MSSQLTips Free SQL Server Conference Registration + Travel Expenses click to learn more

Eliminating bookmark (key/rid) lookups

By: Greg Robidoux

Overview

When we were looking at the index scan and table scan section we were able to eliminate the scan which was replaced with an index seek, but this also introduced a **Key Lookup** which is something else you may want to eliminate to improve performance.

A key lookup occurs when data is found in a non-clustered index, but additional data is needed from the clustered index to satisfy the query and therefore a lookup occurs. If the table does not have a clustered index then a **RID Lookup** occurs instead.

In this section we will look at how to find Key/RID Lookups and ways to eliminate them.

Explanation

The reason you would want to eliminate Key/RID Lookups is because they require an additional operation to find the data and may also require additional I/O. I/O is one of the biggest performance hits on a server and any way you can eliminate or reduce I/O is a performance gain.

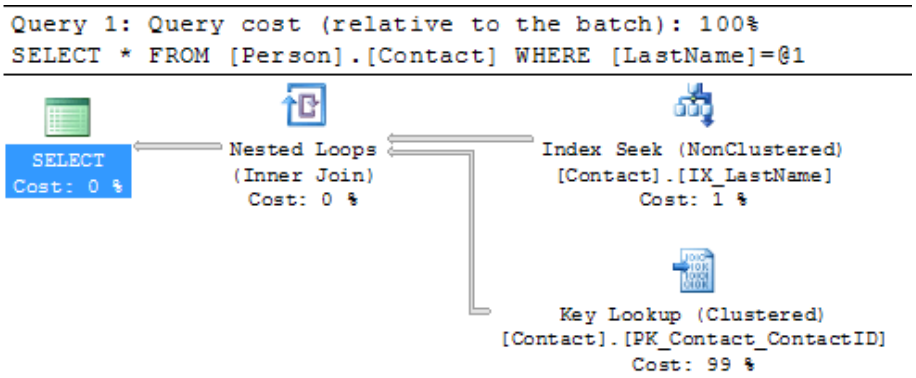
So let's take a look at an example query and the query plan. Before we do this we want to first add the nonclustered index on LastName.

```
USE [AdventureWorks]
GO
CREATE NONCLUSTERED INDEX [IX_LastName]
ON [Person].[Contact] ([LastName])
GO
```

Now we can use Ctrl+M to turn on the actual execution plan and run the select.

```
SELECT * FROM Person.Contact WHERE LastName = 'Russell'
```

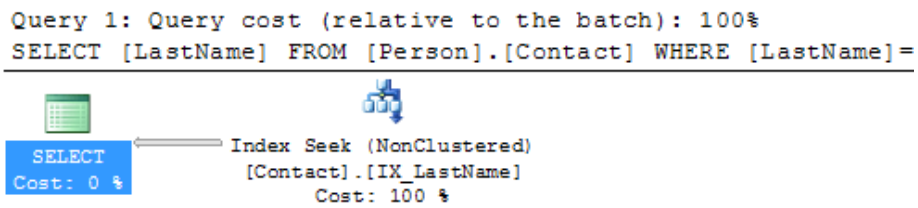
If we look at the execution plan we can see that we have an Index Seek using the new index, but we also have a Key Lookup on the clustered index. The reason for this is that the nonclustered index only contains the LastName column, but since we are doing a SELECT * the query has to get the other columns from the clustered index and therefore we have a Key Lookup. The other operator we have is the Nested Loops this joins the results from the Index Seek and the Key Lookup.



So if we change the query as follows and run this again you can see that the Key Lookup disappears, because the index includes all of the columns.

```
SELECT LastName FROM Person.Contact WHERE LastName = 'Russell'
```

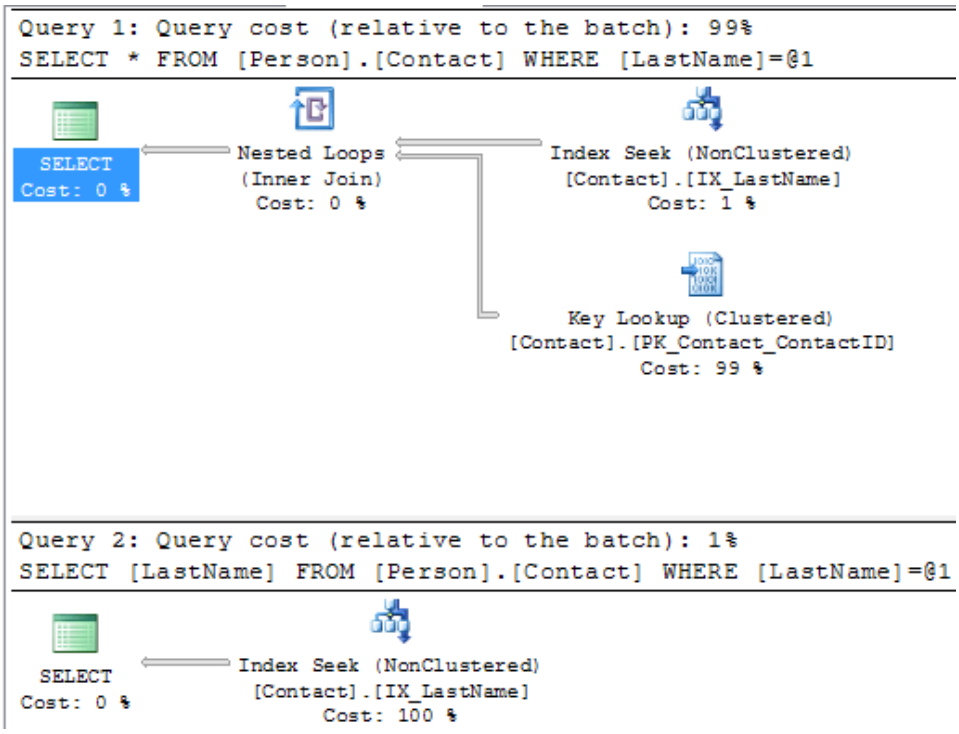
Here we can see that we no longer have a Key Lookup and we also no longer have the Nested Loops operator.



If we run both of these queries at the same time in one batch we can see the improvement by removing these two operators.

```
SELECT * FROM Person.Contact WHERE LastName = 'Russell'
SELECT LastName FROM Person.Contact WHERE LastName = 'Russell'
```

Below we can see that the first statement takes 99% of the batch and the second statement takes 1%, so this is a big improvement.

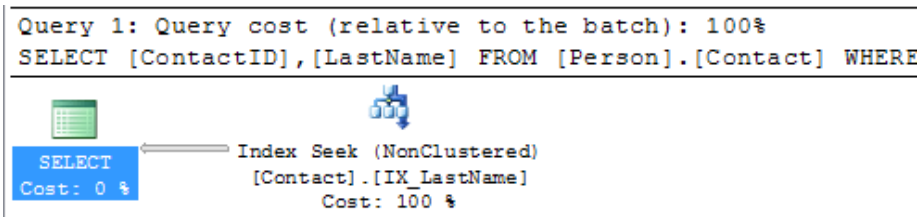


This should make sense that since the index includes LastName and that is the only column that is being used for both the SELECTed columns and the WHERE clause the index can handle the entire query. Another thing to be aware of is that if the table has a clustered index we can include the clustered index column or columns as well without doing a Key Lookup.

The Person.Contact table has a clustered index on ContactID, so if we include this column in the query we can still do just an Index Seek.

```
SELECT ContactID, LastName FROM Person.Contact WHERE LastName = 'Russell'
```

Here we can see that we only need to do an Index Seek to include both of these columns.



So that's great if that is all you need, but what if you need to include other columns such as FirstName. If we change the query as follows then the Key Lookup comes back again.

```
SELECT FirstName, LastName FROM Person.Contact WHERE LastName = 'Russell'
```

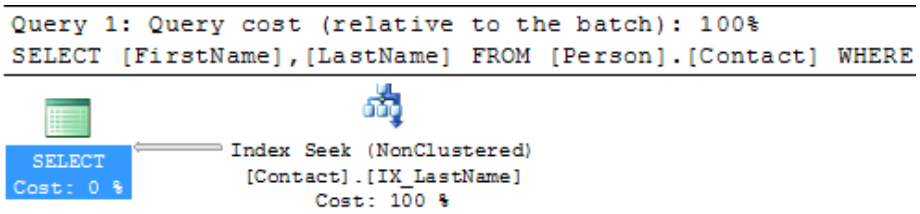
Luckily there are a few options to handle this.

Creating a Covering Index

A covering index basically does what it implies, it covers the query by including all of the columns that are needed. So if our need is to always include FirstName and LastName we can modify our index as follows to include both LastName and FirstName.

```
DROP INDEX [IX_LastName] ON [Person].[Contact]
GO
CREATE NONCLUSTERED INDEX [IX_LastName]
ON [Person].[Contact] ([LastName], [FirstName])
GO
```

And if we look at the execution plan we can see that we eliminated the Key Lookup once again.



Creating an Index with Included Columns

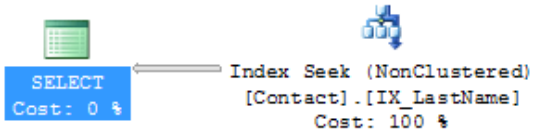
Another option is to use the included columns feature for an index. This allows you to include additional columns so they are stored with the index, but are not part of the index tree. So this allows you to take advantage of the features of a covering index and reduces storage needs within the index tree. Another benefit is that you can include additional data types that can not be part of a covering index.

The syntax for the the index with included columns is as follows:

```
DROP INDEX [IX_LastName] ON [Person].[Contact]
GO
CREATE NONCLUSTERED INDEX [IX_LastName]
ON [Person].[Contact] ([LastName])
INCLUDE ([FirstName])
GO
```

Here we can see the exuection plan is the same for both options.

```
Query 1: Query cost (relative to the batch): 100%
SELECT [FirstName],[LastName] FROM [Person].[Contact] WHERE
```



Additional Information

Here are some additional items related to the Key/RID Lookups.

- Improve SQL Server Performance with Covering Index Enhancements
- Understanding SQL Server Indexing
- SQL Server Indexing Basics

<< Previous Next >>

More SQL Server Solutions

Post a comment or let the author know this tip helped.

All comments are reviewed, so stay on subject or we may delete your comment. Note: your email address is not published. Required fields are marked with an asterisk (*).

*Name *Email Notify for updates

*** NOTE *** - If you want to include code from SQL Server Management Studio (SSMS) in your post, please copy the code from SSMS and paste the code into a text editor like NotePad before copying the code below to remove the SSMS formatting.

p

Send me SQL tips:

I'm not a robot

reCAPTCHA
Privacy - Terms

Save Comment

Thursday, May 19, 2016 - 5:40:31 AM - Munavvar Back To Top

Now I understand the concept of key/RID lookup, Thanks

Get Free SQL Tips

enter business email Join

Copyright (c) 2006-2018 Edgewood Solutions, LLC All rights reserved

Some names and products listed are the registered trademarks of their respective owners.