

Home » Blog » A Join A Day – The Left Anti Semi Join

A Join A Day – The Left Anti Semi Join

2012-12-14 - A Join A Day, Fundamentals, General, Series

Introduction

This is the fourteenth post in my [A Join A Day](#) series about SQL Server Joins. Make sure to let me know how I am doing or ask your burning join related questions by leaving a comment below.

Today we are going to talk about the Left Anti Semi Join. And with that it is time for another short Latin lesson: "Anti"

The prefix "anti" means as much as "opposite of". While most dictionaries list it as originating from the Latin language, its roots are actually deeper in ancient greek: ἀντί.

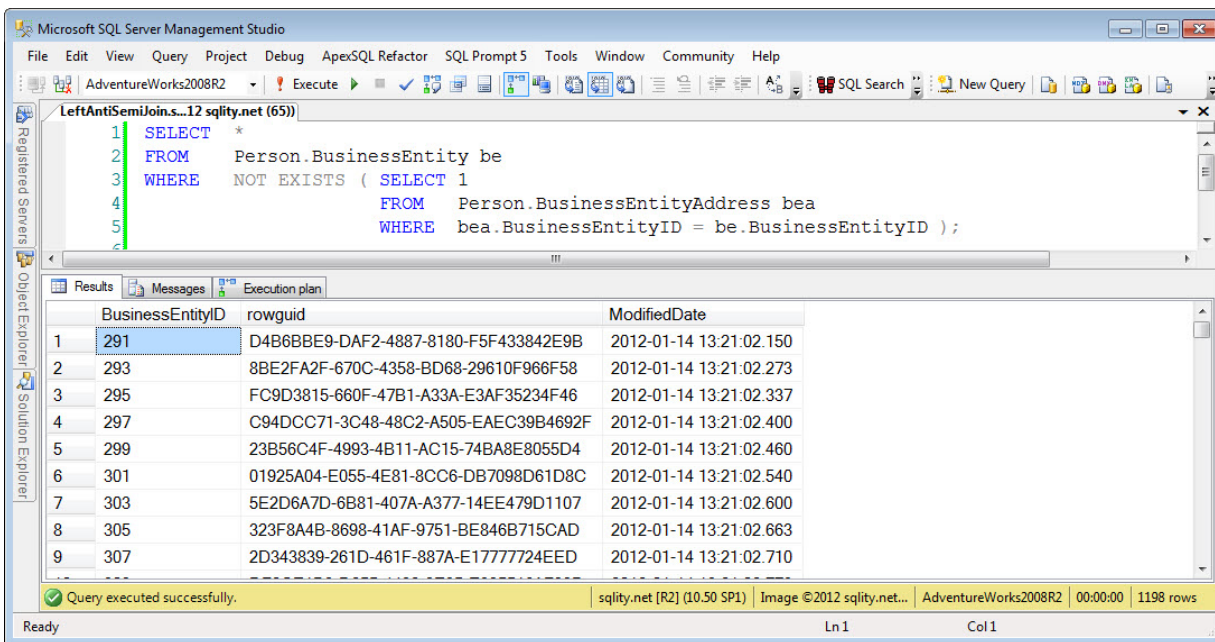
So, the Left Anti Semi Join is the opposite of a [Left Semi Join](#). However, that does not make it a [right semi join](#). Instead "Anti" affects which rows are returned and which aren't. Like the Left Semi Join, the Left Anti Semi Join returns only rows from the left row source. Each row is also returned at most once. And duplicates are also not eliminated. However, other than the Left Semi Join, the Left Anti Semi Join returns only rows for which no match on the right side exists. Let's look at an example.

LEFT ANTI SEMI JOIN Example

The easiest way to write a query with a logical left anti semi join is a `NOT EXISTS()` query like this:

```
[sql] SELECT *  
FROM Person.BusinessEntity be  
WHERE NOT EXISTS ( SELECT 1  
FROM Person.BusinessEntityAddress bea  
WHERE bea.BusinessEntityID = be.BusinessEntityID );  
[/sql]
```

This query returns the 1198 entities that do not have an address at all:



So exactly all the entities that did not get returned by the Left Semi Join are the ones that the Left Anti Semi Join is returning.

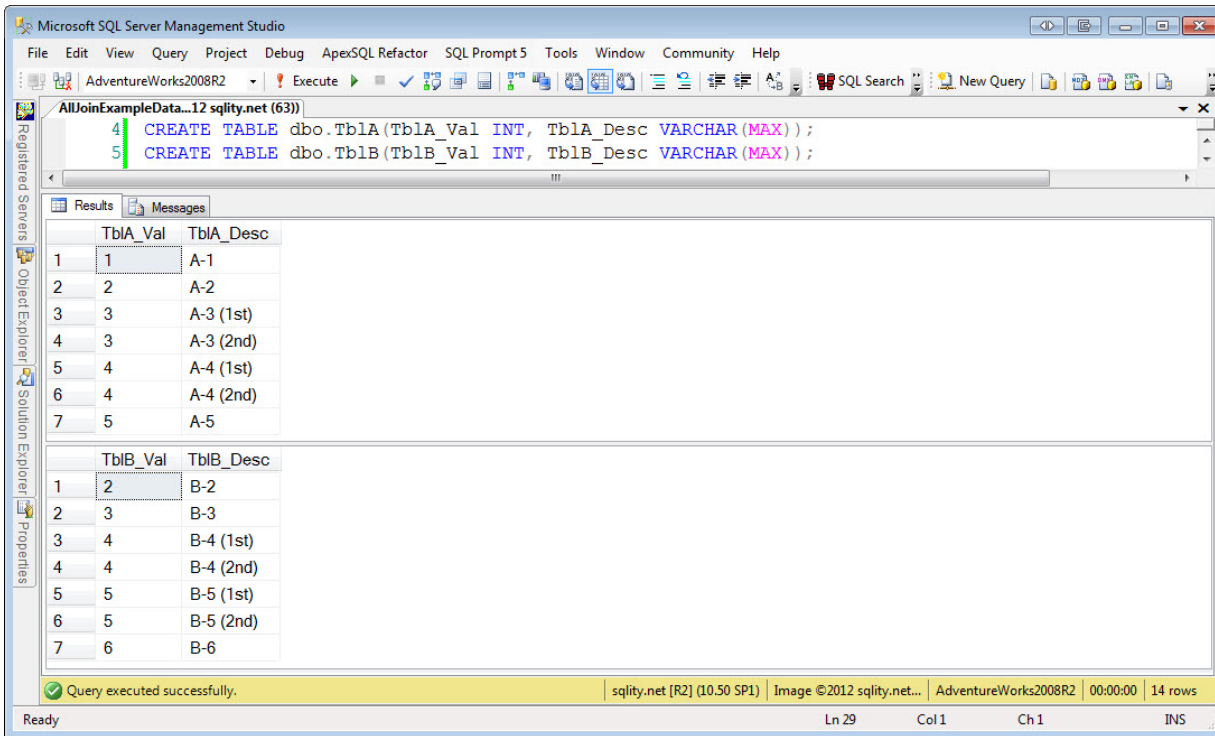
Let's look at the Tb1A - Tb1B example again with this query:

```

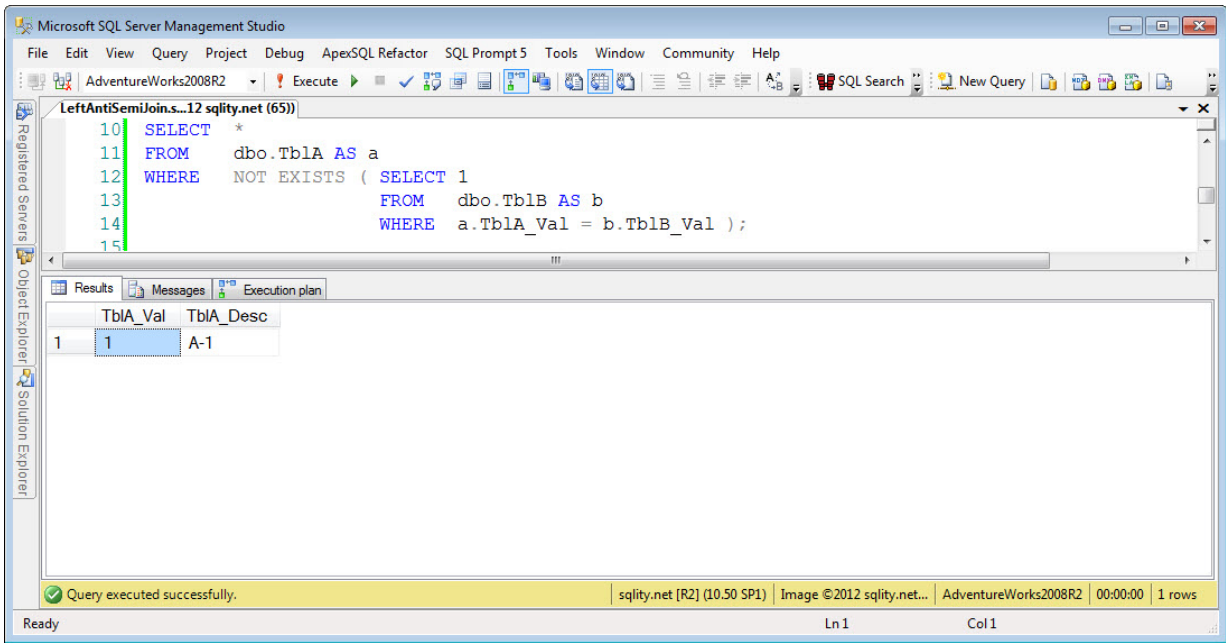
[sql] SELECT *
FROM dbo.Tb1A AS a
WHERE NOT EXISTS ( SELECT 1
FROM dbo.Tb1B AS b
WHERE a.Tb1A_Val = b.Tb1B_Val );
[/sql]

```

As always, this is the data in the two tables:



The query returns just a single row:



The row `TblA_Val = 1` is the only row that does not have a match in `TblB`. Therefore it is the only one returned by this query.

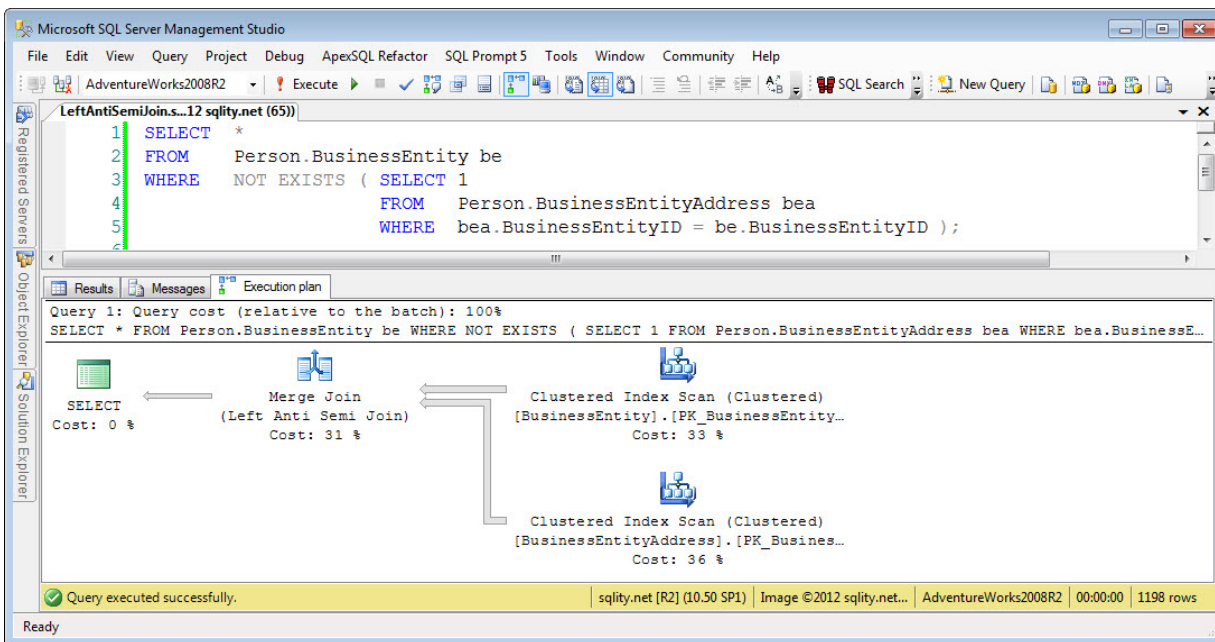
LEFT ANTI SEMI JOIN Syntax

There is again no statement or command that can be directly linked to a Left Anti Semi Join. There are many ways to write a query like this. However, the `NOT EXISTS()` syntax used in both examples above is the most common way to write a Left Anti Semi Join query.

LEFT ANTI SEMI JOIN Operator

Like with the Left Semi Join, there is no command or statement that will always lead to a Left Anti Semi Join operator. The optimizer will consider the Left Anti Semi Join operator for any query that fits the model. However there is no guarantee that the operator will be used.

For above `BusinessEntity-without-Address` query the optimizer selected a Left Anti Semi Join operator:

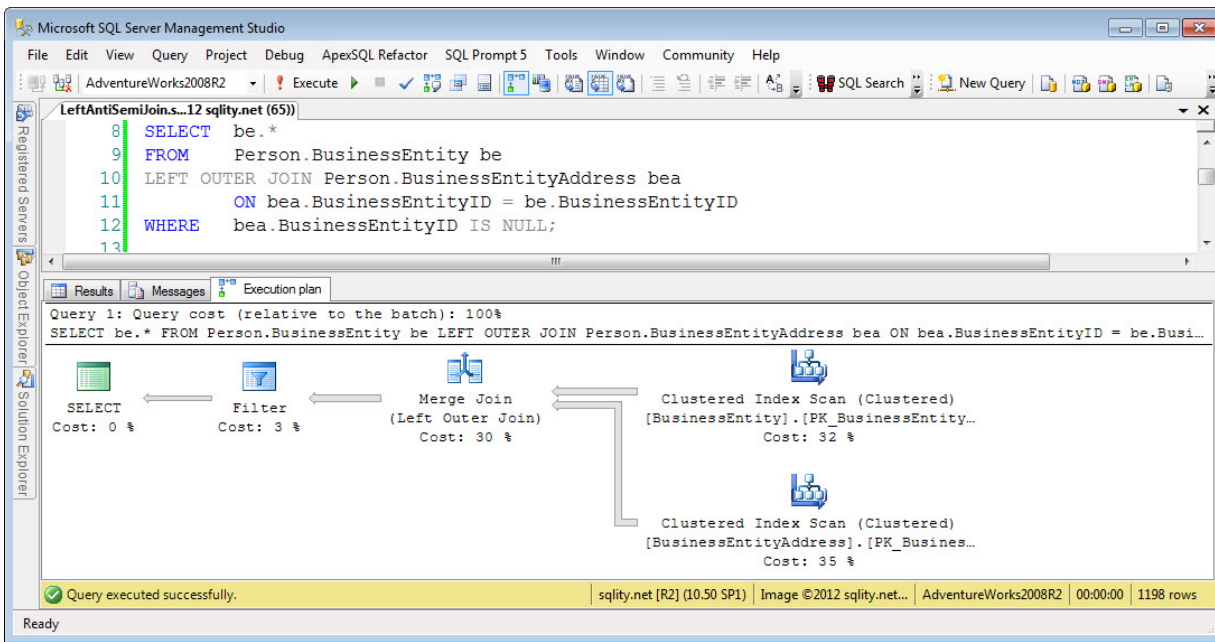


However, let's look at this logically equivalent query:

```
[sql] SELECT be.*
FROM Person.BusinessEntity be
LEFT OUTER JOIN Person.BusinessEntityAddress bea
ON bea.BusinessEntityID = be.BusinessEntityID
WHERE bea.BusinessEntityID IS NULL;
[/sql]
```

It first finds executes a left outer join between the **BusinessEntity** table and the **BusinessEntityAddress** table. Of the 20812 rows that come out of the left outer join, most get filtered out by the where clause. Only the 1198 rows that do not have a match get returned.

Even though this query is logically equivalent it ends up with a different execution plan:



The execution plan reflects exactly the way the query is written. First a Left Outer Join operator joins the two tables together and then an additional Filter operator removes the rows for which a match exists. Because of the additional operator and because of the 20812 rows that get passed into it, only to get discarded en mass (only 1198 rows survive) this execution plan is actually less efficient than the Left Anti Semi Join execution plan.

The SQL Server optimizer cannot do an exhaustive search of all possible plans when compiling a query. Therefore sometimes a sub-optimal plan gets selected. However, the team developing the optimizer is constantly working on improving this already amazing piece of software and it is possible that with the next version or even the next service pack both queries will result in the same plan.

Summary

The Left Anti Semi Join filters out all rows from the left row source that have a match coming from the right row source. Only the orphans from the left side are returned. While there is a Left Anti Semi Join operator, there is no direct SQL command to request this operator. However, the `NOT EXISTS()` syntax shown in the above examples will often result in this operator being used.

A Join A Day

This post is part of my December 2012 "A Join A Day" blog post series. You can find the table of contents with all posts published so far in the introductory post: [A Join A Day – Introduction](#). Check back there frequently throughout the month.

Categories: [A Join A Day](#), [Fundamentals](#), [General](#), [Series](#)

6 Responses to *A Join A Day – The Left Anti Semi Join*

Leave a Reply

You must be [logged in](#) to post a comment.



We
kno
w
ho
w
to
ma
ke a
dat
aba

The 10 Most Common Database Vulnerabilities

The 10 Most Common Database Vulnerabilities

1. SQL Injection
SQL injection is a type of attack where an attacker inserts a malicious SQL statement into an input field. This can allow the attacker to view or modify data in the database. [Learn more](#)

2. Cross-Site Scripting (XSS)
Cross-site scripting (XSS) is a type of attack where an attacker inserts a malicious script into a web page. This can allow the attacker to steal data or perform other malicious actions. [Learn more](#)

3. Denial of Service (DoS)
Denial of service (DoS) is a type of attack where an attacker attempts to make a system unavailable to its intended users. This can be done by flooding the system with traffic or by exploiting a vulnerability in the system. [Learn more](#)

4. Buffer Overflow
A buffer overflow is a type of attack where an attacker writes more data to a buffer than it can hold. This can cause the program to crash or execute arbitrary code. [Learn more](#)

5. Remote File Inclusion (RFI)
Remote file inclusion (RFI) is a type of attack where an attacker includes a remote file into a web page. This can allow the attacker to execute arbitrary code on the server. [Learn more](#)

6. Local File Inclusion (LFI)
Local file inclusion (LFI) is a type of attack where an attacker includes a local file into a web page. This can allow the attacker to view or modify files on the server. [Learn more](#)

7. Command Injection
Command injection is a type of attack where an attacker injects a command into a program. This can allow the attacker to execute arbitrary commands on the server. [Learn more](#)

8. Directory Traversal
Directory traversal is a type of attack where an attacker traverses the directory structure of a file system. This can allow the attacker to view or modify files on the server. [Learn more](#)

9. Integer Overflow
Integer overflow is a type of attack where an attacker causes an integer to exceed its maximum value. This can cause the program to crash or execute arbitrary code. [Learn more](#)

10. Floating Point Overflow
Floating point overflow is a type of attack where an attacker causes a floating point number to exceed its maximum value. This can cause the program to crash or execute arbitrary code. [Learn more](#)

sqrlty.net

Get Yours Now!

