

Bases de Datos

- Estructura sentencia SQL
- Operadores relacionales
- Operadores lógicos
- Funciones grupo
- Ordenamiento
- Patrones, operador like

La Sentencia SELECT

La sentencia SELECT se utiliza para obtener un conjunto de registros que puede proceder de una o más tablas. El siguiente ejemplo ilustra la sintaxis más elemental:

SELECT

En primer lugar, aparece la palabra **'SELECT'** seguida de los nombres, separados por comas, de los campos que se quieren extraer. La palabra reservada **'FROM'**, que sucede a los nombres de los campos, se utiliza para indicar la tabla o tablas de donde va a extraerse la información.

Ejemplo

```
SELECT id, nombre, apellido  
FROM clientes
```

| ID | Nombre | Apellido |
|----|-----------|----------|
| 1 | Manuel | Longo |
| 2 | Macela | Ciccone |
| 3 | Susana | Valerio |
| 4 | Marcelo | Suarez |
| 5 | Vierginia | Páez |

La Cláusula WHERE

WHERE



Con frecuencia suele ser necesario aplicar algún tipo de condición que restrinja el número de registros devuelto por una consulta.



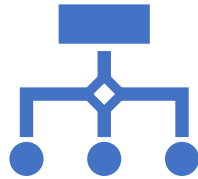
```
SELECT id, nombre, apellido  
FROM clientes  
WHERE id=1
```



La sentencia anterior sirve para obtener el nombre y los apellidos de aquellos clientes cuyo identificador único –el campo ID de la tabla– es igual a 1.

| ID | Nombre | Apellido |
|----|--------|----------|
| 1 | Manuel | Longo |

Estructura



Estructura completa de SQL

SELECT COL1, COL2,...COLN. **FUNCIONES DE GRUPO**,
EXPRESIONES MAT o CADENAS

FROM TABLA/S, VISTAS

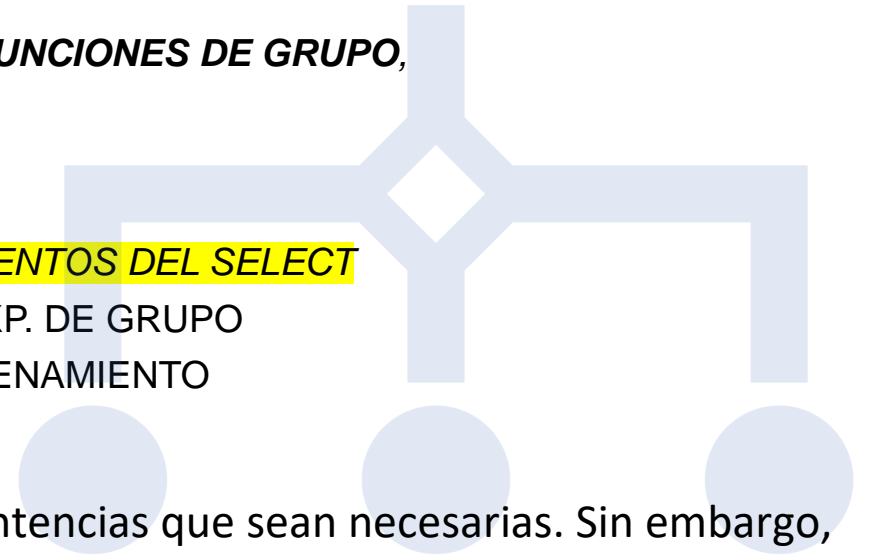
WHERE CONDICIONES

GROUP BY AGRUPA POR ELEMENTOS DEL SELECT

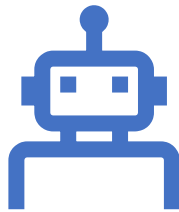
HAVING CONDICIONES DE EXP. DE GRUPO

ORDER BY CRITERIOS DE ORDENAMIENTO

Se usará la combinación de sentencias que sean necesarias. Sin embargo, deben seguir el orden mostrado



OPERAD ORES

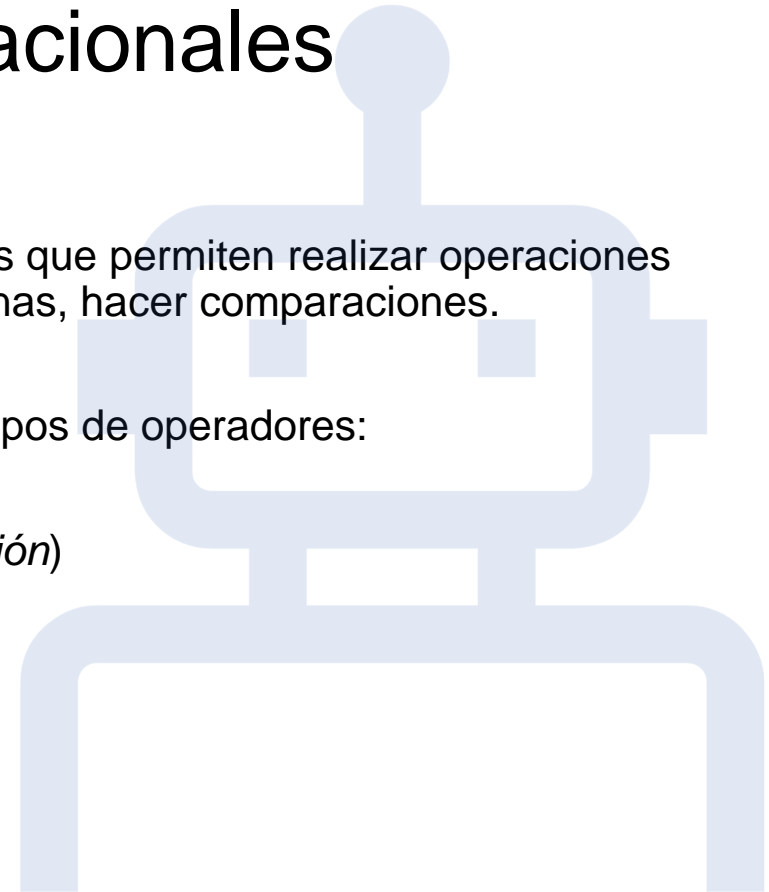


Operadores relacionales

- Los operadores son símbolos que permiten realizar operaciones matemáticas, concatenar cadenas, hacer comparaciones.

Oracle/MySQL reconoce de 4 tipos de operadores:

- 1) relacionales (*o de comparación*)
- 2) aritméticos
- 3) de concatenación
- 4) lógicos



OPERADORES

Operadores relacionales

Los operadores relacionales (o de comparación) nos permiten comparar dos expresiones, que pueden ser variables, valores de campos, etc.

Hemos aprendido a especificar condiciones de igualdad para seleccionar registros de una tabla; por ejemplo

```
SELECT id, nombre, apellido  
FROM clientes  
WHERE id=1
```

Utilizamos el operador relacional de igualdad.
Los operadores relacionales vinculan un campo con un valor para que Oracle compare cada registro (el campo especificado) con el valor dado.

Los operadores relacionales son los siguientes:

| | |
|----|---------------|
| = | igual |
| <> | distinto |
| > | mayor |
| < | menor |
| >= | mayor o igual |
| <= | menor o igual |

OPERADORES

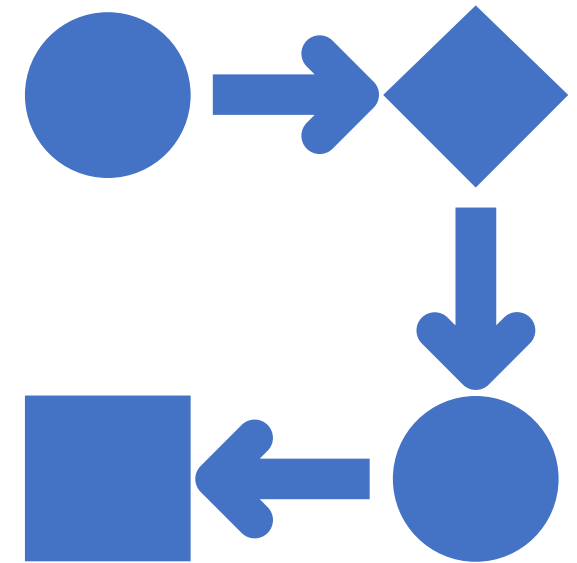
Operadores lógicos

Hasta el momento, hemos aprendido a establecer una condición con "where" utilizando operadores relacionales. Podemos establecer más de una condición con la cláusula "where", para ello aprenderemos los operadores lógicos.

Son los siguientes:

Las operaciones lógicas, son componentes fundamentales de la programación y la ciberseguridad. Estas operaciones, que incluyen **Y (AND)**, **O (OR)** y **NO (NOT)**, permiten la toma de decisiones basadas en múltiples condiciones.

- and, significa "y",
- or, significa "y/o",
- not, significa "no", invierte el resultado
- (), paréntesis



Operadores AND, OR y NOT

OPERADORES

Los registros recuperados en una sentencia que une dos condiciones con el operador "**and**", cumplen con las 2 condiciones.

Los registros recuperados con una sentencia que une dos condiciones con el operador "**or**", cumplen una de las condiciones o ambas.

El operador "**not**" invierte el resultado de la condición a la cual antecede.

Los registros recuperados en una sentencia en la cual aparece el operador "**not**", no cumplen con la condición a la cual afecta el "**NOT**".

Los paréntesis se usan para encerrar condiciones, para que se evalúen como una sola expresión.

LOGICA

Tablas de verdad

Operador AND

| p | q | R |
|-----------|-----------|-----------|
| VERDADERO | VERDADERO | VERDADERO |
| VERDADERO | FALSO | FALSO |
| FALSO | VERDADERO | FALSO |
| FALSO | FALSO | FALSO |

Operador OR

| p | q | R |
|-----------|-----------|-----------|
| VERDADERO | VERDADERO | VERDADERO |
| VERDADERO | FALSO | VERDADERO |
| FALSO | VERDADERO | VERDADERO |
| FALSO | FALSO | FALSO |

Operador NOT

| p | q | R | NOT |
|-----------|-----------|-----------|-----------|
| VERDADERO | VERDADERO | VERDADERO | FALSO |
| VERDADERO | FALSO | FALSO | VERDADERO |
| FALSO | VERDADERO | FALSO | VERDADERO |
| FALSO | FALSO | FALSO | VERDADERO |



Uso de paréntesis

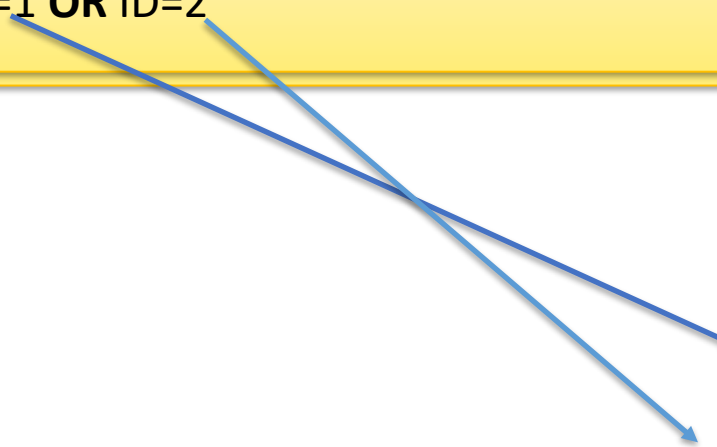
- Si bien los paréntesis no son obligatorios en todos los casos, se recomienda utilizarlos para evitar confusiones.
- El orden de prioridad de los operadores lógicos es el siguiente: "not" se aplica antes que "and" y "and" antes que "or", si no se especifica un orden de evaluación mediante el uso de paréntesis. El orden en el que se evalúan los operadores con igual nivel de precedencia es indefinido, por ello se recomienda usar los paréntesis.
- Entonces, para establecer más de una condición en un "where" es necesario emplear operadores lógicos. "and" significa "y", indica que se cumplan ambas condiciones; "or" significa "y/o", indica que se cumpla una u otra condición (o ambas); "not" significa "no.", indica que no se cumpla la condición especificada.

LOGICA

Operador lógico OR

Cuando existen varias condiciones, éstas pueden agruparse utilizando operadores OR:

```
SELECT ID, nombre, apellido  
FROM clientes  
WHERE ID=1 OR ID=2
```



| ID | Nombre | Apellido |
|----|--------|----------|
| 1 | Manuel | Longo |
| 2 | Macela | Ciccone |

Operador lógico AND

Cuando existen varias condiciones, éstas pueden agruparse utilizando operadores AND:

```
SELECT *  
FROM clientes  
WHERE ID=1 and CP=5501
```

El selector * (asterisco) le indica a Select que debe obtener todas las columnas de la tabla

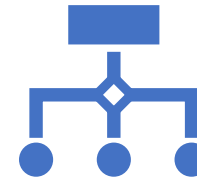
| ID | Nombre | Apellido | Localidad | Domicilio | CP |
|----|--------|----------|------------|--------------------|------|
| 1 | Manuel | Longo | Godoy Cruz | Av. San Martín 322 | 5501 |

LOGICA

Función de Grupo

Funciones de grupo (group by)

SELECT *FUNCIONES DE GRUPO*
FROM *TABLA/S, VISTAS*
WHERE *CONDICIONES*
GROUP BY *AGRUPA POR ELEMENTOS DEL SELECT*
HAVING *CONDICIONES DE EXP. DE GRUPO*



SUM() Realiza una suma sobre una columna de datos numéricos

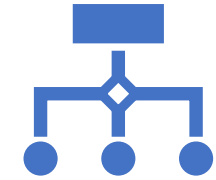
AVG() Obtiene un promedio sobre una columna de datos numéricos

MAX() Obtiene el máximo valor sobre una columna de datos numéricos

MIN() Obtiene el mínimo valor sobre una columna de datos numéricos

COUNT() Cuenta cantidad de registros que contiene una tabla. Hay variantes donde podemos contar registros que cumplen una condición, en este caso las columnas sobre las que se ejecuta la cuenta no consideran los valores NULL

Funciones de grupo (group by)



Función de Grupo

```
SELECT sum(credito)
FROM cliente
```

```
SELECT avg(credito)
FROM cliente
```

```
SELECT max(credito)
FROM cliente
```

```
SELECT min(credito)
FROM cliente
```

```
SELECT count(credito)
FROM cliente
```

| id | nombre | apellido | credito |
|----|----------|----------|---------------|
| 1 | Manuel | Longo | \$ 50.000,00 |
| 2 | Macela | Ciccone | \$ 75.000,00 |
| 3 | Susana | Valerio | \$ 80.000,00 |
| 4 | Marcelo | Suarez | \$ 80.000,00 |
| 5 | Virginia | Páez | \$ 65.000,00 |
| | | suma | \$ 350.000,00 |
| | | promedio | \$ 70.000,00 |
| | | max | \$ 350.000,00 |
| | | min | \$ 50.000,00 |
| | | contar | 5 |





Funciones de grupo (group by)

Función de Grupo

La flexibilidad de SQL me permite construir una sentencia como esta

```
SELECT sum(credito), avg(credito), max(credito), min(credito), count(credito)  
FROM cliente
```

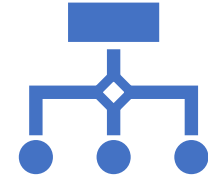
SQL puede combinar funciones de grupo con condiciones en el WHERE

```
SELECT sum(credito), avg(credito), max(credito), min(credito), count(credito)  
FROM cliente
```

```
WHERE CP > 5515
```


Función de Grupo

Funciones de grupo (group by)



Funciones de grupo combinada con un atributo o campo independiente.

En el ejemplo deseamos obtener el total de los créditos de los clientes agrupados por código postal.

El resultado está sujeto a la participación de la sentencia GROUP BY la cual es necesaria para que el resultado sea correcto, es decir que agrupe las sumas de créditos por código postal.

```
SELECT  CP, SUM(credito)
FROM    cliente
GROUP BY CP
```

SQL puede combinar funciones de grupo con condiciones en el WHERE

```
SELECT  CP, SUM(credito)
FROM    cliente
WHERE   CP > 5515
GROUP BY CP
```

ORDER BY

Obtener resultados ordenados. Cláusula ORDER BY

Podemos ordenar el resultado de un **"select"** para que los registros se muestren ordenados por algún campo, para ello usamos la cláusula **"order by"**.

La sintaxis básica es la siguiente:

```
select * from NOMBRETABLA  
order by CAMPO;
```

Por ejemplo, recuperamos todos los registros de la tabla "clientes" ordenados por el **apellido**:

```
select * from clientes  
order by apellido;
```

NOTA: para ordenar en forma descendente uso el modificador **DESC**

Ejemplo:

```
select * from clientes  
order by apellido desc;
```

Tabla Clientes

| ID | Nombre | Apellido | Localidad | Domicilio | CP |
|----|----------|----------|---------------|---------------------|------|
| 2 | Macela | Ciccone | Codoy Cruz | Rafaél Obligado 600 | 5501 |
| 1 | Manuel | Longo | Godoy Cruz | Av. San Martín 322 | 5501 |
| 5 | Virginia | Páez | Luján de Cuyo | Av. San Martín 4500 | 5507 |
| 4 | Marcelo | Suarez | Lavalle | Belgrano 33 | 5560 |
| 3 | Susana | Valerio | Guaymallén | Alem 1639 | 5519 |

ORDER BY

1. Aparecen los registros ordenados alfabéticamente por el campo especificado.
2. También podemos colocar el número de orden del campo por el que queremos que se ordene en lugar de su nombre, es decir, referenciar a los campos por su posición en la lista de selección. Por ejemplo, queremos el resultado del "select" ordenado por "precio":

select apellido, nombre, localidad
from clientes order by 3;

- Si colocamos un número mayor a la cantidad de campos de la lista de selección, aparece un mensaje de error y la sentencia no se ejecuta.
- Por defecto, si no aclaramos en la sentencia, los ordena de manera ascendente (de menor a mayor). Podemos ordenarlos de mayor a menor, para ello agregamos la palabra clave "desc":



| Apellido | Nombre | Localidad |
|----------|----------|---------------|
| Ciccone | Macela | Godoy Cruz |
| Longo | Manuel | Godoy Cruz |
| Valerio | Susana | Guaymallén |
| Suarez | Marcelo | Lavalle |
| Páez | Virginia | Luján de Cuyo |

PATRONES

Búsqueda de patrones (like - not like)

Existe un operador relacional que se usa para realizar comparaciones **exclusivamente de cadenas**, "like" y "not like".

Hemos realizado consultas utilizando operadores relacionales para comparar cadenas. Por ejemplo, sabemos recuperar los clientes cuyo apellido sea igual a la cadena "Longo":

```
select * from clientes  
where apellido = 'Longo';
```

El operador igual ("=") nos permite comparar cadenas de caracteres, pero al realizar la comparación, busca coincidencias de cadenas completas, realiza una búsqueda exacta.

| id | nombre | apellido | credito |
|----|----------|----------|--------------|
| 1 | Manuel | Longo | \$ 50.000,00 |
| 2 | Macela | Ciccone | \$ 75.000,00 |
| 3 | Susana | Valerio | \$ 80.000,00 |
| 4 | Marcelo | Suarez | \$ 80.000,00 |
| 5 | Virginia | Paez | \$ 65.000,00 |
| 6 | Vanina | Suarez | \$ 55.000,00 |
| 7 | Roberto | Sanchez | \$ 70.000,00 |
| 8 | Ramiro | Longo A | |

PATRONES

Búsqueda de patrones (like - not like)

Imaginemos que tenemos registrados estos 2 clientes:

"Manuel", "Longo";
"Ramiro", "Longo A";

Si queremos recuperar todos los clientes de apellido **"Longo"** y especificamos la siguiente condición:

```
select * from clientes  
where apellido='Longo';
```

sólo aparecerá el primer registro, ya que la cadena "Longo" no es igual a la cadena **"Longo A"**.

| id | nombre | apellido | credito |
|----|----------|--------------|--------------|
| 1 | Manuel | <u>Longo</u> | \$ 50.000,00 |
| 2 | Macela | Ciccone | \$ 75.000,00 |
| 3 | Susana | Valerio | \$ 80.000,00 |
| 4 | Marcelo | Suarez | \$ 80.000,00 |
| 5 | Virginia | Paez | \$ 65.000,00 |
| 6 | Vanina | Suarez | \$ 55.000,00 |
| 7 | Roberto | Sanchez | \$ 70.000,00 |
| 8 | Ramiro | Longo A | |

Esto sucede porque el operador "=" (igual), también el operador "<>" (distinto) comparan cadenas de caracteres completas. Para comparar porciones de cadenas utilizamos los operadores **"like"** y **"not like"**.

Búsqueda de patrones (like - not like)

PATRONES

Entonces, podemos comparar trozos de cadenas de caracteres para realizar consultas. Para recuperar todos los registros cuyo apellido contenga la cadena "Longo" debemos tipear:

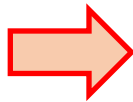
```
select * from clientes  
where apellido like "%Longo%";
```

El símbolo "%" (porcentaje) reemplaza cualquier cantidad de caracteres (*incluyendo ningún carácter*). Es un carácter comodín. "like" y "not like" son operadores de comparación que señalan igualdad o diferencia.

Para seleccionar todos los clientes que comiencen con "S":

```
select * from libros  
where apellido like 'S%';
```

| id | nombre | apellido | credito |
|----|----------|----------------|--------------|
| 1 | Manuel | <u>Longo</u> | \$ 50.000,00 |
| 2 | Macela | Ciccone | \$ 75.000,00 |
| 3 | Susana | Valerio | \$ 80.000,00 |
| 4 | Marcelo | Suarez | \$ 80.000,00 |
| 5 | Virginia | Paez | \$ 65.000,00 |
| 6 | Vanina | Suarez | \$ 55.000,00 |
| 7 | Roberto | Sanchez | \$ 70.000,00 |
| 8 | Ramiro | <u>Longo A</u> | |



| id | nombre | apellido | credito |
|----|---------|----------|--------------|
| 4 | Marcelo | Suarez | \$ 80.000,00 |
| 6 | Vanina | Suarez | \$ 55.000,00 |
| 7 | Roberto | Sanchez | \$ 70.000,00 |

PATRONES

Búsqueda de patrones (like - not like)

Así como "%" reemplaza cualquier cantidad de caracteres, el guión bajo "_" reemplaza un caracter, es otro caracter comodín.

Por ejemplo, queremos ver los clientes que comiencen con "Marcel" pero no recordamos si se escribe "Marcela" o "Marcelo", entonces tipeamos esta condición:

"like" se emplea con tipos de datos caracter y date. Si empleamos "like" con tipos de datos que no son caracteres, Oracle/MySQL convierte (*si es posible*) el tipo de dato a caracter.

Por ejemplo, queremos buscar todos los clientes cuyo credito se encuentre entre 30.000 y 35.000:

```
select * from clientes  
where nombre like "Marcel_";
```

```
select apellido, nombre, credito  
from clientes  
where credito like '1_ _ _ ,%';
```



Gracias...