Difference between WHERE and ON in SQL

Last modified: February 01, 2021

Is there a difference between the WHERE and ON clause?

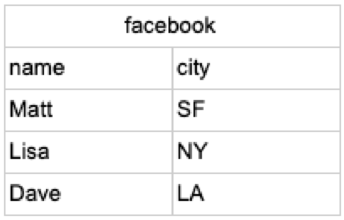
Yes. ON should be used to define the join condition and WHERE should be used to filter the data. I used the word should because this is not a hard rule. The splitting of these purposes with their respective clauses makes the query the most readable, it also prevents incorrect data being retrieved when using JOINs types other than INNER JOIN.

To go more in depth we will cover the two use cases that either WHERE or ON can support:

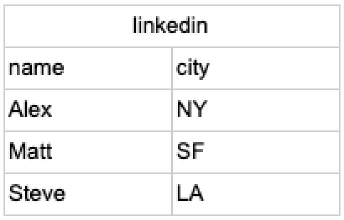
- Joining data
- Filtering data

Joining data

The way both of these clauses can be used to help join data is through defining the condition on which the two tables are joined. To demonstrate this, lets use an example data set of facebook friends and linkedin connections.



facebook	
name	city
Matt	SF
Lisa	NY
Dave	LA



linkedin	
name	city
Alex	NY
Matt	SF
Steve	LA

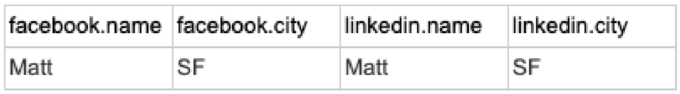
We want to see the people who are both our friend and our connection. So in this case it would only be Matt. Lets now query using a variety of defining the JOIN condition.

All three of these queries produce the same correct result:

```
SELECT *
FROM facebook
JOIN linkedin
ON facebook.name = linkedin.name

SELECT *
FROM facebook
JOIN linkedin
WHERE facebook.name = linkedin.name
```

```
SELECT *
FROM facebook, linkedin
WHERE facebook.name = linkedin.name
```



facebook.name	facebook.city	linkedin.name	linkedin.city
Matt	SF	Matt	SF

The first two are types of explicit joins and the last is an implicit join. An explicit JOIN explicitly tells you how to JOIN the data by specifying the type of JOIN and the join condition in the ON clause. An Implicit JOIN does not specify the JOIN type and use the WHERE clause to define the join condition.

Readability

The main difference between these queries is how easy it is to understand what is going on. In the first query we can easily see the tables being joined in the FROM and JOIN clause. We can also clearly see the join condition in the ON clause. In the second query it seems just as clear however we may do a double take on the WHERE clause since this is typically used to filter data and not JOIN it. In the last query we have to look closely to both establish what table are being JOINed and how they are being JOINed.

The last query is using what is called an implicit JOIN(a JOIN that is not explicitly stated in the query. In most cases implicit JOINs will act as INNER JOINs. If you want to use a JOIN other than an INNER JOIN stating it explicitly makes it clear what is going on.

JOINing in the WHERE clause can be confusion since this is not it's typical purpose. It is most often used to filter the data. So when more filtering conditions are added to the WHERE clause in addition to using it to define how to JOIN the data it becomes harder to understand.

```
SELECT *
FROM facebook, linkedin
WHERE facebook.name = linkedin.name AND (facebook.name = Matt OR linkedin.city = "

SELECT *
FROM facebook
JOIN linkedin
ON facebook.name = linkedin.name
WHERE facebook.name = Matt OR linkedin.city = "SF"
```

Even though the first query has fewer characters than the second it is not as easily understood.

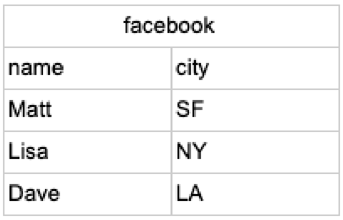
Optimization

Sometimes writing a query in a different way can yield speed improvements. However in this case there should be no speed benefits because of something called a query plan. A query plan is the code that SQL comes up with to execute the query. It takes the query and then creates an optimized way to find the data. Using WHERE or ON to JOIN the data should produce the same query plan.

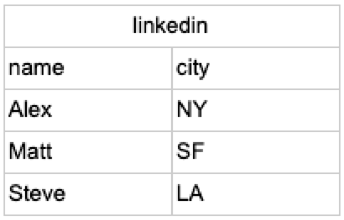
However the way query plans are created may vary across SQL languages and versions, again in this instance it should all be the same but you can test it out on your Database to see if you get anymore performance. Be careful of caching affecting the results of your queries.

Filtering data

Both the ON and WHERE clause can be used to filter data in a query. There are readability and accuracy concerns to address with filtering in the ON clause. Let's use a slightly larger data set to demonstrate this.



facebook	
name	city
Matt	SF
Lisa	NY
Dave	LA



linkedin	
name	city
Alex	NY
Matt	SF
Steve	LA

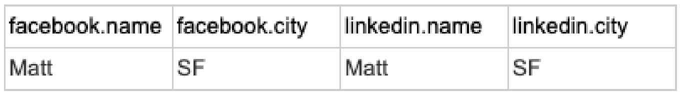
This time we are looking for which people are both our friends and connections, but we only want to see the one(s) who also live in SF.

Readability

Let's evaluate how readable each option is, these two queries will produce the same output:

```
SELECT *
JOIN linkedin
ON facebook.name = linkedin.name
WHERE facebook.city = 'SF'

SELECT *
FROM facebook
JOIN linkedin
ON facebook.name = linkedin.name AND facebook.city = 'SF'
```



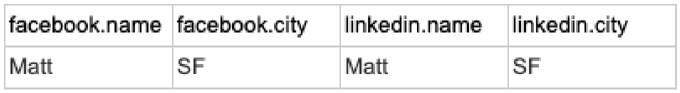
facebook.name	facebook.city	linkedin.name	linkedin.city
Matt	SF	Matt	SF

The first query is clear, each clause has its own purpose. The second query is more difficult to understand because the ON clause is being used to both JOIN the data and filter it.

Accuracy

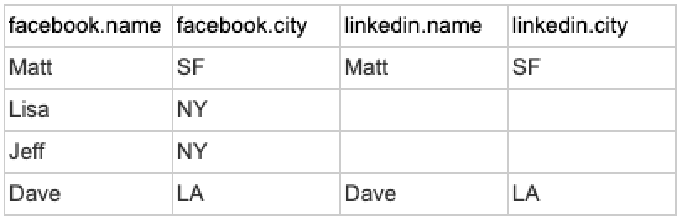
Filtering in the ON clause may produce unexpected results when using a LEFT, RIGHT, or OUTER JOIN. These two queries will not produce the same output:

```
SELECT *
FROM facebook
LEFT JOIN linkedin
ON facebook.name = linkedin.name
WHERE facebook.city = 'SF'
```



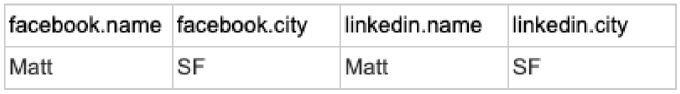
facebook.name	facebook.city	linkedin.name	linkedin.city
Matt	SF	Matt	SF

In a LEFT JOIN it brings in every row from the first table “facebook” and joins wherever the join condition is true (facebook.name = linkedin.name) this would be true for both Matt and Dave. So the interim table would have been.



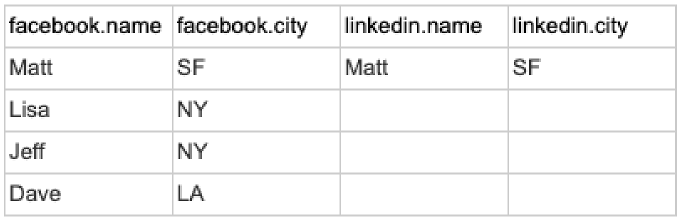
facebook.name	facebook.city	linkedin.name	linkedin.city
Matt	SF	Matt	SF
Lisa	NY		
Jeff	NY		
Dave	LA	Dave	LA

Then the WHERE clause filters these result to rows where facebook.city = ‘SF’, leaving the one row.



facebook.name	facebook.city	linkedin.name	linkedin.city
Matt	SF	Matt	SF

```
SELECT *
FROM facebook
LEFT JOIN linkedin
ON facebook.name = linkedin.name AND facebook.city = 'SF'
```



facebook.name	facebook.city	linkedin.name	linkedin.city
Matt	SF	Matt	SF
Lisa	NY		
Jeff	NY		
Dave	LA		

The join condition is different in this query. The LEFT JOIN brings in every row and the data that is JOINed in from linkedin only happens when facebook.name = linkedin.name AND facebook.city = ‘SF’. It does not filter out all of the rows that didn’t have facebook.city = ‘SF’

Optimization

There is potential variation here of how the query plan is constructed so there might be benefits with trying out filtering in the ON. Some SQL languages may filter while joining and others may wait until the full table is built before filtering. The first plan would be faster.

Summary

Keep the context separate between **joining** the tables and **filtering** the joined table. It is the most readable, least likely to be inaccurate, and should not be less performant.

- JOIN data in ON
- Filter data in WHERE
- Write explicit JOINs to make your Query more readable
- Filter data in the WHERE clause instead of the JOIN to ensure it is correct and readable
- Different SQL languages may have different query plans based on filtering in the ON clause vs the WHERE clause, so test the performance on your database

Written by: Matt David
Reviewed by:

Find more free data resources at DataSchool.com

- Home
- Web Books
- Contributors
- Mission
- Contribute
- Slack Community

Our Web Books

- Data Conversations
- Cloud Data Management
- How to Design a Dashboard
- How to Teach People SQL
- Learn SQL
- Avoid Misrepresenting Data
- SQL Optimization
- Fundamentals of Analysis