Table of Contents

1. Predicates and nulls

The expressions we have been using in the Where clause are called predicates- they are expressions that have a truth value- that value can be True, False, or Unknown. Let's consider the z_name value in the zoo table. Suppose we write the following query:

```
select   z_id, z_type, z_name
from     zoo
where    z_name in ('Sam', 'Sue')
;
```

This query is executed by looking at each row in the animals table, one row at a time, evaluating that expression. These are the rows I currently have in that table.

```
+------+-----------+---------+
| z_id | z_type    | z_name  |
+------+-----------+---------+
|   23 | Giraffe   | Sam     |
|   25 | Armadillo | Abigail |
|   56 | Lion      | Leon    |
|   57 | Lion      | Lenora  |
|   85 | Giraffe   | Sally   |
|   43 | Zebra     | Huey    |
|   44 | Zebra     | Dewey   |
|   45 | Zebra     | Louie   |
|  125 | Horse     | NULL    |
+------+-----------+---------+
```

The query is **logically** executed as getting data from the zoo table into a temporary storage area

Then evaluating the Where clause- looking at the first row

```
23 Giraffe                    Sam
```

and evaluating the expression in the Where clause; for this row that expression evaluates as True and the row is passed into the result set.

Then the query is logically executed as looking at the next row

```
25 Armadillo                  Abigail
```

and evaluating the expression in the Where clause; for this row that expression evaluates as False and the row is not passed into the result set.

Then the query is logically executed as looking at the next row

```
56 Lion                    Leon
```

and evaluating the expression in the Where clause; for this row that expression evaluates as False and the row is not passed into the result set.

The predicate is evaluated independently for each row as the query executes. The value of the predicate is dependent on the value of the column z_name for this row. In the case of the row for z_id 125, the name column is null and the truth value of the expression is unknown.

We are not sure what the human meaning is of the null in the z_name column for this row- it could be that is a horse with no name or it could be that the horse has a name but the person entering the data did not know what the name is; it could even be that the horse is a very private animal and did not want his name entered in the database! But as far as SQL and queries are concerned, the query is evaluated as if we do not know the name and therefore we do not know if the name is Sam or Sue- we just do not know- and so the value of the predicate is Unknown. The rules for a Where clause is that if the predicate evaluates as True then the row is returned into the result set; if the predicate is evaluated as False or a Unknown the row is not returned into the result set.

Suppose we change the Where clause to Where z_name is Null. With that predicate, the value of the predicate is True for the row with z_id = 125 and False for the other rows.

2. Logical processing order

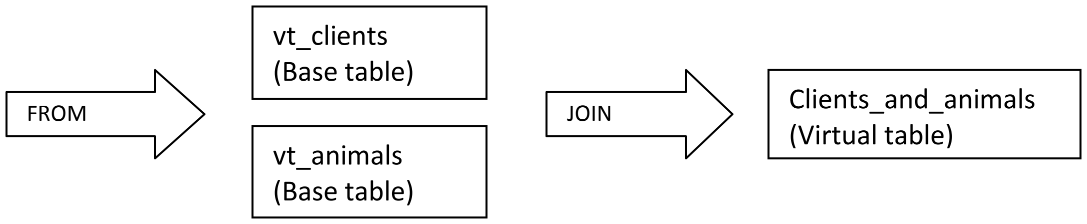
When you create a select statement you write the statements with the following model.

```
   Select   column-expressions
   From     table-expression
   Where    filter_expressions
   Order By sort-keys;
```

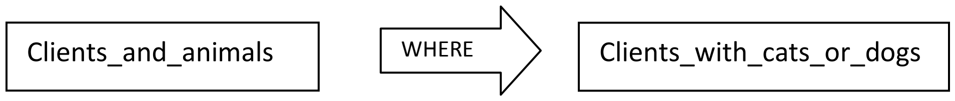
When the query is presented to the dbms, the parser and optimizer determine the actual steps used in the execution and you do not directly control that. But you should consider the statement as being processed in the following order.

1. The FROM clause
2. The WHERE clause
3. The SELECT clause
4. The ORDER BY clause

The From clause assembles a collection of rows from the table or tables to be used. In this unit we start to talk about more complex From clauses- but the purpose of the From clause is to get data from the tables and build the first virtual table. The optimizer ends up determining the order in which these rows are ordered in the virtual table.



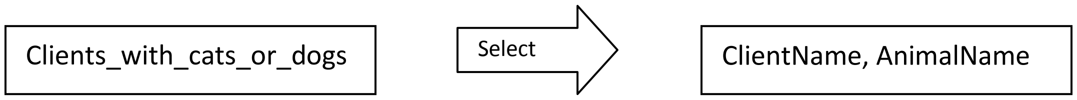
The Where clause filters that first virtual table for the rows that meet the Where clause filters. Assume we have a Where clause that filters for an_type in ('cat', 'dog'). The Where clause now produces a second virtual table that includes only the clients with cats and/or dogs. We have no control over the order of the rows in this virtual table.



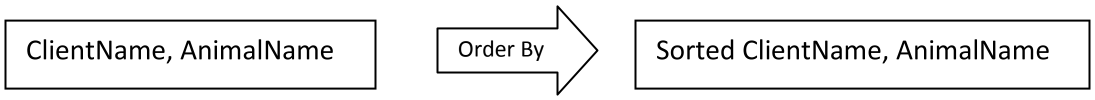
The Select clause now takes that virtual table and returns only the columns and expressions in the Select clause. Perhaps we want to display only the client name and the animal name. We can also provide column aliases at this step. Since the column alias is defined only in the Select clause, MySQL does not let you use column aliases in the Where clause. MySQL also does not let you use the column alias for one of the columns to calculate another column in the select.

The following is not allowed; ClientName cannot be used in the second expression since we do not determine the order in which the columns are calculated.

```
select concat(cl_name_last , ' ', cl_name_first) as ClientName
, concat(ClientName, ' lives in ', cl_state )
from vt_clients;
```



The Order By clause takes that last virtual table and sorts the rows.



Since the column aliases are now defined, we can use them in the Order By clause.

```
select concat(cl_name_last , ' ', cl_name_first) as ClientName
from vt_clients
order by ClientName;
```