



# CURSO SQL DESDE CERO

**Calidad:** "Puedes tener un software de calidad o puedes tener aritmética de punteros, pero no puedes tener ambas cosas al mismo tiempo"

-- Bertrand Meyer --

[Obviar](#)

[Búsqueda avanzada](#)

- [Índice general](#) < [Cursos SQL](#) < [Curso SQL desde cero](#)
- [Cambiar tamaño de la fuente](#)
- [Imprimir vista](#)
- [FAQ](#)
- [Registrarse](#)
- [Identificarse](#)

[Donar](#)

## [Lección 5 - Operadores \(SQL WHERE\)](#)



Un [operador](#), como su propio nombre indica, es quien opera, normalmente entre dos [operandos](#), estableciendo una operación que al ejecutarla se obtiene un resultado.

Por ejemplo en matemáticas:  $3 + 4$

"+" es el operador y, "3" y "4" son los operandos. el resultado de esta operación es 7.

A modo de apunte diremos, aunque poco tenga que ver con esta lección, que el SQL nos permite calcular ciertas operaciones matemáticas tanto en la cláusula SELECT, para obtener resultados, como en la cláusula WHERE, para establecer condiciones:

Código: [Seleccionar todo](#)

```
select 3 + 4
```



[llevar código al banco de pruebas](#)

A la pregunta: ¿Qué empleados tienen un salario mayor de 1350?, podríamos construir la consulta SQL así, sustituyendo 1350 por:  $1300 + 50$ .

Código: [Seleccionar todo](#)

```
select NOMBRE , APELLIDOS  
from EMPLEADOS  
where SALARIO > 1300 + 50
```



[llevar código al banco de pruebas](#)

### Lógica booleana

Lo que se pretende abordar en esta lección principalmente es la [lógica booleana](#), que es la que nos permite establecer condiciones. Advierto que esta lección puede resultar un poco dura, si tiene dificultades para entender lo que trataremos, no se preocupe e intente quedarse con la idea de fondo.

Cuando usted jugaba al ya clásico juego de ¿quién es quién?,



usted estaba aplicando lógica booleana para descubrir el personaje que su contrincante escondía. Usted lanzaba preguntas con el propósito de saber si una característica del personaje misterioso era cierta o falsa, y si era cierta, descartaba los personajes que no presentaban esa característica. El SGBD hace algo muy parecido, usted establece una característica en la cláusula WHERE de la consulta SQL, y el SGBD descarta los registros en que la característica es falsa, mientras que mantiene los que es cierta.

### Expresiones booleanas

Si usted jugado a ¿Quién es quién? preguntaba a su contrincante: ¿tiene barba?, lo que en realidad estaba preguntando era: ¿EL PERSONAJE MISTERIOSO tiene BARBA? pues bien, esto es una expresión booleana donde:

"tiene" es el operador, "EL PERSONAJE MISTERIOSO" es un operando variable, que depende del personaje elegido por su contrincante, y "BARBA" es un operando constante. El resultado de aplicar esta expresión a un personaje concreto dará cierto si el personaje tiene barba, y falso en caso contrario.

Cuando en la primera consulta de este curso:

Código: [Seleccionar todo](#)

```
select NOMBRE , APELLIDOS
  from EMPLEADOS
 where SALARIO > 1350
```

 [llevar código al banco de pruebas](#)

indicábamos en la clausula WHERE: SALARIO > 1350(¿es SALARIO mayor a 1350?), estábamos estableciendo una expresión booleana donde:

">" es el operador, "SALARIO" es un operando variable, que tomará valores de cada registro de la tabla EMPLEADOS, y "1350" es un operando constante. El resultado de esta expresión depende del valor que tome la variable SALARIO, pero en cualquier caso sólo puede dar dos posibles resultados, o cierto o falso. Por lo tanto diremos que una expresión booleana sólo tiene dos posibles resultados.

El motor SQL evalúa la expresión booleana de la clausula WHERE para cada registro de la tabla, y el resultado determina si el registro que se está tratando se tomará en consideración. Lo hará si el resultado de evaluar la expresión es cierto, y lo ignorará si el resultado es falso.

Ejemplo de expresiones booleanas:

- "4 > 3" : ¿es cuatro mayor que tres?
- "3 = 12" : ¿es tres igual a doce?

Fíjese que los operandos son del mismo tipo, en este caso tipo INT. Sin embargo el resultado obtenido no es un dato de tipo INT, sino booleano, sus posibles valores son cierto o falso.

(4 > 3) = cierto

(3 = 12) = falso

## Operadores

Algunos de los operadores que nos permiten construir expresiones booleanas son:

- > : "A > B" devuelve cierto si A es estrictamente **mayor que** B, de lo contrario devuelve falso.
- < : "A < B" devuelve cierto si A es estrictamente **menor que** B, de lo contrario devuelve falso.
- = : "A = B" devuelve cierto si A es **igual a** B, de lo contrario devuelve falso.
- >= : "A >= B" devuelve cierto si A es **mayor o igual a** B, de lo contrario devuelve falso.
- <= : "A <= B" devuelve cierto si A es **menor o igual a** B, de lo contrario devuelve falso.
- != : "A != B" devuelve cierto si A es **distinto a** B, de lo contrario devuelve falso.

Al construir expresiones con estos operadores, los dos operandos deben ser del mismo tipo, ya sean números, cadenas o fechas.

Ejemplo de expresión booleana con cadenas: ('Aranda, Pedro' < 'Zapata, Mario') = cierto, puesto que por orden alfabético 'Aranda, Pedro' está posicionado antes que 'Zapata, Mario', por lo tanto es menor el primero que el segundo.

## Operadores lógicos

Los [operadores lógicos](#) permiten formar expresiones booleanas tomando como operandos otras expresiones booleanas. Fíjese que en las expresiones vistas anteriormente, los operandos debían ser números, cadenas o fechas, ahora sin embargo los operandos deben ser expresiones booleanas, el conjunto forma una nueva

expresión booleana que, como toda expresión booleana, dará como resultado cierto o falso.

- **AND** : "A and B" devuelve cierto si A y B valen cierto, y falso en cualquier otro caso.
- **OR** : "A or B" devuelve cierto si A o B valen cierto, y falso únicamente cuando tanto A como B valen falso.
- **NOT** : "not A" devuelve falso si A vale cierto, y cierto si A vale falso.

Veamos una aplicación en el mundo cotidiano.

Supongamos el siguiente anuncio:

Mi jefe quiere contratar a una persona para repartir genero, solamente pueden optar a la vacante aquellos candidatos que tengan vehículo propio y licencia de conducir automóviles. Como candidatos tenemos a Ángela, que tiene licencia pero no tiene vehículo. A Salva, que tiene licencia y vehículo. Y a Teresa, que tiene vehículo pero de momento no tiene licencia. ¿Quiénes pueden pasar al proceso de selección?

Convertimos el anuncio en una expresión booleana:

Sea C: pasa al proceso de selección.

Sea A: tiene vehículo propio.

Sea B: tiene licencia de conducir automóviles.

Entonces para que un candidato satisfaga C, se debe dar A y B:

**C = A and B**

Resolvamos la expresión para cada candidato:

Aplicado a Ángela: **C** = (A and B) = (falso and cierto) = **falso**. Luego no pasa al proceso de selección.

Aplicado a Salva: **C** = (A and B) = (cierto and cierto) = **cierto**. Luego pasa al proceso de selección.

Aplicado a Teresa: **C** = (A and B) = (cierto and falso) = **falso**. Luego no pasa al proceso de selección.

Veamos ahora esto mismo aplicado al SQL:

Consideremos ahora la tabla PERSONAS, donde hemos guardado una "S" en el campo RUBIA si la persona es rubia y una "N" en caso contrario, análogamente se ha aplicado el mismo criterio para ALTA y GAFAS, es decir, para indicar si es alta y si lleva gafas.

ID_PERSONA	NOMBRE	RUBIA	ALTA	GAFAS
1	Manuel	S	S	N
2	Maria	N	N	S
3	Carmen	S	N	S
4	José	S	S	S
5	Pedro	N	S	N

### El operador AND

Como ya hemos dicho el operador AND devuelve cierto si ambas expresiones son ciertas, y falso en cualquier otro caso. Supongamos que queremos saber ¿qué personas son rubias y altas?, para ello construimos la siguiente consulta SQL:

Código: [Seleccionar todo](#)

```
select NOMBRE
  from PERSONAS
 where (RUBIA = 'S') and (ALTA = 'S')
```

 [llevar código al banco de pruebas](#)

Resultado:

NOMBRE
Manuel
José

Evaluar (RUBIA = 'S') da como resultado cierto o falso, al igual que evaluar (ALTA = 'S'), son de echo los dos operandos booleanos del operador AND. Si para un registro ambos son ciertos el resultado es cierto, y se mostrarán los datos de ese registro que indica la clausula SELECT.

En el caso de tener una expresión de la forma: "A and B and D" la expresión se evalúa por partes por orden de aparición:

primero se evalúa (A and B) = E  
y finalmente (E and D)

Si todos los operadores de la expresión son AND, entonces todas las expresiones deben valer cierto para que el resultado sea cierto.

Veamos un ejemplo de esto mientras jugamos a ¿Quién es quién?

Usted pregunta:

¿es **rubia**? y la respuesta es **cierto**

¿es **alta**? y la respuesta es **falso**

¿lleva **gafas**? y la respuesta es **cierto**

Por lo tanto debe ser una persona que sea rubia y, no sea alta y, lleve gafas.

¿Quién es el personaje?

Código: [Seleccionar todo](#)

```
select NOMBRE
  from PERSONAS
 where (RUBIA = 'S') and (ALTA = 'N') and (GAFAS='S')
```



[llevar código al banco de pruebas](#)

Resultado:

<u>NOMBRE</u>
Carmen

Antes de dejar el operador AND, recordar del modo equivalente y más simplificado que se comentó en la lección 3 en que podemos condicionar un campo a un rango de valores mediante el operador BETWEEN:

Código: [Seleccionar todo](#)

```
where SALARIO >= 1300 and SALARIO <=1500
```

que el salario sea mayor o igual a 1300 y menor o igual a 1500

forma equivalente:

Código: [Seleccionar todo](#)

```
where SALARIO between 1300 and 1500
```

que el salario este entre 1300 y 1500

## El operador OR

Con el operador OR basta que uno de los dos operandos sea cierto para que el resultado sea cierto:

Supongamos que queremos saber las personas que son rubias o bien altas, es decir, queremos que si es rubia la considere con independencia de su altura, y a la inversa, también queremos que la seleccione si es alta independientemente del color de pelo. La consulta sería la siguiente.

Código: [Seleccionar todo](#)

```
select NOMBRE
  from PERSONAS
 where (RUBIA = 'S') or (ALTA = 'S')
```



[llevar código al banco de pruebas](#)

Resultado:

NOMBRE
Manuel
Carmen
José
Pedro

Si todos los operadores de la expresión son OR, por ejemplo "A or B or C", entonces todas las expresiones deben valer falso para que el resultado sea falso, con que solo una valga cierto el resultado es cierto.

Supongamos que quiere seleccionar tres registros concretos de la tabla EMPLEADOS para ver sus datos, le interesan los registros con identificador 1, 2 y 4. Para esta situación debe usar el operador OR, puesto que su consulta debe establecer la condición: que el identificador sea 1, 2 o 4:

Código: [Seleccionar todo](#)

```
select *
  from EMPLEADOS
 where ID_EMPLEADO = 1 or ID_EMPLEADO = 2 or ID_EMPLEADO = 4
```



[llevar código al banco de pruebas](#)



Resultado:

ID_EMPLEADO	NOMBRE	APELLIDOS	F_NACIMIENTO	SEXO	CARGO	SALARIO
1	Carlos	Jiménez Clarín	1985-05-03	H	Mozo	1500
2	Elena	Rubio Cuestas	1978-09-25	M	Secretaria	1300
4	Margarita	Rodríguez Garcés	1992-05-16	M	Secretaria	1325.5

El asterisco presente en la cláusula SELECT equivale a indicar todos los campos de la tabla.

Fijese como en la anterior consulta para cualquier registro de la tabla EMPLEADOS que el campo ID\_EMPLEADO contenga un valor distinto a 1, 2 o 4, el resultado de evaluar la expresión será falso, puesto que todas las expresiones booleanas darán falso, pero con que una de ellas valga cierto, el registro será seleccionado. De hecho si un mismo campo aparece dos o más veces en expresiones de la cláusula WHERE como en el ejemplo anterior, carece de sentido que el operador sea AND: usted puede esperar que una persona sea rubia o morena, pero no puede esperar que sea rubia y morena. Del mismo modo el identificador de un registro nunca podrá ser 1 y 2, o es 1 o es 2 o es x.

Un modo de simplificar la anterior consulta es mediante la palabra clave IN, donde se establece una lista de valores posibles que debe contener el campo indicado para que el registro sea seleccionado. La palabra clave IN equivale a establecer condiciones sobre un mismo campo conectadas por el operador OR.

Código: [Seleccionar todo](#)

```
select *
  from EMPLEADOS
 where ID_EMPLEADO in (1,2,4)
```

 [llevar código al banco de pruebas](#)

Resultado:

ID_EMPLEADO	NOMBRE	APELLIDOS	F_NACIMIENTO	SEXO	CARGO	SALARIO
1	Carlos	Jiménez Clarín	1985-05-03	H	Mozo	1500
2	Elena	Rubio Cuestas	1978-09-25	M	Secretaria	1300
4	Margarita	Rodríguez Garcés	1992-05-16	M	Secretaria	1325.5

## El operador NOT

Este operador tan solo tiene un operando, el resultado es negar el valor del operando de modo que:

"(4 > 3) = cierto" luego "not (4>3) = falso"

Si negamos dos veces una expresión booleana es equivalente a la expresión original:

"(4 > 3) = cierto" luego "not ( not (4>3) ) = cierto"

Cuando descubrimos al personaje misterioso, a la pregunta ¿es alta? la respuesta era falso, luego podríamos haber establecido lo siguiente: "not (ALTA = 'S')" en lugar de "(ALTA = 'N')":

Código: [Seleccionar todo](#)

```
select NOMBRE
  from PERSONAS
 where (RUBIA = 'S') and not (ALTA = 'S') and (GAFAS='S')
```

 [llevar código al banco de pruebas](#)

Resultado:

<b>NOMBRE</b>
Carmen

Otro ejemplo: si negamos toda la expresión de la clausula WHERE, estaremos precisamente seleccionando los registros que antes descartábamos, y al revés, descartando los que antes seleccionábamos.

Tomemos la anterior consulta y neguemos la clausula WHERE, si antes el resultado era: Carmen, ahora el resultado ha de ser todas las persona menos Carmen. Para hacer esto cerramos entre paréntesis toda la expresión y le colocamos el operador "not" delante, de ese modo primero se resolverá lo que esta dentro de los paréntesis y finalmente se negará el resultado.

Código: [Seleccionar todo](#)

```
select NOMBRE
  from PERSONAS
 where not ((RUBIA = 'S') and not(ALTA = 'S') and (GAFAS= 'S'))
```

 [llevar código al banco de pruebas](#)

Efectivamente el resultado es justo lo contrario que antes:

NOMBRE
Manuel
Maria
José
Pedro

Y aun otro ejemplo de este operador junto la palabra clave IN: tomemos la consulta que seleccionaba tres registros concretos de la tabla EMPLEADOS, y modifiquémosla únicamente incluyendo el operador NOT para que devuelva lo contrario, es decir, todos los registros menos los tres que antes seleccionaba:

Código: [Seleccionar todo](#)

```
select *
  from EMPLEADOS
 where ID_EMPLEADO not in (1,2,4)
```

 [llevar código al banco de pruebas](#)

Resultado:

ID_EMPLEADO	NOMBRE	APELLIDOS	F_NACIMIENTO	SEXO	CARGO	SALARIO
3	José	Calvo Sisman	1990-11-12	H	Mozo	1400

### El uso de paréntesis

Los paréntesis se comportan como en matemáticas, no es lo mismo "5 + 4 / 3" donde primero se calculará la división y después se sumará 5, que "(5 + 4) / 3" donde primero se resolverá la suma y el resultado parcial se dividirá por 3. Sin paréntesis la división tiene prioridad sobre la suma, con paréntesis forzamos a que la operación se realice en el orden deseado.

Los operadores AND y OR tienen la misma prioridad de modo que se evalúa la expresión por orden de aparición:

No es lo mismo: "RUBIA and ALTA or GAFAS" = "(RUBIA and ALTA) or GAFAS" que, "RUBIA and (ALTA or GAFAS)". En primer caso estamos diciendo: "que sea rubia y alta, o bien lleve gafas", y en el segundo "que sea rubia y, sea alta o lleve gafas".

Código: [Seleccionar todo](#)

```
select NOMBRE
  from PERSONAS
 where RUBIA = 'S' and ALTA = 'S' or GAFAS= 'S'
```

 [llevar código al banco de pruebas](#)

Resultado:

NOMBRE
Manuel
Maria
Carmen
José

Código: [Seleccionar todo](#)

```
select NOMBRE
  from PERSONAS
 where RUBIA = 'S' and (ALTA = 'S' or GAFAS= 'S')
```

 [llevar código al banco de pruebas](#)

Resultado:

NOMBRE
Manuel
Carmen
José

\* \* \*

## Resumen

En esta lección se ha descrito como construir expresiones booleanas y como trabajar con ellas para establecer condiciones en la cláusula WHERE de una consulta SQL.

Las expresiones booleanas con operadores tales como ( $>$ ,  $=$ ,  $\neq$  ...) precisan operandos de tipo número, cadena o fecha, y el resultado de evaluar la expresión devuelve siempre cierto o falso.

Ejemplo: (SALARIO  $>$  1350)

Las expresiones con operadores tales como (AND , OR , NOT) precisan expresiones booleanas como operandos, el conjunto es una nueva expresión booleana que al evaluarla devolverá siempre cierto o falso.

Ejemplo: RUBIA = 'S' and ALTA = 'S'

El uso de paréntesis garantiza que, en una expresión compleja las expresiones simples que la forman se evalúen en el orden que usted desea.

Ejemplo: not ( (RUBIA = 'S' or ALTA = 'S') and (ALTA = 'N' or GAFAS = 'S') )

\* \* \*

## Ejercicio 1

Cree una consulta SQL que devuelva las personas que son altas, o bien son rubias con gafas.

## Ejercicio 2

Cree una consulta SQL que devuelva los empleados que son mujer y cobran más de 1300 euros.

En la tabla empleados se guarda una "H" en el campo SEXO para indicar que es hombre, o una "M" para indicar que es mujer.

## Ejercicio 3

Usando solo expresiones (ALTA = 'S') , (RUBIA = 'S') , (GAFAS = 'S') combinadas con el operador NOT resuelva:

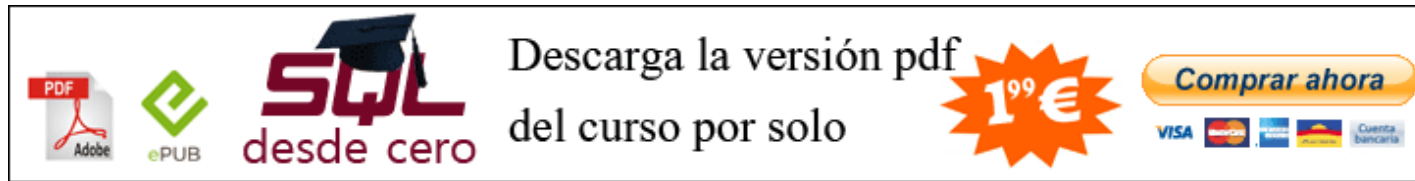
¿Quién es quién? Lleva gafas y no es alta ni rubia.

## Ejercicio 4 (optativo)

Suponiendo que A vale cierto y B vale falso, evalúe la siguiente expresión booleana:

$C = ((A \text{ and } B) \text{ and } (A \text{ or } (A \text{ or } B))) \text{ or } A$





Este curso está sujeto a la licencia Reconocimiento-NoComercial-SinObraDerivada 3.0 de Creative Commons. Usted puede copiarla, distribuirla y comunicarla públicamente siempre que especifique su autor y [deletesql.com](http://deletesql.com); no la utilice para fines comerciales; y no haga con ella obra derivada. Puede usted consultar la licencia completa [aquí](#).

[Arriba](#)

[Volver a Curso SQL desde cero](#)

Saltar a:

## ¿Quién está conectado?

Usuarios navegando por este Foro: No hay usuarios registrados visitando el Foro y 1 invitado

- [Índice general](#)
- [El Equipo](#) • [Borrar todas las cookies del Sitio](#) • Todos los horarios son UTC + 1 hora

Powered by [phpBB](#) © 2000, 2002, 2005, 2007 phpBB Group  
Traducción al español por [Huan Manwë](#)