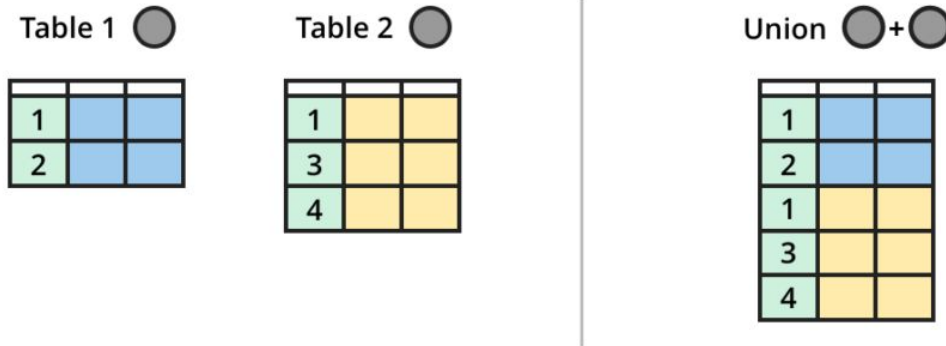
Apellido	Logro
Fleming	Descubrimiento de la Penicilina
Eratóstenes	Cálculo del perímetro terrestre
Newton	Ley de gravitación universal
Newton	Desarrollo del Cálculo
Fernandez	
	Cura del cancer

LogroID	PersonaID	Logro
1	1	Descubrimiento de la Penicilina
2	2	Cálculo del perímetro terrestre
3	3	Ley de gravitación universal
4	3	Desarrollo del Cálculo
5		Cura del cancer

UNION

# UNION

UNION permite **combinar el resultado de varias consultas**: todas las consultas se ejecutan por separado y luego se **concatenan** los resultados.



A diferencia del JOIN, **no se aplica ningún tipo de lógica relacional** para unir registros. Sin embargo la cantidad de columnas en cada consulta debe ser la misma, y las columnas de cada tabla deben tener el mismo tipo de datos.

# UNION

## SINTAXIS

```
SELECT Columna1,Columna2, ...  
FROM NombreDeLaTabla1  
UNION  
SELECT Columna1,Columna2, ...  
FROM NombreDeLaTabla1
```

## EJEMPLO

```
Select LogroID,PersonaID, Logro  
FROM Logros1  
UNION  
Select LogroID, PersonaID, Logro  
From Logros2
```

IdLogro	PersonalID	Logro
1	1	Descubrimiento de la penicilina
2	2	Cálculo del perimetro terrestre
3	3	Ley de gravitación universal
4	4	Desarrollo del cálculo
5		Cura del cancer



IdLogro	PersonalID	Logro
1	1	Descubrimiento de la penicilina
2	2	Cálculo del perimetro terrestre
3	3	Ley de gravitación universal
4	4	Desarrollo del cálculo
5		Cura del cancer
6	10	Descubrimiento de la gravedad
7	11	Primer nobel de quimica
8	11	Creación del datawarehouse



IdLogro	PersonalID	Logro
6	10	Descubrimiento de la gravedad
7	11	Primer nobel de quimica
8	11	Creación del datawarehouse



# IMPORTANTE

Las sentencias tienen un **orden** para su correcto funcionamiento.

## REQUERIDO

- 1.**SELECT**
- 2.**FROM**

## OPCIONAL

- 1.**JOIN**
- 2.**WHERE**
- 3.**GROUP BY**
- 4.**HAVING**
- 5.**ORDER BY**