## Table of Contents



Someone made a post about table aliases. He could follow the patterns but felt uncertain that he understood then. This is a common situation; we start by following a pattern and the query works. But we are not sure why.

This gets into ambiguous references and into joins.

These examples use the vets database tables

# 1. Ambiguous References

Suppose we have a query that joins the vt_animals table and the vt_exam_headers table. And we want to display the animal id and the exam id. The following will not work. I get an error message that the column name an_id is ambiguous (unclear or inexact because a choice between alternatives has not been made). We have a column named an_id in the animals table and a column named an_id in the vt_exam_headers table. And we did not make it clear which one of these we wanted to use.

```
select an_id, ex_id
from vt_animals
join vt_exam_headers on vt_animals.an_id = vt_exam_headers.an_id;
```

In this example you may say that you don't care because this is an inner join and the value would be the same across the join. But SQL ( like most programming systems) is fussy about things like that. So you have to provide a table reference for an_id in the Select. Now vt_animals.an_id is a fully qualified name.

```
select vt_animals.an_id, ex_id
from vt_animals
join vt_exam_headers on vt_animals.an_id = vt_exam_headers.an_id;
```

In the previous query ex_id is not fully qualified but it is not ambiguous since it only occurs in the exam_headers table. You could write the following:

```
select vt_animals.an_id, vt_exam_headers.ex_id
from vt_animals
join vt_exam_headers on vt_animals.an_id = vt_exam_headers.an_id;
```

An ambiguous column reference means that you have a column name in part of the query (it could be in the Select, the Where clause, the Order by clause- or any place else) and the column name appears in more than one of the tables in the From clause. In the tables I use, I try to avoid using the same column name in multiple tables but a lot of companies use the column names "id" , "name" , "city" etc in many of their tables.

# 2. Table Aliases

Let's use a single table from the vets database: vt_clients. I want to display a number of columns from this table.

```
select cl_id, cl_name_last, cl_name_first, cl_postal_code, cl_city, cl_phone
from  vt_clients
order by cl_id;
```

That will work fine. I have only one table in the From clause, so all of the columns are from the vt_clients table.

Some companies have a rule that query are written only fully qualified names . So I write the query as:

```
select vt_clients.cl_id, vt_clients.cl_name_last, vt_clients.cl_name_first,
vt_clients.cl_postal_code, vt_clients.cl_city, vt_clients.cl_phone
from  vt_clients
order by vt_clients.cl_id;
```

That certainly made the query longer. I could use a table alias to make the query shorter. A table alias is a substitute name for a table that is defined in the From clause. I often use the alias CL for the vt_clients table.

```
select CL.cl_id, CL.cl_name_last, CL.cl_name_first, CL.cl_postal_code,
CL.cl_city, CL.cl_phone
from  vt_clients  CL
order by CL.cl_id;
```

That is certainly shorter, the columns are fully qualified and I think it is easier to read. You will find a lot of disagreements on this topic on various web pages. Some people think it is wrong to qualify names when there is only one table in the From clause; some people think it is inefficient to qualify names when there is only one table in the From clause; other people think it is more efficient. When you read these, it usually comes down to a personal choice. People seldom show any data to support their arguments.

Now suppose we want to also show the animal id and animal name along with the client data. We need two tables.

We can do the following with fully qualified names - no table aliases:

```
select vt_animals.an_id, vt_clients.cl_id, vt_clients.cl_name_last,
vt_clients.cl_name_first, vt_clients.cl_postal_code, vt_clients.cl_city,
vt_clients.cl_phone, vt_animals.an_name
from vt_clients
join vt_animals on vt_animals.cl_id =  vt_clients.cl_id;
```

Or the following with table aliases:

```
select AN.an_id, CL.cl_id, CL.cl_name_last, CL.cl_name_first,
CL.cl_postal_code, CL.cl_city, CL.cl_phone, AN.an_name
from vt_clients  CL
join vt_animals AN  on AN.cl_id =  CL.cl_id;
```

Again I find the version with table aliases easier to read.

You are not required to qualify all column in the above query- only the ones that would cause an ambiguous reference. The following is OK. The only column name that appears in both tables is cl_id. Many SQL shops would have a style rule against this.

```
select an_id, CL.cl_id, cl_name_last, cl_name_first, cl_postal_code, cl_city,
cl_phone, an_name
from vt_clients  CL
join vt_animals AN  on AN.cl_id =  CL.cl_id;
```

There are a few situation where you are required to use table aliases. One, mentioned in this unit's notes, is when you have to join a table to itself. There is no reasonable example of this with the vets database, so see the example in the notes under self join.

One rule regarding table aliases is that once you define an alias in the From clause you have to use in the rest of that query. We sometimes called the table alias an alternate name- it really is more of a substitute name.

The following will not run. You will get an error message about the column vt_clients.cl_name_last. This is one of those errors that does not seem fair- but that is the way that SQL works.

```
select CL.cl_id, vt_clients.cl_name_last
from  vt_clients  CL
order by CL.cl_id;
```

There is no rule that table aliases are limited to single letters. The alias can be longer than the table name if that is more meaningful. Our table names are reasonably meaningful, but some companies have table names like Sem234Zghyp and Sem243Zghyp and if you are trying to write a query that joins these two tables I would highly suggest using table aliases. And maybe even look for another place to work unless they have a good reason for those names.

There is a class rule that the table aliases must be meaningful. There is a rule in programming in general that the names of things should suggest their meaning. The use of table aliases a, b, c, d is not allowed ( unless each of these is meaningful for that table). One reason people don't like table aliases is that is people use aliases that are not meaningful; then the query is hard to understand. When you want to read a query, you should start with the From clause where you will find the aliases. The From clause drives the query. (The Select query really should have the From clause first- but it is too late to change that.)

If you want to type out the full table name for each column reference that is OK with me.

If you want to use table aliases that is OK with me.

If you want to qualify only the columns that would be ambiguous that is OK with me.