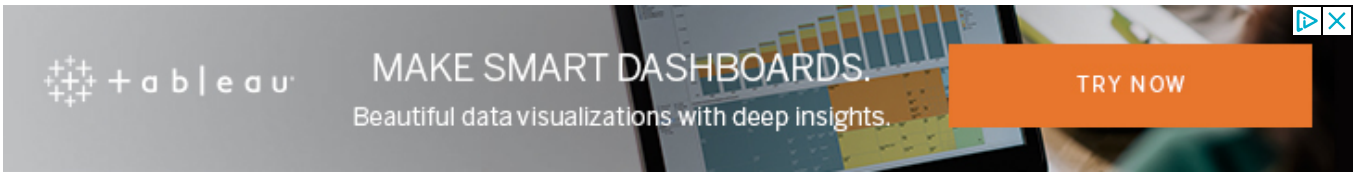
MSSQLTips Free SQL Server Conference Registration + Travel Expenses click to learn more

Index Scans and Table Scans



By: Greg Robidoux

Overview

There are several things that you can do to improve performance by throwing more hardware at the problem, but usually the place you get the most benefit from is when you tune your queries. One common problem that exists is the lack of indexes or incorrect indexes and therefore SQL Server has to process more data to find the records that meet the queries criteria. These issues are known as Index Scans and Table Scans.

In this section will look at how to find these issues and how to resolve them.

Explanation

An index scan or table scan is when SQL Server has to scan the data or index pages to find the appropriate records. A scan is the opposite of a seek, where a seek uses the index to pinpoint the records that are needed to satisfy the query. The reason you would want to find and fix your scans is because they generally require more I/O and also take longer to process. This is something you will notice with an application that

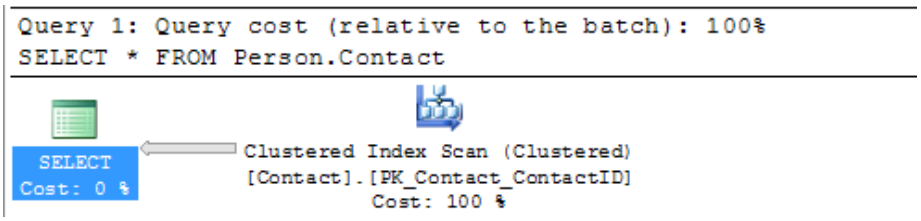
grows over time. When it is first released performance is great, but over time as more data is added the index scans take longer and longer to complete.

To find these issues you can start by running Profiler or setting up a server side trace and look for statements that have high read values. Once you have identified the statements then you can look at the query plan to see if there are scans occurring.

Here is a simple query that we can run. First use Ctrl+M to turn on the actual execution plan and then execute the query.

```
SELECT * FROM Person.Contact
```

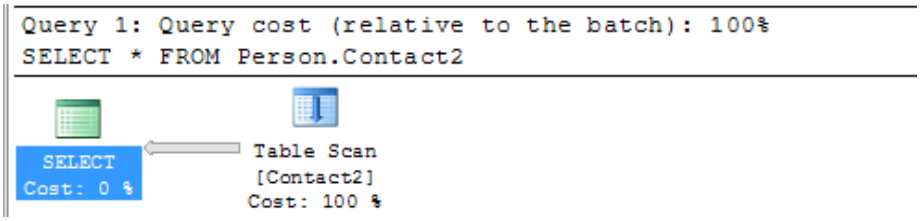
Here we can see that this query is doing a Clustered Index Scan. Since this table has a clustered index and there is not a WHERE clause SQL Server scans the entire clustered index to return all rows. So in this example there is nothing that can be done to improve this query.



In this next example I created a new copy of the Person.Contact table without a clustered index and then ran the query.

```
SELECT * FROM Person.Contact2
```

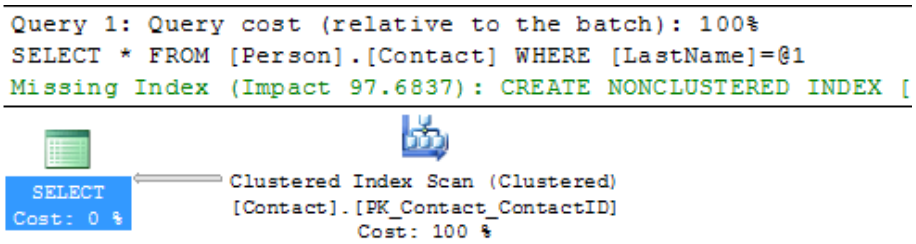
Here we can see that this query is doing a Table Scan, so when a table has a Clustered Index it will do a Clustered Index Scan and when the table does not have a clustered index it will do a Table Scan. Since this table does not have a clustered index and there is not a WHERE clause SQL Server scans the entire table to return all rows. So again in this example there is nothing that can be done to improve this query.



In this next example we include a WHERE clause for the query.

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

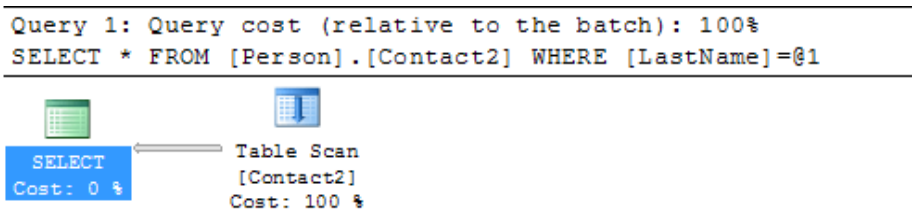
Here we can see that we still get the Clustered Index Scan, but this time SQL Server is letting us know there is a missing index. If you right click on the query plan and select **Missing Index Details** you will get a new window with a script to create the missing index.



Let's do the same thing for our Person.Contact2 table.

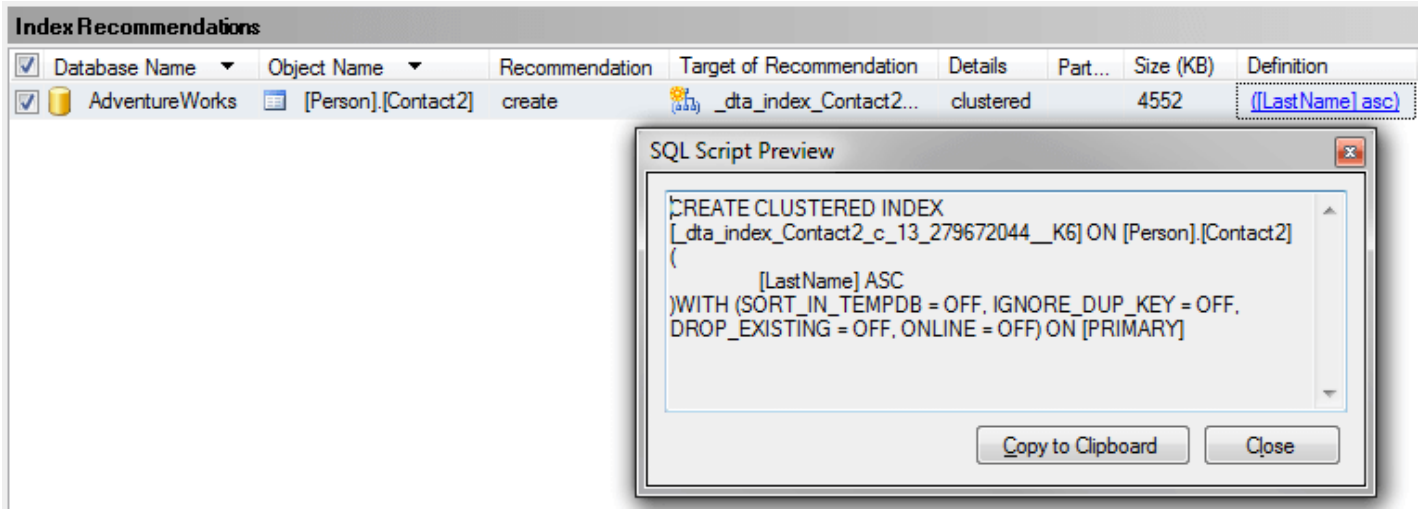
```
SELECT * FROM Person.Contact2 WHERE LastName = 'Russell'
```

We can see that we still have the Table Scan, but SQL Server doesn't offer any suggestions on how to fix this.



Another thing you could do is use the Database Engine Tuning Advisor to see if it gives you any suggestions. If I select the query in SSMS, right click and select **Analyze Query in Database Engine Tuning Advisor** the tools starts up and I can select the options and start the analysis.

Below is the suggestion this tool provides and we can see that recommends creating a new index, so you can see that using both tools can be beneficial.



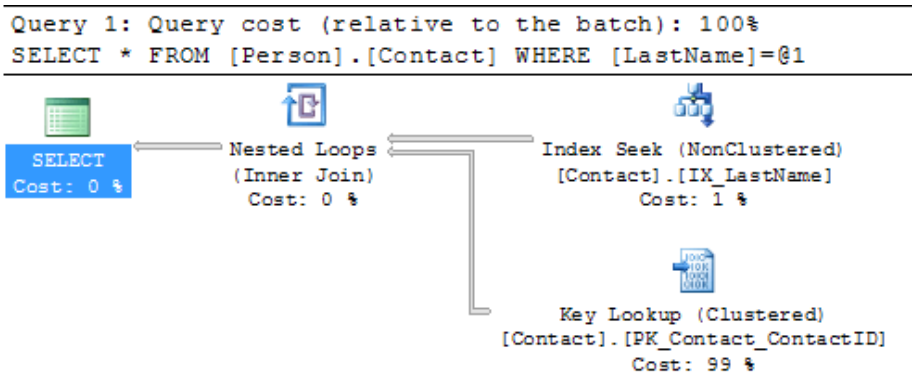
Create New Index

So let's create the recommended index on Person.Contact and run the query again.

```
USE [AdventureWorks]
GO
CREATE NONCLUSTERED INDEX [IX_LastName]
```

```
ON [Person].[Contact] ([LastName])
GO
SELECT * FROM Person.Contact WHERE LastName = 'Russell'
```

Here we can see the query plan has changed and instead of a Clustered Index Scan we now have an Index Seek which is much better. We can also see that there is now a Key Lookup operation which we will talk about in the next section.



Summary

By finding and fixing your Index Scans and Table Scans you can drastically improve performance especially for larger tables. So take the time to identify where your scans may be occurring and create the necessary indexes to solve the problem. One thing that you should be aware of is that too many indexes also causes issues, so make sure you keep a balance on how many indexes you create for a particular table.

Additional Information

Here are some additional items related to the Index Scans and Table Scans.

- Using Hints To Test SQL Server Indexes
- Avoid SQL Server functions in the WHERE clause for Performance
- Understanding SQL Server Indexing
- SQL Server Indexing Basics
- SQL Server Index Checklist
- Clustered Tables vs Heap Tables

<< 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

Save Comment

reCAPTCHA

Privacy - Terms

Thursday, February 15, 2018 - 9:53:47 AM - Apan Back To Top

Thanks for this article. You are amazing in explaining things, please keep posting.

Get Free SQL Tips

enter business email Join

Today's Tip

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

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