Table of Contents

[1. Compound criteria 1](#_Toc363598153)

[1.1. The AND logical operator 1](#_Toc363598154)

[1.2. The OR logical operator 3](#_Toc363598155)

[1.3. The NOT logical operator 5](#_Toc363598156)

[1.4. Xor 6](#_Toc363598157)

[2. Hierarchy of evaluation of the logical operators 7](#_Toc363598158)

[3. DeMorgan's laws 9](#_Toc363598159)

[4. Three-way logic and truth tables 9](#_Toc363598160)

For more interesting queries, we can use compound criteria. These are criteria that contain multiple conditions joined with the logical operators AND, OR, and NOT.

1. Compound criteria

For more interesting queries, we can use compound criteria. These are criteria that contain multiple conditions joined with the logical operators AND, OR, and NOT.

* 1. The AND logical operator

With this operator, the compound test has a true value if both conditions are true.

1. We want to see employees hired in 2008.

Select emp\_id, name\_last as "Employee", hire\_date, salary

From a\_emp.employees

Where hire\_date BETWEEN '2008-01-01' AND '2008-12-31';

+--------+----------+------------+----------+

| emp\_id | Employee | hire\_date | salary |

+--------+----------+------------+----------+

| 101 | Koch | 2008-06-17 | 98005.00 |

| 145 | Russ | 2008-03-30 | 65000.00 |

| 205 | Higgs | 2008-06-01 | 15000.00 |

+--------+----------+------------+----------+

1. We want to see employees hired in 2008 who earn more than 50000. A row has to pass both tests to be included in the result set

Select emp\_id, name\_last as "Employee", hire\_date, salary

From a\_emp.employees

Where hire\_date BETWEEN '2008-01-01' AND '2008-12-31'

AND salary > 50000

;

+--------+----------+------------+----------+

| emp\_id | Employee | hire\_date | salary |

+--------+----------+------------+----------+

| 101 | Koch | 2008-06-17 | 98005.00 |

| 145 | Russ | 2008-03-30 | 65000.00 |

+--------+----------+------------+----------+

When we AND in another filter we will generally reduce the number of rows returned by the query.

1. We want to see jobs that do not seem to be in Sales with a minimum salary more than 40000. We cannot be certain that these are all of the non-sales jobs- just that they are jobs which do not have Sales in the job title.

Select job\_id, min\_salary, max\_salary

From a\_emp.jobs

Where job\_title NOT LIKE '%Sales%'

AND min\_salary > 40000;

+--------+------------+------------+

| job\_id | min\_salary | max\_salary |

+--------+------------+------------+

| 16 | 60000.00 | 120000.00 |

| 32 | 60000.00 | NULL |

| 64 | 60000.00 | NULL |

| 128 | 60000.00 | NULL |

+--------+------------+------------+

1. This shows employees with a salary between 12000 and 25000

Select emp\_id, name\_last as "Employee", salaryFrom a\_emp.employees

Where salary between 12000 and 25000

order by salary;

+--------+----------+----------+

| emp\_id | Employee | salary |

+--------+----------+----------+

| 108 | Green | 12000.00 |

| 109 | Fiet | 15000.00 |

| 160 | Dorna | 15000.00 |

| 161 | Dewal | 15000.00 |

| 201 | Harts | 15000.00 |

| 204 | King | 15000.00 |

| 205 | Higgs | 15000.00 |

| 100 | King | 24000.00 |

+--------+----------+----------+

8 rows in set (0.00 sec)

1. If you need to exclude the end point, then use fld > x and fld < y.

Select emp\_id, name\_last as "Employee", salary From a\_emp.employees

Where salary > 12000

AND salary < 25000

Order by salary;

+--------+----------+----------+

| emp\_id | Employee | salary |

+--------+----------+----------+

| 109 | Fiet | 15000.00 |

| 160 | Dorna | 15000.00 |

| 161 | Dewal | 15000.00 |

| 201 | Harts | 15000.00 |

| 204 | King | 15000.00 |

| 205 | Higgs | 15000.00 |

| 100 | King | 24000.00 |

+--------+----------+----------+

7 rows in set (0.00 sec)

1. Avoid writing tests that logically can never have a True value.

Select emp\_id, name\_last as "Employee", salary From a\_emp.employees

Where salary < 12000

AND salary > 25000

Order by salary;

Empty set (0.00 sec)

1. You are not limited to combining two tests.

Select emp\_id, name\_last as "Employee"

, hire\_date, salary, job\_id

From a\_emp.employees

Where hire\_date between '1985-01-01' and '2005-12-31'

AND salary > 15000

AND job\_id in (8, 16)

;

+--------+----------+------------+----------+--------+

| emp\_id | Employee | hire\_date | salary | job\_id |

+--------+----------+------------+----------+--------+

| 155 | Hiller | 2004-03-05 | 80000.00 | 8 |

+--------+----------+------------+----------+--------+

1 row in set (0.00 sec)

1. Earlier we had a row constructor with an equality test

Select prod\_id, prod\_name, catg\_id, prod\_warranty\_period

From a\_prd.products

Where row(catg\_id, prod\_warranty\_period ) = row('HW', 12);

We could do this with an AND test.

Select prod\_id, prod\_name, catg\_id, prod\_warranty\_period

From a\_prd.products

Where catg\_id= 'HW' and prod\_warranty\_period = 12;

* 1. The OR logical operator

With this operator, the compound test has a true value if either one or both conditions are true.

1. Find employees who work in either dept 20 or 30. It would be better to use an IN operator for this test. Notice that you have to repeat the full test for each OR clause.

Select emp\_id, name\_last as "Employee", dept\_idFrom a\_emp.employees

Where dept\_id = 30

OR dept\_id = 20

order by `Employee`;

+--------+----------+---------+

| emp\_id | Employee | dept\_id |

+--------+----------+---------+

| 110 | Chen | 30 |

| 109 | Fiet | 30 |

| 206 | Geitz | 30 |

| 108 | Green | 30 |

| 201 | Harts | 20 |

| 205 | Higgs | 30 |

| 204 | King | 30 |

| 101 | Koch | 30 |

| 203 | Mays | 30 |

+--------+----------+---------+

9 rows in set (0.00 sec)

1. Here we want employees who earn more than 50000

Select emp\_id, name\_last as "Employee", hire\_date, salary, job\_id

From a\_emp.employees

Where salary > 50000;

+--------+------------+----------+--------+

| emp\_id | hire\_date | salary | job\_id |

+--------+------------+----------+--------+

| 101 | 2008-06-17 | 98005.00 | 16 |

| 145 | 2008-03-30 | 65000.00 | 4 |

| 146 | 2011-06-15 | 88954.00 | 64 |

| 155 | 2004-03-05 | 80000.00 | 8 |

| 162 | 2011-03-17 | 98000.00 | 16 |

| 200 | 2011-06-17 | 65000.00 | 16 |

| 206 | 2011-06-15 | 88954.00 | 32 |

| 207 | 2011-06-17 | 65000.00 | 8 |

+--------+------------+----------+--------+

8 rows in set (0.00 sec)

1. Here we want employees who earn more than 50000 or who were hired between 1985 and 2005

Select emp\_id, name\_last as "Employee", hire\_date, salary, job\_id

From a\_emp.employees

Where hire\_date between '1985-01-01' and '2005-12-31'

OR salary > 50000;

+--------+------------+----------+--------+

| emp\_id | hire\_date | salary | job\_id |

+--------+------------+----------+--------+

| 100 | 1989-06-17 | 24000.00 | 1 |

| 101 | 2008-06-17 | 98005.00 | 16 |

| 108 | 1995-04-14 | 12000.00 | 16 |

| 145 | 2008-03-30 | 65000.00 | 4 |

| 146 | 2012-02-29 | 88954.00 | 64 |

| 150 | 2001-10-28 | 6500.00 | 8 |

| 155 | 2004-03-05 | 80000.00 | 8 |

| 162 | 2011-03-17 | 98000.00 | 16 |

| 200 | 2011-06-17 | 65000.00 | 16 |

| 201 | 2004-08-25 | 15000.00 | 2 |

| 206 | 2011-06-15 | 88954.00 | 32 |

| 207 | 2011-06-17 | 65000.00 | 8 |

+--------+------------+----------+--------+

12 rows in set (0.00 sec)

1. Now we add another possibility - that the employee's job id is 8 or 16

Select emp\_id, name\_last as "Employee" ,hire\_date, salary, job\_id

From a\_emp.employees

Where hire\_date between '1985-01-01' and '2005-12-31'

OR salary > 15000

OR job\_id in (8, 16);

+--------+------------+----------+--------+

| emp\_id | hire\_date | salary | job\_id |

+--------+------------+----------+--------+

| 100 | 1989-06-17 | 24000.00 | 1 |

| 101 | 2008-06-17 | 98005.00 | 16 |

| 108 | 1995-04-14 | 12000.00 | 16 |

| 145 | 2008-03-30 | 65000.00 | 4 |

| 146 | 2012-02-29 | 88954.00 | 64 |

| 150 | 2001-10-28 | 6500.00 | 8 |

| 155 | 2004-03-05 | 80000.00 | 8 |

| 161 | 2011-06-15 | 15000.00 | 16 |

| 162 | 2011-03-17 | 98000.00 | 16 |

| 200 | 2011-06-17 | 65000.00 | 16 |

| 201 | 2004-08-25 | 15000.00 | 2 |

| 203 | 2010-06-30 | 44450.00 | 16 |

| 205 | 2008-06-01 | 15000.00 | 16 |

| 206 | 2011-06-15 | 88954.00 | 32 |

| 207 | 2011-06-17 | 65000.00 | 8 |

+--------+------------+----------+--------+

15 rows in set (0.00 sec)

With each additional Or clause we add, we have the potential of having more rows match.

1. We had a previous query for max\_salary >= 20000 Here we are also including the nulls with an IS NULL test

Select job\_id, job\_title, min\_salary, max\_salary

From a\_emp.jobs

Where max\_salary >= 20000

OR max\_salary is null;

+--------+---------------+------------+------------+

| job\_id | job\_title | min\_salary | max\_salary |

+--------+---------------+------------+------------+

| 1 | President | 10000.00 | 100000.00 |

| 2 | Marketing | 5000.00 | 75000.00 |

| 4 | Sales Manager | 15000.00 | 60000.00 |

| 8 | Sales Rep | 10000.00 | 30000.00 |

| 16 | Programmer | 60000.00 | 120000.00 |

| 32 | Code Debugger | 60000.00 | NULL |

| 64 | DBA | 60000.00 | NULL |

| 128 | RD | 60000.00 | NULL |

+--------+---------------+------------+------------+

* 1. The NOT logical operator

1. The NOT operator works on a single test and reverses the value of that test. The NOT test is commonly used in combination with AND or OR tests.
2. We want employees who are **not** in department 20 or 30. This could also be written as a NOT IN test.

Select emp\_id, name\_last as "Employee", dept\_idFrom a\_emp.employees

Where NOT dept\_id IN ( 30, 20)

order by `Employee`;

+--------+----------+---------+

| emp\_id | Employee | dept\_id |

+--------+----------+---------+

| 102 | D'Haa | 215 |

| 161 | Dewal | 215 |

| 160 | Dorna | 215 |

| 104 | Ernst | 210 |

| 155 | Hiller | 80 |

| 162 | Holme | 35 |

| 103 | Hunol | 210 |

| 100 | King | 10 |

| 146 | Partne | 215 |

| 207 | Russ | 35 |

| 145 | Russ | 80 |

| 150 | Tuck | 80 |

| 200 | Whale | 35 |

+--------+----------+---------+

* 1. Xor

MySQL supports the XOr operator; this is used when you have two logical expressions and you test that they have different truth values. This is not commonly used but sometimes it is the easiest way to write a query.

1. This is a simple OR. Rows are returned if the dept id is 30 or if the salary < 200 or if both are true. We have some rows for people from dept 30 with a low salary

Select emp\_id, name\_last as Employee, dept\_id , salary

From a\_emp.employees

Where dept\_id =30

or salary < 20000

order by dept\_id , salary;

+--------+----------+---------+----------+

| emp\_id | Employee | dept\_id | salary |

+--------+----------+---------+----------+

| 201 | Harts | 20 | 15000.00 |

| 108 | Green | 30 | 12000.00 |

| 205 | Higgs | 30 | 15000.00 |

| 109 | Fiet | 30 | 15000.00 |

| 204 | King | 30 | 15000.00 |

| 110 | Chen | 30 | 30300.00 |

| 203 | Mays | 30 | 44450.00 |

| 206 | Geitz | 30 | 88954.00 |

| 101 | Koch | 30 | 98005.00 |

| 150 | Tuck | 80 | 6500.00 |

| 103 | Hunol | 210 | 9000.00 |

| 161 | Dewal | 215 | 15000.00 |

| 160 | Dorna | 215 | 15000.00 |

+--------+----------+---------+----------+

13 rows in set (0.00 sec)

1. With the XOR operation a person who is in dept 30 and who has a low salary is not returned. We get rows for people in dept 30 who do not have a low salary and we get rows for people with a low salary who are not in dept 30.

Select emp\_id, name\_last as Employee, dept\_id , salary

From a\_emp.employees

Where dept\_id =30

Xor salary < 20000

order by dept\_id , salary;

+--------+----------+---------+----------+

| emp\_id | Employee | dept\_id | salary |

+--------+----------+---------+----------+

| 201 | Harts | 20 | 15000.00 |

| 110 | Chen | 30 | 30300.00 |

| 203 | Mays | 30 | 44450.00 |

| 206 | Geitz | 30 | 88954.00 |

| 101 | Koch | 30 | 98005.00 |

| 150 | Tuck | 80 | 6500.00 |

| 103 | Hunol | 210 | 9000.00 |

| 160 | Dorna | 215 | 15000.00 |

| 161 | Dewal | 215 | 15000.00 |

+--------+----------+---------+----------+

9 rows in set (0.00 sec)

1. Hierarchy of evaluation of the logical operators

If you write a criterion that includes more than one logical operator, you need to be concerned about the hierarchy of evaluation. The order of operations is first the NOT operators are evaluated then the ANDs and then the ORs. Parentheses are used to change the order of operations.

Suppose we want to see products that are either pet supplies or sporting goods that cost less than 100. We can look at the current set of data and find the following rows.

PROD\_I PROD\_LIST\_PRICE CATG\_I

------ --------------- ------

1020 12.95 SPG

1030 29.95 SPG

1140 14.99 PET

1141 99.99 PET

1142 2.5 PET

1150 4.99 PET

1151 14.99 PET

1152 55 PET

4576 29.95 PET

4577 29.95 PET

1. This query following the wording of the task description but does not do the job. We have two Pet items that cost more than $100.

Select prod\_id, prod\_list\_price, catg\_id

From a\_prd.products

Where catg\_id = 'PET' or catg\_id = 'SPG'

And prod\_list\_price < 100;

+---------+-----------------+---------+

| prod\_id | prod\_list\_price | catg\_id |

+---------+-----------------+---------+

| 1020 | 12.95 | SPG |

| 1030 | 29.95 | SPG |

| 1140 | 14.99 | PET |

| 1141 | 99.99 | PET |

| 1142 | 2.50 | PET |

| 1150 | 4.99 | PET |

| 1151 | 14.99 | PET |

| 1152 | 55.00 | PET |

| 4567 | 549.99 | PET |

| 4568 | 549.99 | PET |

| 4576 | 29.95 | PET |

| 4577 | 29.95 | PET |

+---------+-----------------+---------+

12 rows in set (0.00 sec)

1. If we reverse the testing of the two categories, we get sporting goods items that cost more than $100. That is not right.

Select prod\_id, prod\_list\_price, catg\_id

From a\_prd.products

Where catg\_id = 'SPG' or catg\_id = 'PET'

And prod\_list\_price < 100;

+---------+-----------------+---------+

| prod\_id | prod\_list\_price | catg\_id |

+---------+-----------------+---------+

| 1010 | 150.00 | SPG |

| 1020 | 12.95 | SPG |

| 1030 | 29.95 | SPG |

| 1040 | 349.95 | SPG |

| 1050 | 269.95 | SPG |

| 1060 | 255.95 | SPG |

| 1140 | 14.99 | PET |

| 1141 | 99.99 | PET |

| 1142 | 2.50 | PET |

| 1150 | 4.99 | PET |

| 1151 | 14.99 | PET |

| 1152 | 55.00 | PET |

| 4576 | 29.95 | PET |

| 4577 | 29.95 | PET |

+---------+-----------------+---------+

What is happening here is that we have an AND operator and an OR operator. The rules of precedence is that the AND operator is evaluated first. So the second of these where clauses

where catg\_id = 'SPG' or catg\_id = 'PET' and prod\_list\_price < 100;

is evaluated as shown here and all of the sporting goods items are returned and Pet supplies that cost more than $100 are returned.

where catg\_id = 'SPG' or (catg\_id = 'PET' and prod\_list\_price < 100);

We can use parentheses to change the order of evaluation. The order of precedence for these operators is:

NOT

AND

XOR

OR

1. Adding the parentheses gives us the correct result.

Select prod\_id, prod\_list\_price, catg\_id

From a\_prd.products

Where (catg\_id = 'SPG' or catg\_id = 'PET')

And prod\_list\_price < 100;

+---------+-----------------+---------+

| prod\_id | prod\_list\_price | catg\_id |

+---------+-----------------+---------+

| 1020 | 12.95 | SPG |

| 1030 | 29.95 | SPG |

| 1140 | 14.99 | PET |

| 1141 | 99.99 | PET |

| 1142 | 2.50 | PET |

| 1150 | 4.99 | PET |

| 1151 | 14.99 | PET |

| 1152 | 55.00 | PET |

| 4576 | 29.95 | PET |

| 4577 | 29.95 | PET |

+---------+-----------------+---------+

1. It is better to use the IN operator, avoiding the AND/OR Issue.

Select prod\_id, prod\_list\_price, catg\_id

From a\_prd.products

Where catg\_id IN ( 'SPG', 'PET')

And prod\_list\_price < 100;

+---------+-----------------+---------+

| prod\_id | prod\_list\_price | catg\_id |

+---------+-----------------+---------+

| 1020 | 12.95 | SPG |

| 1030 | 29.95 | SPG |

| 1140 | 14.99 | PET |

| 1141 | 99.99 | PET |

| 1142 | 2.50 | PET |

| 1150 | 4.99 | PET |

| 1151 | 14.99 | PET |

| 1152 | 55.00 | PET |

| 4576 | 29.95 | PET |

| 4577 | 29.95 | PET |

+---------+-----------------+---------+

1. DeMorgan's laws

Often, there is more than one way to write a complex logical expression. The following equivalencies are known as DeMorgan's Laws.

Where expP and expQ represent logical expressions

NOT (expP AND expQ) is equivalent to NOT expP OR NOT expQ

NOT (expP OR expQ) is equivalent to NOT expP AND NOT expQ

1. Three-way logic and truth tables

Generally we think of logical expressions having two possible values — True and False. Because database systems allow the use of Null, we have to be concerned with three logical values —True, False, and Unknown. Suppose we have a row in the jobs table with no value for the attribute max\_salary , and we evaluate the logical expression: max\_salary > 25000 the value of the expression is Unknown for that row. If you are executing a query with a Where clause, if the value of the test is Unknown, the row is not returned.

Remember, NULL is a data value, UNKNOWN is a logical value.

These are the truth tables for the operators NOT, AND, Or and XOR.

The evaluation of the True and False cases are straight forward. With the NOT operator, if I do not know the value of an expression is True or False then I do not know if the negation of that expression is True or False.

|  |  |
| --- | --- |
| NOT |  |
| True | False |
| Unknown | Unknown |
| False | True |

For the AND operator to Return True both of the operands must have a True value. So if one of the operands is True and the other is unknown, then I cannot know if the ANDed expression is true- so the value is unknown. But if one of the operands is False, then the ANDed expression cannot be true and we know its value is False.

|  |  |  |  |
| --- | --- | --- | --- |
| AND | True | Unknown | False |
| True | True | Unknown | False |
| Unknown | Unknown | Unknown | False |
| False | False | False | False |

For the OR operator to Return True at least one of the operands must have a True value. So if one of the operands is True and the other is unknown, then the ORed expression is TRUE. If one of the operands is False and the other is unknown then I cannot know the value of the Ored expression and its value is Unknown.

|  |  |  |  |
| --- | --- | --- | --- |
| OR | True | Unknown | False |
| True | True | True | True |
| Unknown | True | Unknown | Unknown |
| False | True | Unknown | False |

For the XOR operator to Return one of the operands must have a True value and the other operand a False value. If both operands and True or if both operands are false then the result is False.

|  |  |  |  |
| --- | --- | --- | --- |
| XOR | True | Unknown | False |
| True | False | Unknown | True |
| Unknown | Unknown | Unknown | Unknown |
| False | True | Unknown | False |