Argentina programa 4.0

Ministerio de Economía
Argentina

Secretaría de
Economía del Conocimiento

primero la gente

UNTREF
UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

Clase 20: Bases de datos Relacionales

Filtros y Operadores SQL

Ministerio de
Desarrollo Productivo
Argentina

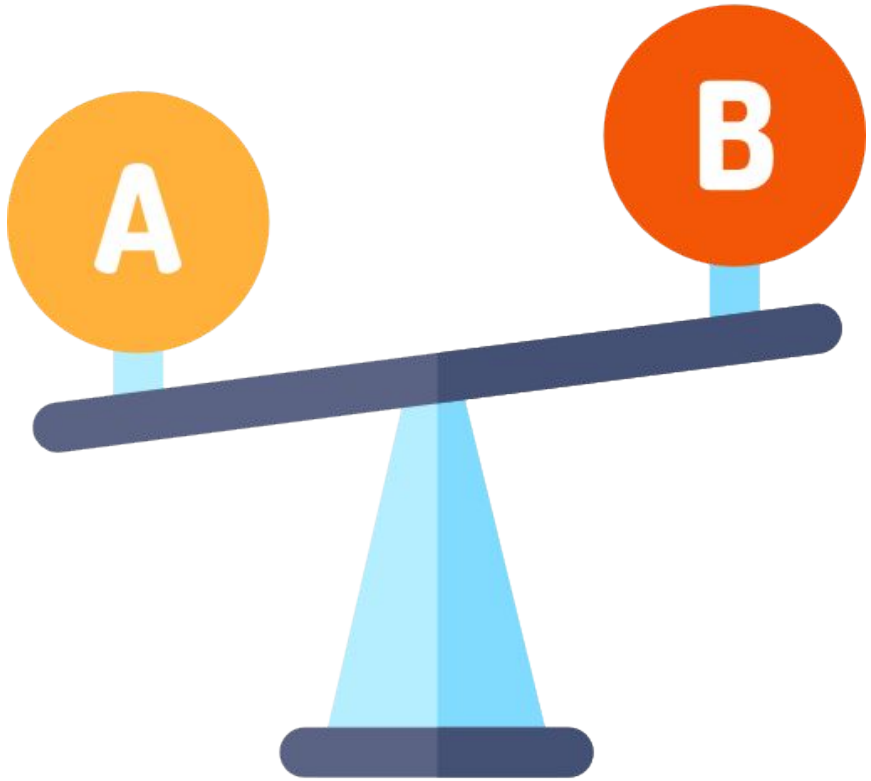
Agenda de hoy

A. Los operadores de comparación
B. Operadores lógicos
C. La cláusula WHERE
    a. Filtrar información
        i. filtro simple
        ii. operador lógico AND
        iii. operador lógico OR
        iv. otros operadores
D. El operador LIKE
    a. Los caracteres comodín

Los operadores de comparación

Los **operadores de comparación** permiten evaluar una condición y determinar si el resultado de dicha condición es **verdadero** o **falso**.

Como ya podemos imaginar, estos operadores son comunes no solo a SQL, sino también a todos los lenguajes de programación más allá de alguna pequeña diferencia en la forma de escribirlos.



Los operadores de comparación

Los **operadores de comparación**:

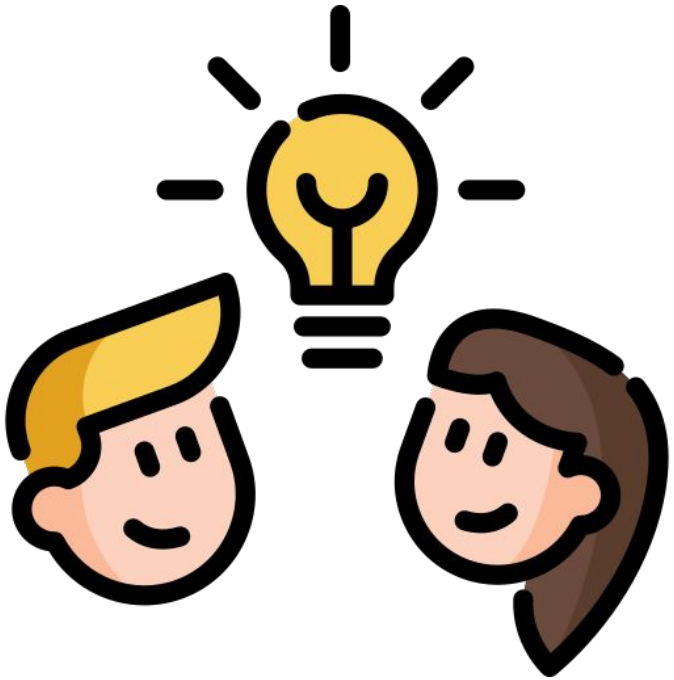
`=`	*igual a*	`IS [NOT] NULL`	*no es nulo*	`BETWEEN`	*entre*
`<`	*menor a*	`NOT`	*NOT lógico*	`[NOT] BETWEEN`	*no esta entre*
`>`	*mayor a*	`LIKE`	*es como*	`IN`	*en (lista)*
`<=`	*menor o igual a*	`[NOT] LIKE`	*no es como*	`[NOT] IN`	*no esta en (lista)*
`=>`	*mayor o igual a*	`IS [NOT] TRUE`	*no es verdadero*	`IS [NOT] FALSE`	*no es falso*
`!= ó <>`	*distinto de*	`AND`	*AND lógico*	`OR`	*OR lógico*

¿Los tienen identificados de algún otro lado? 😋

Los operadores de comparación

Como apreciamos en el slide anterior, esta lista de operadores es CROSS a cualquier lenguaje de programación, más allá de algunas pequeñas diferencias que puedan surgir entre formas de comparar.

El operador “distinto de”, originalmente se escribía de forma diferente entre los sabores más conocidos de SQL. En estos últimos años, se hicieron compatibles ambas opciones entre los principales motores SQL del mercado.



Los operadores lógicos

De esta lista, los operadores lógicos, identificados con el recuadro de color, son los que nos permiten, o combinar más de una posible opción en la cláusula de consulta, o nos permiten invertir el operador de comparación buscando, por ejemplo, aquellos que NO CUMPLEN una determinada condición.

=	*igual a*	IS [NOT] NULL	*no es nulo*	BETWEEN	*entre*
<	*menor a*	NOT	*NOT lógico*	[NOT] BETWEEN	*no esta entre*
>	*mayor a*	LIKE	*es como*	IN	*en (lista)*
<=	*menor o igual a*	[NOT] LIKE	*no es como*	[NOT] IN	*no esta en (lista)*
=>	*mayor o igual a*	IS [NOT] TRUE	*no es verdadero*	IS [NOT] FALSE	*no es falso*
!= ó <>	*distinto de*	AND	*AND lógico*	OR	*OR lógico*

La cláusula WHERE

La cláusula WHERE

Si bien en nuestro encuentro anterior, comenzamos a aplicar filtros de acuerdo a diferentes condiciones, hoy sumamos una de las cláusulas que mejor aplica la lógica condicional sobre un conjunto de registros resultantes.

Veamos de qué se trata y qué rol cumple dentro de las consultas de selección SQL.

La cláusula WHERE

Para poder obtener registros que cumplan una condición específica, debemos aplicar en la sentencia SQL, la cláusula **WHERE**.

Esta permite establecer condiciones para filtrar los resultados.

La cláusula WHERE

El uso de WHERE en cualquier tipo de consulta SQL permite aplicar diferentes filtros sobre la tabla o tablas que le indiquemos, consiguiendo así uno o más campos y especificando sobre éstos el valor a filtrar, integrando en la cláusula a los operadores de comparación.

La cláusula WHERE

El uso de WHERE en cualquier tipo de consulta SQL permite aplicar diferentes filtros sobre la tabla o tablas que le indiquemos, consiguiendo así uno o más campos y especificando sobre éstos el valor a filtrar, integrando en la cláusula a los operadores de comparación.

La cláusula WHERE

La cláusula WHERE

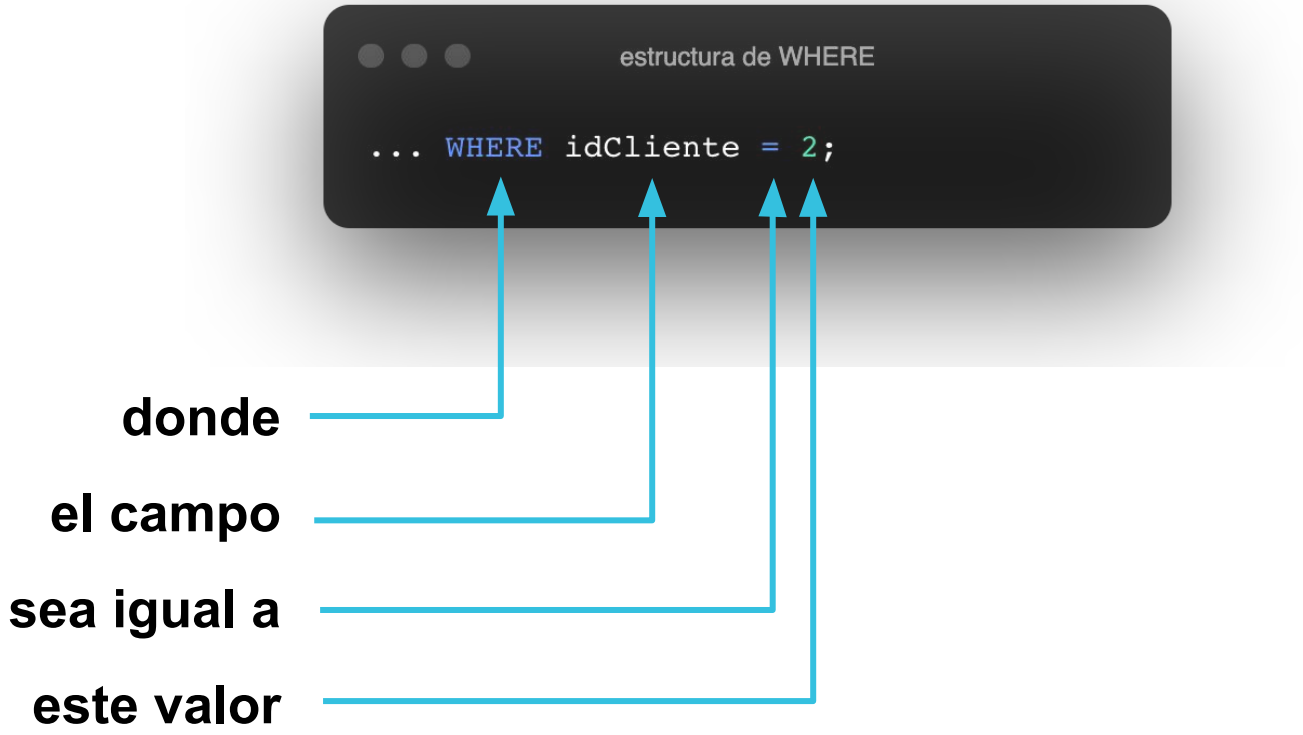
```
SELECT idCliente, nombre FROM clientes WHERE idCliente = 2;
```

idCliente	nombre
1	Marissa Mayer
2	Susan Wojcicki
3	Mira Murati
4	Melinda Gates

idCliente	nombre
2	Susan Wojcicki

Ejemplo de la cláusula **WHERE** aplicada junto al operador igual (=).

La cláusula WHERE



Ejemplo de la cláusula **WHERE** aplicada junto al operador igual (=).

Los operadores de comparación

El operador igual se reemplaza por cualquier otro operador de comparación de acuerdo a la tabla.

Los valores numéricos **difieren de los valores del tipo cadenas de texto.**

Los valores alfanuméricos **deben ser encerrados entre comillas, mientras que en los valores numéricos no es necesario realizar este paso.**

Consultas de filtro simple

WHERE y el operador de comparación =

WHERE numérico

```
SELECT idCliente, nombre FROM clientes WHERE idCliente = 2;
```

WHERE string

```
SELECT idCliente, nombre FROM clientes WHERE nombre = 'Julian';
```

Ejemplo de uso de la cláusula **WHERE** utilizando un campo del tipo numérico, versus la misma cláusula consultando un campo del tipo texto.

WHERE y el operador de comparación =

WHERE booleano

```
SELECT idCliente, nombre FROM clientes WHERE activo = TRUE;
```

Ante el caso de utilizar un campo de tabla booleano dentro de la cláusula WHERE, podremos definir tanto su valor booleano estándar (TRUE ó FALSE), como también su valor numérico (1 ó 0).

En ambos casos, los parámetros o valores los debemos definir sin comillas. SQL se ocupará de traducir el valor booleano al valor numérico, que es lo que este motor entiende.

Otros operadores de comparación

El operador de comparación (mayor a)

El **operador de comparación >** (*mayor a*), se puede aplicar en cualquier campo con cualquier tipo de dato.

Comúnmente sobre tipos de datos numéricos o fechas más que en campos del tipo cadenas de texto.

WHERE operador mayor a

```
… FROM OrderDetails WHERE Quantity > 50;

… FROM Orders WHERE OrderDate > '1997-01-01';
```

El operador de comparación (menor a)

El **operador de comparación <** (*mayor a*), cumple con los mismos requisitos y condiciones que el operador (*mayor a*), pero obviamente a la inversa.

WHERE booleano

```
... FROM OrderDetails WHERE Quantity < 20;

... FROM Orders WHERE OrderDate < '1997-08-30';
```

El operador de comparación (mayor a)

El **operador de comparación >=** (*mayor o igual a*), de igual forma que el **operador mayor a**, con la diferencia de que incluye en el resultado cualquier valor que sea estrictamente igual al parámetro definido, además de todo aquel valor superior.

WHERE operador mayor o igual a

```
… FROM Orders WHERE OrderID >= 11000;
```

El operador de comparación (mayor o igual a)

El **operador de comparación >=** (*mayor o igual a*), de igual forma que el **operador mayor a**, con la diferencia de que incluye en el resultado cualquier valor que sea estrictamente igual al parámetro definido, además de todo aquel valor superior.

WHERE operador mayor o igual a

```
… FROM Orders WHERE OrderID >= 11000;
```

Sección práctica

Vayamos experimentando el uso de los operadores de comparación vistos hasta aquí.

Como ya tenemos práctica en el manejo de filtros de la mano de funciones de orden superior, no tendremos mayor problema en aplicar el manejo de los mismos pero dentro de diferentes cláusulas SQL.

Prácticas

Tiempo estimado: **15 minutos**

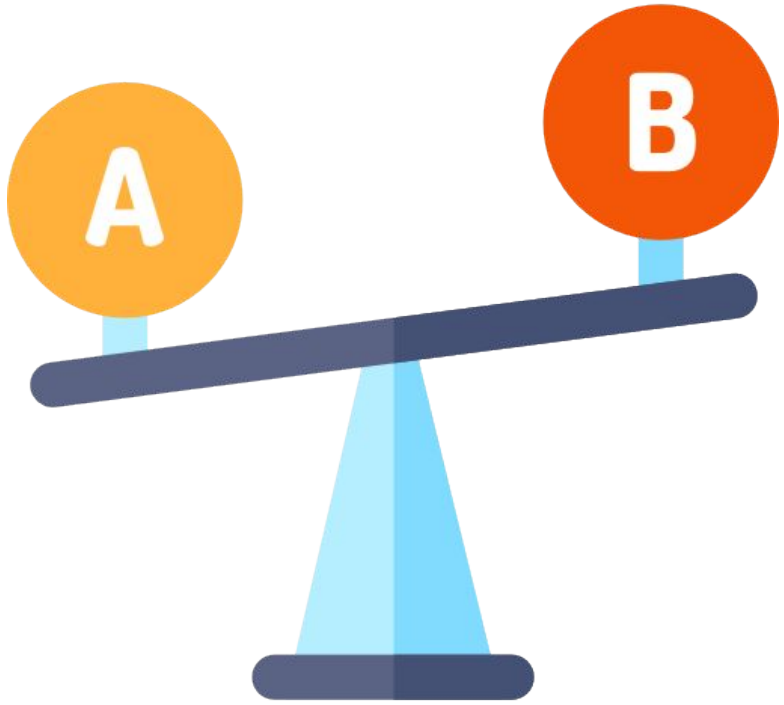
1. **Ejecuta una consulta de selección sobre todos los campos de la tabla Customers**
   - **donde la ciudad sea igual a ‘Buenos Aires’**
2. **Ejecuta una consulta de selección sobre los siguientes campos de la tabla Customers:**
   - **customerID, CompanyName, ContactName, ContactTitle**
   - **donde el campo City sea igual a ‘London’**
3. **Ejecuta una consulta de selección sobre todos los campos de la tabla Employees:**
   - **donde el campo Title sea igual a ‘Sales Representative’**
   - **Order por el campo City de forma descendente**
4. **Ejecuta una consulta de selección sobre los siguientes campos de la tabla Employees:**
   - **LastName, FirstName, Title, City**
   - **donde el campo Country sea igual a ‘USA’**
   - **Ordena esta consulta por el campo LastName**
5. **Ejecuta una consulta de selección sobre los siguientes campos de la tabla Suppliers:**
   - **SupplierID, CompanyName, ContactName**
   - **donde el campo ContactTitle sea igual a ‘Accounting Manager’**

El operador lógico AND

El operador lógico AND

El **operador lógico AND** se utiliza para **combinar múltiples condiciones** en una cláusula WHERE.

Cuando se utiliza este operador, **todas las condiciones deben ser verdaderas** para que la fila o registro se incluya en el resultado de la consulta.



El operador lógico AND

Por ejemplo, ante una consulta de selección de datos de clientes donde queremos buscar un cliente con un determinado código, o Identificador, y que además esté con su estado **Activo**, debemos combinar allí ambas condiciones mediante el operador lógico AND.

WHERE operadores lógicos

```
… FROM clientes WHERE idCliente = 2 AND estado = 'Activo';
```

El operador lógico AND

Solo obtendremos como resultado aquel o aquellos registros que coincidan con ambos parámetros y valores especificados. Si ninguno cumple con los dos parámetros y valores, entonces la consulta no retornará ninguna fila como resultado.

WHERE operadores lógicos

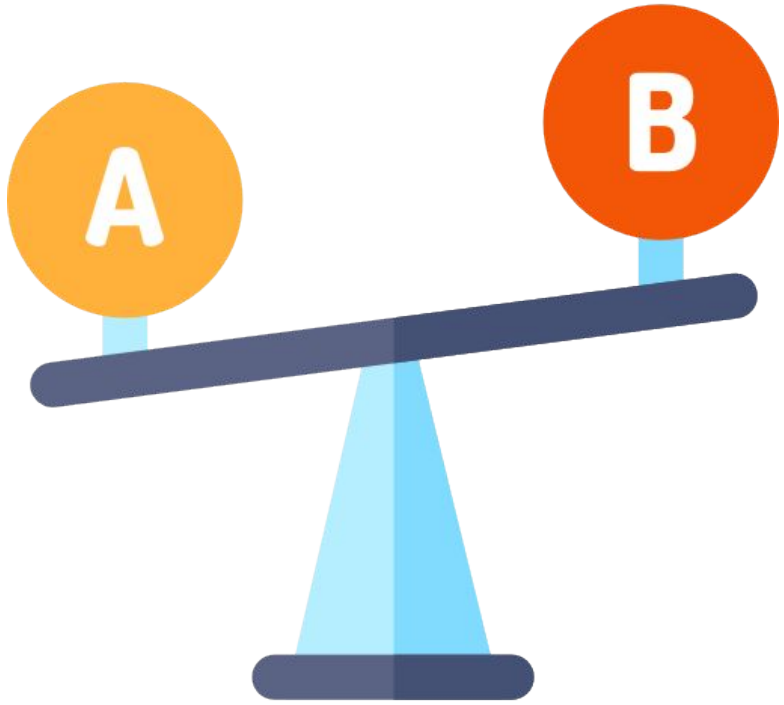
```
… FROM clientes WHERE idCliente = 2 AND estado = 'Activo';
```

El operador lógico OR

El operador lógico OR

El **operador lógico OR** se utiliza para **combinar una condición con un valor específico** más una posible condición como valor de fallback.

Cuando se utiliza este operador, **si la primera condición no encuentra resultados en la tabla** se irá a buscar registros que coincidan con la segunda condición especificada.



El operador lógico OR

En este ejemplo, intentamos visualizar en la consulta resultante solo aquellos productos que tengan las unidades en stock en cero. Si no existe ningún producto con esta condición, entonces se intentará obtener aquellos productos cuyo campo Discontinued es igual a TRUE, o 1, que es el equivalente en valor booleano.

WHERE operadores lógicos

```
SELECT * FROM Products
WHERE UnitsInStock = 0 OR Discontinued = 1;
```

El operador lógico OR

En el caso que no se cumpla una condición o la otra, el resultado de esta consulta arrojará una tabla sin registros.

El operador lógico OR funciona como un operador de cortocircuito (operando 1 ú operando 2), de la misma forma que sucede en JavaScript.

WHERE operadores lógicos

```sql
SELECT * FROM Products
WHERE UnitsInStock = 0 OR Discontinued = 1;
```

Otros operadores de comparación

El operador IN

El operador de comparación IN funciona como la intersección de Conjuntos en Matemáticas.

Debemos definir todos aquellos valores que deben estar incluidos en el campo que utilizamos para aplicar la condición.

WHERE otros operadores

```
SELECT * FROM Products
WHERE categoryID IN (3, 4, 7);
```

El operador IN

Traerá como resultado solo aquellos productos cuyo campo categoryID tenga como valor el número 3, el número 4 o el número 7.

WHERE otros operadores

```
SELECT * FROM Products
WHERE categoryID IN (3, 4, 7);
```

El operador “distinto de”

Este operador se ocupa de recuperar todos aquellos registros cuyo parámetro sea distinto del valor indicado.

WHERE otros operadores

```
SELECT * FROM Products
WHERE categoryID <> 3;
```

El operador “distinto de”

Desde hace algunos años, MySQL soporta como operador “*distinto de*” el uso de los signos **<>** y también de los signos !=. Originalmente la primera de las opciones se utilizaba solamente en SQL Server, mientras que esta última opción se utilizaba dentro de todas las bases SQL que se ajustaban al estándar ANSI SQL.

WHERE otros operadores

```
SELECT * FROM Products
WHERE categoryID != 3;
```

El operador Between

Este operador nos permite especificar un rango de valores para el parámetro por el cual vamos a realizar la consulta de selección.

En este caso, los valores del campo **UnitsInStock** deben encontrarse entre **10** y **40** unidades.

WHERE otros operadores

```
SELECT * FROM Products
WHERE UnitsInStock BETWEEN 10 AND 40;
```

El operador Between

Sería el equivalente a utilizar en algún lenguaje de programación la conjugación de **Mayor o igual** a y **Menor o igual a**, solo que en SQL se utiliza la palabra reservada BETWEEN en conjunción con el operador lógico AND.

WHERE otros operadores

```sql
SELECT * FROM Products
WHERE UnitsInStock BETWEEN 10 AND 40;
```

El operador LIKE

El operador LIKE

El **operador LIKE** se utiliza para buscar patrones en una columna o campo de una tabla.

A diferencia del resto de los operadores vistos hasta aquí, LIKE es un poco más laxo, dado que nos permite buscar un patrón aproximado al valor que le indicamos, y no exacto como es el caso de la mayoría de los otros operadores que trabajamos.

El operador LIKE

operador LIKE

```
SELECT idCliente, nombre
FROM clientes
WHERE nombre LIKE 'Ju%';
```

En este ejemplo, intentamos ubicar a uno o más clientes que posean como valor en su campo nombre, el valor *'Ju'*. Si prestamos atención al parámetro indicado como valor, veremos que este posee además de las letras indicadas, el uso del caracter **%**. Este caracter oficia de comodín, representando al resto de los posibles caracteres que pueda llegar a contener el nombre en cuestión.

El operador LIKE

operador LIKE

```
SELECT idCliente, nombre
FROM clientes
WHERE nombre LIKE 'Ju%';
```

En este ejemplo, intentamos ubicar a uno o más clientes que posean como valor en su campo nombre, el valor *'Ju'*. Si prestamos atención al parámetro indicado como valor, veremos que este posee además de las letras indicadas, el uso del caracter **%**. Este caracter oficia de comodín, representando al resto de los posibles caracteres que pueda llegar a contener el nombre en cuestión.

El operador LIKE

operador LIKE

```
SELECT idCliente, nombre
FROM clientes
WHERE nombre LIKE 'Ju%';
```

De la forma en la cual está planteado este parámetro como valor, podríamos obtener una lista de nombres con valores como los siguientes:

"Julia, Julio, Julián, Juliana, Julieta, Juan", entre otros...

El operador LIKE

operador LIKE

```
SELECT idCliente, nombre
FROM clientes
WHERE nombre LIKE 'Ju%';
```

Aquí vemos la flexibilidad que nos da el uso de LIKE. Nos acerca un valor aproximado al indicado, pudiendo contener cero, uno o más caracteres aparte de los indicados como valor del parámetro.

Los caracteres comodín

Los caracteres comodín

Los caracteres comodín se utilizan junto a las cláusulas WHERE y LIKE, para especificar diferentes formas de obtener información específica.

El caracter **%** nos da una perspectiva indefinida además de los caracteres que le indiquemos junto a este caracter comodín.

Los caracteres comodín

Aquí tenemos un ejemplo amplio de cómo podemos aprovechar el uso del caracter comodín **%**, para poder buscar un término con diferentes posibles combinaciones dentro de una palabra.

WHERE otros operadores

```
...WHERE nombre LIKE 'Ju%';
-- retorna todos los nomrbes que comiencen con 'Ju'

...WHERE nombre LIKE '%ju';
-- retorna todos los nomrbes que finalicen con 'Ju'

...WHERE nombre LIKE '%ju%';
-- retorna todos los nomrbes que comiencen, finalicen,
o contengan en alguna parte del mismo el término 'ju'
```

Los caracteres comodín

Existe una diversidad de caracteres comodines y combinaciones, las cuales nos permiten alcanzar una amplia variedad de resultados, según la complejidad de búsqueda ante la cual nos encontremos.

Caracteres comodín y LIKE

```
SELECT * FROM Tabla WHERE Campo LIKE '_r%';

SELECT * FROM Tabla WHERE Campo LIKE 'Au_____';

SELECT * FROM Tabla WHERE Campo LIKE 'A_%';

SELECT * FROM Tabla WHERE Campo LIKE 'A_____a';
```

Los caracteres comodín

El uso del caracter _ (*underscore*) representa una sola letra en un parámetro.

Caracteres comodín y LIKE

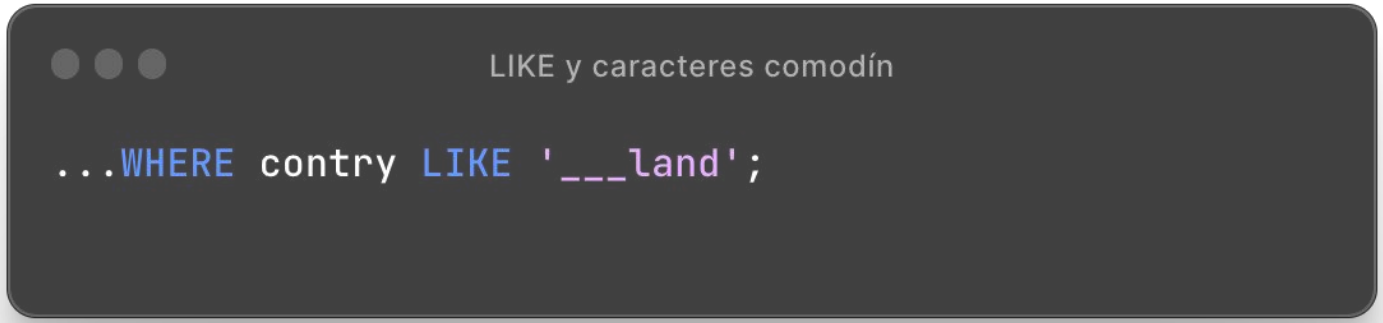
```
SELECT * FROM Tabla WHERE Campo LIKE '_r%';

SELECT * FROM Tabla WHERE Campo LIKE 'Au_____';

SELECT * FROM Tabla WHERE Campo LIKE 'A_%';

SELECT * FROM Tabla WHERE Campo LIKE 'A_____a';
```

Los caracteres comodín



En este ejemplo, si filtramos de una tabla todos los países del mundo, obtendremos como resultado los posibles valores *'Ireland'* y *'Iceland'*, dentro de la consulta resultante.

Sección práctica

Ahora que ampliamos nuestro conocimiento con otros tantos operadores de comparación junto a la cláusula WHERE, pongamos a ejercitar los mismos realizando las siguientes consignas sobre la bb.dd Northwind.

Prácticas

1. Ejecuta una consulta de selección sobre todos los campos de la tabla **Products**
    - ordena la consulta por el campo **productName**
2. Ejecuta una consulta de selección similar a la primera, aplicando la siguiente condición
    - **el campo CategoryID sea igual a 4 y 6**
    - mantén el ordenamiento indicado anteriormente
3. Ejecuta otra consulta de selección similar a la primera, aplicando la siguiente condición
    - el campo **SupplierID sea igual a 5** y el campo **CategoryID sea igual a 4**
4. Ejecuta otra consulta de selección similar a la primera, aplicando la siguiente condición
    - el campo **UnitsInStock** tenga valores **entre 25 y 40** unidades
5. Abre la tabla Products y modifica manualmente el campo **discontinued = 1**, en al menos 5 registros al azar. Recuerda aplicar / guardar los cambios efectuados
6. Ejecuta una consulta de selección similar a la primera, aplicando la siguiente condición
    - el campo **UnitsInStock sea mayor a 400** o el campo **discontinued sea verdadero**
    - ordena la consulta por el campo productName

Muchas gracias.

Ministerio de Economía
Argentina

Secretaría de
Economía del Conocimiento

primero
la gente