Index Seek vs Index Scan

The question comes up when looking at an execution plan of whats the difference between an index seek and an index scan.

What it really comes down to is speed, for tables with a large number of rows, and thus indexes with a large number of rows, an index seek may take significantly less time than an index scan. This is very hard to see the difference in small tables, the reason is that the difference is insignificant.

In the following scenarios the term page refers to the 8k pages that SQL server uses for memory allocation and/or data storage on disk. SQL Server Indexes are organized as a B-Tree of pages based on the columns specified in an index.

Index Scan

The term Index Scan refers to accessing every single page in the index to get the results that you are looking for. The table scan always accesses every page to check for the data that it is looking for. In some cases this may be faster than a full table scan, but not by much. Depending on the design of index and the size of the table, it could be more expensive that a full table scan.

Index Seek

Index Seek refers to process just walking through the B-Tree structure of the index to find the required data. B-Trees are very efficient and can get you to the data with just a few pages being walked through. For performance tuning, the Index Seek is where you really get the value out of your indexes.

Small Tables / Small Indexes

Think of a small table that has a few thousand rows, with a integer primary key used in an index, that index may take up a few pages, 3 to 10. With a index seek, it may require 2 pages to be accessed to find the rows that you are looking for. With an index scan it would require accessing all of the pages in the index to get to the rows, which in this case would be 3 to 10 pages, still not so bad.

Large Tables / Large Indexes

In a large table with a uniqueidentifier key that is indexed, the index could take up thousands or more pages. To get to the needed rows with a table seek, it may take reading 3 to 6 pages depending on the depth of the b-tree that the index is made up of. The table scan would require reading through every page in the index, in this case several thousand or more pages, which could take several thousand times as long to get to your data.

How do I adjust my query to get an index seek rather than an index scan?

It depends on the query and the indexes involved. Sometimes it might be simple by adding an additional comparison to the where clause which would allow the index to work better. Another thing that would throw off the query optimizer for Index Scan vs Index seek would be statistics (to be covered later), if SQL Server doesn't have up to date or accurate statistics for the index, then it may not do an Index Seek. Sometimes it may require adjusting or adding an index to get the index seek to work the way that you want.

Need help tracking down SQL Server performance issues like he Index Scan, take a look at the FREE application, Database Health Monitor to find your slow queries.