Table of Contents

1. Row filters

Most of our queries so far have returned all of the data from the tables in the From clause. We are working with very small tables. Imagine the output if we had thousands of rows in our tables. We seldom want to see all of the data in a table. Rather we want to see only a subset of the data- a subset that matches some search condition. If you are using the CCSF WebStars system to see your current schedule- you do not want to see everyone's schedule- just yours.

To filter the output add a WHERE clause to the SQL statement after the From clause. The Where clause specifies which rows should be returned. The Where clause contains a logical expression (a predicate or test) that is applied to each row in the table. If the logical expression evaluates to true for a row from the table (if that row passes the test), that row is returned in the result set. The row is not returned if the predicate evaluates as False or as Unknown. We will cover a variety of comparison operators and conditional expressions this week and next week. We will also investigate the concept of Unknown..

Our query model is now

```
Select col_expressions
From table_expression
WHERE predicate
Order By sort_keys;
```

In this unit, we will look at several filters: we will discuss more filters soon.

- the Is Null and Is Not Null filters
- the In list filter
- the Between filter

A few general rules for writing filters

- You cannot test a column alias in a Where clause; you need to test using the column name.
- Character literals are enclosed in single quotes: 'CA', 'Anderson'
- In tests against character literals, leading blanks are significant. The value `'      CA'` does not match the value `'CA'`. Trailing blanks are not significant.
- Numbers are not enclosed in quotes; do not include punctuation such as commas or dollar signs when you write numeric literals
- Dates are more complex; for now write date literals in the standard default format: '2008-04-01'. Note that this is a string literal; MySQL will cast it to a date for date testing.

For these queries the table expressions will follow the format of databaseName.tableName such as

```
from a_oe.order_headers
```

This means you can run these queries from within any of your databases.

2. Testing for nulls

You may want to write your queries to skip any rows where certain column values are null. The way to test for this is to add a Where clause after the From clause that filters for the nulls.

Testing for missing values requires the use of the IS NULL operator. Think of IS NULL and IS NOT NULL as single operators. You use the same operator IS NULL for any data type that you are testing.

If you try to test using the syntax = NULL you will get no rows returned. This is not flagged as an error.

Demo 01: Test for empty attributes by using the test IS NULL. You use the same test for attributes of any data type.

```
Select ord_id, ord_date, sales_rep_id, shipping_mode
From a_oe.order_headers
WHERE shipping_mode IS NULL;
+--------+---------------------+--------------+---------------+
| ord_id | ord_date            | sales_rep_id | shipping_mode |
+--------+---------------------+--------------+---------------+
|    116 | 2012-11-12 00:00:00 |          155 | NULL          |
|    117 | 2012-11-28 00:00:00 |          150 | NULL          |
|    118 | 2012-11-28 00:00:00 |          150 | NULL          |
|    119 | 2012-11-28 00:00:00 |          155 | NULL          |
|    120 | 2013-01-02 00:00:00 |         NULL | NULL          |
|    121 | 2013-01-03 00:00:00 |         NULL | NULL          |
+--------+---------------------+--------------+---------------+
```

Demo 02: Test for empty attributes by using the test IS NULL. This is testing an integer column.

```
Select ord_id, ord_date, sales_rep_id
From a_oe.order_headers
WHERE sales_rep_id IS NULL;
+--------+---------------------+--------------+
| ord_id | ord_date            | sales_rep_id |
+--------+---------------------+--------------+
|    115 | 2012-11-08 00:00:00 |         NULL |
|    120 | 2013-01-02 00:00:00 |         NULL |
|    121 | 2013-01-03 00:00:00 |         NULL |
|    525 | 2012-05-09 00:00:00 |         NULL |
|    527 | 2012-05-01 00:00:00 |         NULL |
+--------+---------------------+--------------+
```

Demo 03: Test for empty attributes by using the test IS NULL. I do not have to show the column being tested.

```
Select prod_id, prod_name, prod_list_price
From a_prd.products
WHERE prod_warranty_period IS NULL;
+---------+---------------------+-----------------+
| prod_id | prod_name           | prod_list_price |
+---------+---------------------+-----------------+
|    2014 | B000005INR          |           15.95 |
|    2234 | B000002I7U          |           15.88 |
|    2337 | B000005H40          |           15.87 |
. . . rows ommitted
|    4567 | Deluxe Cat Tree     |          549.99 |
|    4568 | Deluxe Cat Bed      |          549.99 |
|    4569 | Mini Dryer          |          349.95 |
|    4575 | Electric can opener |          49.95 |
|    4576 | Cosmo cat nip       |          29.95 |
|    4577 | Cat leash           |          29.95 |
|    5000 | Fingerling Potatoes |          12.50 |
|    5001 | Ginger Preserve     |           5.00 |
+---------+---------------------+----------------+
```

Demo 04: We can use IS NOT NULL to find rows that have a data value.

```
Select ord_id, ord_date, shipping_mode
From a_oe.order_headers
WHERE shipping_mode IS NOT NULL
Order by ord_date
Limit 5;
+--------+---------------------+---------------+
| ord_id | ord_date            | shipping_mode |
+--------+---------------------+---------------+
|    522 | 2011-02-05 00:00:00 | USPS1         |
|    123 | 2011-12-05 00:00:00 | USPS1         |
|    124 | 2011-12-07 00:00:00 | FEDEX1        |
|    125 | 2011-12-09 00:00:00 | FEDEX1        |
|    127 | 2011-12-15 00:00:00 | USPS1         |
+--------+---------------------+---------------+
```

3. The IN list test

Suppose we want to display all orders for customers with id 402100, 400300,or 40330. This is testing against a specific set of values and we can use an IN list for this. The list of values is enclosed in parentheses and the values are separated by commas.

3.1. Simple In lists

Demo 05: Using the IN list for numeric values. It is not an error to have a customer id (40330) in the list which does not match any rows in the table. Numeric literals are not delimited.

```
Select ord_id
, ord_date
, cust_id
From a_oe.order_headers
WHERE cust_id IN (402100, 400300, 40330);
+--------+---------------------+---------+
| ord_id | ord_date            | cust_id |
+--------+---------------------+---------+
|    378 | 2012-07-14 00:00:00 |  400300 |
|    114 | 2012-11-08 00:00:00 |  402100 |
|    115 | 2012-11-08 00:00:00 |  402100 |
|    117 | 2012-11-28 00:00:00 |  402100 |
+--------+---------------------+---------+
```

Demo 06: Using the IN list for numeric values.

```
Select ord_id
, ord_date
, cust_id
From a_oe.order_headers
WHERE ord_id IN (101, 107, 95, 125)
;
```

```
+--------+---------------------+---------+
| ord_id | ord_date            | cust_id |
+--------+---------------------+---------+
|    107 | 2012-10-02 00:00:00 |  403050 |
|    125 | 2011-12-09 00:00:00 |  409160 |
+--------+---------------------+---------+
```

Demo 07: You can use the NOT IN test to exclude specified data values.

```
Select prod_id, prod_name, catg_id
From a_prd.products
WHERE catg_id NOT IN ('HW', 'PET');
+---------+---------------------+---------+
| prod_id | prod_name           | catg_id |
+---------+---------------------+---------+
|    1010 | Weights             | SPG     |
|    1020 | Dartboard           | SPG     |
. . .  rows omitted
|    1130 | Mini Freezer        | APL     |
|    4569 | Mini Dryer          | APL     |
|    5000 | Fingerling Potatoes | GFD     |
|    5001 | Ginger Preserve     | GFD     |
|    5002 | Ball-Peen Hammer    | HD      |
|    5004 | Dead Blow hammer    | HD      |
|    5005 | Shingler Hammer     | HD      |
|    5008 | Claw Framing        | HD      |
+---------+---------------------+---------+
```

Demo 08: You can use a list that contains only one item.

```
Select prod_id, prod_name, catg_id, prod_warranty_period
From a_prd.products
WHERE prod_warranty_period IN (18);
+---------+------------+---------+----------------------+
| prod_id | prod_name  | catg_id | prod_warranty_period |
+---------+------------+---------+----------------------+
| 1080    | Cornpopper | HW      |                   18 |
| 1100    | Blender    | HW      |                   18 |
+---------+------------+---------+----------------------+
```

Demo 09: You can use a NOT IN list that contains only one item.

```
Select prod_id, prod_name, catg_id, prod_warranty_period
From a_prd.products
WHERE prod_warranty_period  NOT IN (18);
+---------+--------------------+---------+----------------------+
| prod_id | prod_name          | catg_id | prod_warranty_period |
+---------+--------------------+---------+----------------------+
| 1000    | Hand Mixer         | HW      |                   12 |
| 1010    | Weights            | SPG     |                   60 |
| 1020    | Dartboard          | SPG     |                   60 |
| 1030    | Basketball         | SPG     |                   60 |
| 1040    | Treadmill          | SPG     |                   60 |
| 1050    | Stationary bike    | SPG     |                   60 |
| 1060    | Mountain bike      | SPG     |                   36 |
| 1070    | Iron               | HW      |                   36 |
| 1071    | Iron               | HW      |                   36 |
| 1072    | Iron                | HW      |                    36 |
| 1090    | Gas grill           | HW      |                    12 |
| 1110    | Pancake griddle     | HW      |                    12 |
| 1120    | Washer              | APL     |                    12 |
| 1125    | Dryer               | APL     |                    12 |
. . .  rows omitted
+---------+--------------------+---------+---------------------+
```

The above two queries may look like they should return all rows in one or the other of these queries. But they do not return the rows where the prod_warranty_period is null. For that you need to test with the IS NULL test.

Demo 10: Test with IS NULL

```
Select prod_id, prod_name, catg_id, prod_warranty_period
From a_prd.products
WHERE prod_warranty_period  IS NULL;
+---------+---------------------+---------+----------------------+
| prod_id | prod_name           | catg_id | prod_warranty_period |
+---------+---------------------+---------+----------------------+
. . .  rows omitted
|    4567 | Deluxe Cat Tree     | PET     |                 NULL |
|    4568 | Deluxe Cat Bed      | PET     |                 NULL |
|    4569 | Mini Dryer          | APL     |                 NULL |
|    4575 | Electric can opener | HW      |                 NULL |
|    4576 | Cosmo cat nip       | PET     |                 NULL |
|    4577 | Cat leash           | PET     |                 NULL |
|    5000 | Fingerling Potatoes | GFD     |                 NULL |
|    5001 | Ginger Preserve     | GFD     |                 NULL |
+---------+---------------------+---------+----------------------+
```

3.2. In lists that contain nulls

If you try putting a Null in a list you will find that it does not match a row with a null. The rule is that a row is returned if the Where predicate evaluates as True; a null in the table does not match a null in the list. A null value does not match another null value. The logical value of a null matching a null is Unknown; the value of the filter expression must be True for the row to be returned.

Demo 11: Trying to use Null in a list.

```
Select prod_id, prod_name, catg_id, prod_warranty_period
From a_prd.products
WHERE prod_warranty_period IN (18, NULL);
+---------+------------+---------+----------------------+
| prod_id | prod_name  | catg_id | prod_warranty_period |
+---------+------------+---------+----------------------+
| 1080    | Cornpopper | HW      |                   18 |
| 1100    | Blender    | HW      |                   18 |
+---------+------------+---------+----------------------+
```

What may seem more surprising is the following query, which returns no rows. The rule is that a row is returned if the Where predicate evaluates as True; the row is not returned if the predicate evaluates as False or as Unknown. Here we are negating the Unknown value which is still Unknown.

Demo 12: Nulls always get interesting

```
Select prod_id, prod_name, catg_id, prod_warranty_period
From a_prd.products
WHERE prod_warranty_period  NOT IN (18, NULL);
Empty set (0.00 sec)
```

3.3. In lists that contain row values

You can test constructed rows with an In test. In MySQL you can use the expression row(30, 101) to refer to a two part value. The word row is optional; I will use it here to emphasis that we are comparing multi-part row values.

Demo 13: Suppose we wanted to find employees in dept 30 with manager 101. We could use the following. The row values is (30, 101) and it is enclosed in parentheses for the In list. The two columns we are comparing are also enclosed in parentheses.

```
Select emp_id, name_last, dept_id,  emp_mng
From a_emp.employees
Where row(dept_id,  emp_mng) IN( row(30, 101) );
+--------+-----------+---------+---------+
| emp_id | name_last | dept_id | emp_mng |
+--------+-----------+---------+---------+
|    108 | Green     |      30 |     101 |
|    203 | Mays      |      30 |     101 |
|    205 | Higgs     |      30 |     101 |
+--------+-----------+---------+---------+
```

Demo 14: Now suppose we wanted to find employees in dept 30 with manager 101 and also employees in dept 35 with manager 101.

```
Select emp_id
 , name_last
 , dept_id
 , emp_mng
From a_emp.employees
Where row( dept_id, emp_mng ) in (
    row( 30, 101 )
  , row( 35, 101 )
  )
Order by dept_id, emp_mng;
+--------+-----------+---------+---------+
| emp_id | name_last | dept_id | emp_mng |
+--------+-----------+---------+---------+
|    108 | Green     |      30 |     101 |
|    203 | Mays      |      30 |     101 |
|    205 | Higgs     |      30 |     101 |
|    162 | Holme     |      35 |     101 |
|    200 | Whale     |      35 |     101 |
+--------+-----------+---------+---------+
```

Demo 15: You could rewrite this without the keyword row- but include the parentheses which make this a row.

```
Select emp_id, name_last, dept_id,  emp_mng
From   a_emp.employees
Where (dept_id,  emp_mng) IN( (30, 101), (35, 101) )
Order by dept_id, emp_mng;
```

3.4. Testing a literal against an in list

Commonly we think of testing a column against an In list of literals. But with some tests we can reverse that type of thinking. Suppose we are working with the vets database and we are told that the vet got a message that "Edgar is sick" but the message does not say if Edgar is a client or an animal. You can use a Where clause such as this to look for a match for the literal "Edgar with any of these columns.

```
   where 'Edger' in ( cl_name_last, cl_name_first, an_name);
```

Demo 16: And then run this query which looks for the literal Edger in three columns(don't worry about the join in the From clause at this time)

```
   Select cl_name_last, cl_name_first, an_name
   From vt_clients C
   left join vt_animals  A  on C.cl_id = a.cl_id
   Where 'Edger' in ( cl_name_last, cl_name_first, an_name);
```

```
cl_name_last              cl_name_first             an_name
------------------------- ------------------------- -------------------------
Harris                    Eddie                     Edger
Boston                    Edger                     NULL
```

This could use useful for searches where you do not know if a value was entered as a person's first or last name; that happens fairly frequently when people fill out forms.

4. The BETWEEN test

To test data against a range of values, use BETWEEN. The Between test is an inclusive test. If the row being tested matches an end point of the range, the test has a true value, and the row will get into the output display. The range should be an increasing range. If you test `WHERE salary BETWEEN 12000 and 3000` the query will run but no rows will be returned.

Demo 17: Using BETWEEN with a numeric range.

```
   Select emp_id, name_last AS "Employee", salary
   From a_emp.employees
   Where salary BETWEEN 3000 AND 12000;
   +--------+----------+----------+
   | emp_id | Employee | salary   |
   +--------+----------+----------+
   |    103 | Hunol    |  9000.00 |
   |    108 | Green    | 12000.00 |
   |    150 | Tuck     |  6500.00 |
   +--------+----------+----------+
```

Demo 18: Using NOT BETWEEN to exclude values in the range.

```
   Select emp_id, name_last AS "Employee", salary
   From a_emp.employees
   Where salary NOT BETWEEN 10000 AND 60000
   Order by salary;
   +--------+----------+----------+
   | emp_id | Employee | salary   |
   +--------+----------+----------+
   |    150 | Tuck     |  6500.00 |
   |    103 | Hunol    |  9000.00 |
   |    207 | Russ     | 65000.00 |
   |    145 | Russ     | 65000.00 |
   |    200 | Whale    | 65000.00 |
   |    155 | Hiller   | 80000.00 |
   |    146 | Partne   | 88954.00 |
   |    206 | Geitz    | 88954.00 |
   |    162 | Holme    | 98000.00 |
   |    101 | Koch     | 98005.00 |
   +--------+----------+----------+
```

Demo 19: Using BETWEEN with a date range.

```
Select emp_id, name_last AS "Employee", hire_date
From a_emp.employees
Where hire_date BETWEEN '2001-01-01' AND '2007-12-31';
+--------+----------+------------+
| emp_id | Employee | hire_date  |
+--------+----------+------------+
|    150 | Tuck     | 2001-10-28 |
|    155 | Hiller   | 2004-03-05 |
|    201 | Harts    | 2004-08-25 |
+--------+----------+------------+
```

Demo 20: Using BETWEEN with character range.

```
Select emp_id, name_last AS "Employee",dept_id
From a_emp.employees
Where name_last BETWEEN 'J' and 'T'
Order by name_last;
+--------+----------+---------+
| emp_id | Employee | dept_id |
+--------+----------+---------+
|    100 | King     |      10 |
|    204 | King     |      30 |
|    101 | Koch     |      30 |
|    203 | Mays     |      30 |
|    146 | Partne   |     215 |
|    145 | Russ     |      80 |
|    207 | Russ     |      35 |
+--------+----------+---------+
```

You need to be careful with character range tests. If we had an employee with a last name composed of just the letter T, that employee would be returned. But the employee with the name Tuvault is not returned.

Demo 21: Customer with a low credit rating

```
Select cust_id, cust_name_last, cust_name_first, credit_limit
From a_oe.customers
Where credit_limit between 0 and 1000;
+---------+----------------+-----------------+--------------+
| cust_id | cust_name_last | cust_name_first | credit_limit |
+---------+----------------+-----------------+--------------+
|  400801 | Washington     | Geo             |          750 |
|  401250 | Morse          | Samuel          |          750 |
|  402100 | Morise         | William         |          750 |
|  402110 | Coltrane       | John            |          750 |
|  402120 | McCoy          | Tyner           |          750 |
+---------+----------------+-----------------+--------------+
```

Demo 22: But customers with no credit rating were not returned by the previous query.

```
Select cust_id, cust_name_last, cust_name_first, credit_limit
From a_oe.customers
Where credit_limit is null;
+---------+----------------+-----------------+--------------+
| cust_id | cust_name_last | cust_name_first | credit_limit |
+---------+----------------+-----------------+--------------+
|  402500 | Jones          | Elton John      |         NULL |
|  405000 | Day            | David           |         NULL |
+---------+----------------+-----------------+--------------+
```

4.1. Gotchas

Some tests to watch out for with the use of Between.

The Between operator will match no rows if the range is descending or if one of the range points is a null.

Demo 23: Range is decreasing

```
Select emp_id, name_last AS "Employee", salary
From a_emp.employees
Where salary BETWEEN 3000 AND 1000
;
Empty set (0.00 sec)
```

Demo 24: One end of the range is a null

```
Select emp_id, name_last AS "Employee", salary
From a_emp.employees
Where salary BETWEEN 3000 AND null
;
Empty set (0.00 sec)
```

Demo 25: One end of the range is a null

```
Select emp_id, name_last AS "Employee", salary
From a_emp.employees
Where salary BETWEEN null AND 1000
;
Empty set (0.00 sec)
```

Using Between with datetime values can also be a problem. The ex_date in the vets exam headers table is a date time value.

Demo 26: These are all the exam dates in date order

```
Select stf_id, ex_date
From a_vets.vt_exam_headers
Order by ex_date;
+--------+---------------------+
| stf_id | ex_date             |
+--------+---------------------+
|     15 | 2011-11-23 10:30:00 |
|     25 | 2011-12-09 09:00:00 |
|     15 | 2011-12-22 09:00:00 |
|     25 | 2011-12-22 09:00:00 |
|     29 | 2011-12-23 12:15:00 |
|     15 | 2011-12-26 09:30:00 |
|     29 | 2012-01-03 14:30:00 |
|     29 | 2012-01-06 10:45:00 |
|     29 | 2012-01-07 10:45:00 |
|     29 | 2012-01-16 09:15:00 |
|     29 | 2012-02-03 14:30:00 |
|     29 | 2012-02-03 14:30:00 |
|     29 | 2012-02-22 10:45:00 |
|     29 | 2012-02-25 10:45:00 |
|     15 | 2012-03-06 10:30:00 |
|     29 | 2012-03-06 10:45:00 |
|     29 | 2012-03-21 10:45:00 |
|     29 | 2012-03-27 10:45:00 |
|     15 | 2012-04-01 16:30:00 |
|     29 | 2012-04-29 14:30:00 |
|     15 | 2012-05-01 16:30:00 |
|     15 | 2012-05-23 08:30:00 |
|     15 | 2012-06-06 10:30:00 |
|     29 | 2012-06-13 10:45:00 |
|     15 | 2012-06-30 10:30:00 |
|     15 | 2012-07-31 13:00:00 |
|     15 | 2013-01-02 13:00:00 |
|     15 | 2013-01-02 13:00:00 |
|     38 | 2013-01-02 15:30:00 |
|     38 | 2013-01-02 16:30:00 |
+--------+---------------------+
```

Demo 27: But if I filter for exam dates in June 2012 by using between 2012-06-01 and 2012-06-30, I will not get the exam that occurred on June 30, 2012 at 10:30. If you do not include a time component, then the datetime value gets a default time component of midnight.

```
Select stf_id, ex_date
From a_vets.vt_exam_headers
Where ex_date between '2012-06-01' and '2012-06-30';
+--------+---------------------+
| stf_id | ex_date             |
+--------+---------------------+
|     15 | 2012-06-06 10:30:00 |
|     29 | 2012-06-13 10:45:00 |
+--------+---------------------+
```

Demo 28: I could include a time components for the end of the range

```
Select stf_id, ex_date
From a_vets.vt_exam_headers
Where ex_date between '2012-06-01' and '2012-06-30 23:59:59';
+--------+---------------------+
| stf_id | ex_date             |
+--------+---------------------+
|     15 | 2012-06-06 10:30:00 |
|     29 | 2012-06-13 10:45:00 |
|     15 | 2012-06-30 10:30:00 |
+--------+---------------------+
```