Understanding the Tipping Point

Kimberly L. Tripp
OWNER/PRESIDENT – SQLSKILLS.COM
@kimberlyltripp www.sqlskills.com/blogs/kimberly

Module Overview

What is the tipping point?

When is a query selective enough?

How the tipping point varies

Bookmark lookups are expensive

Are Bookmark Lookups Optimal?

At what range are bookmark lookups not optimal?

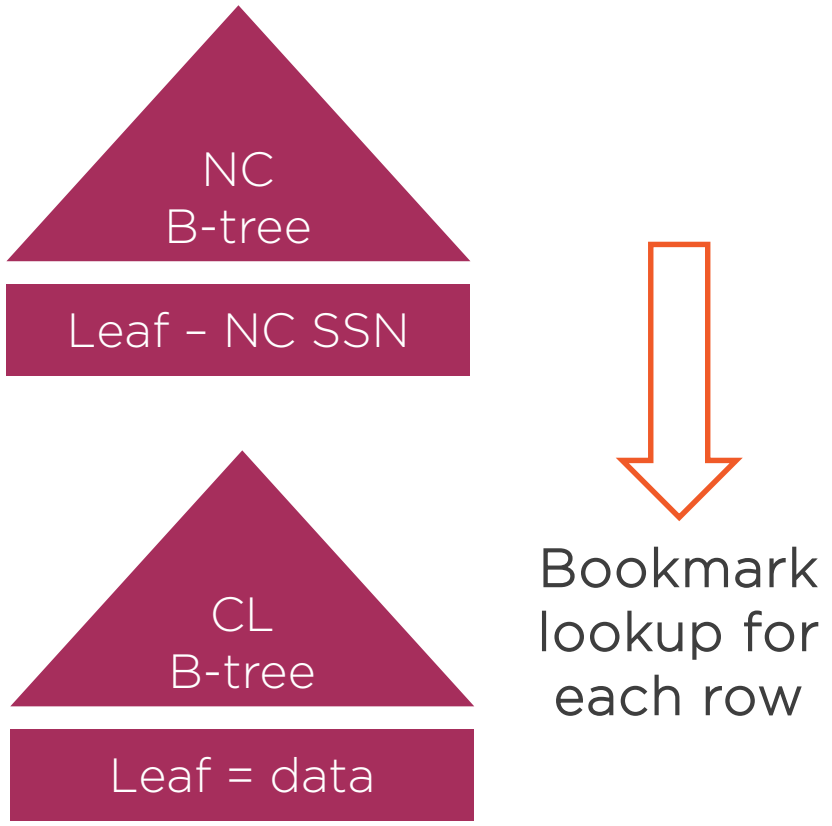
```
SELECT [e].*
FROM [dbo].[Employee] AS [e]
WHERE [e].[SSN] BETWEEN x AND y;
```

??? rows

Table scan (Employee) = 4,000 pages

When number of rows >= 4,000, table scan is definitely better!

In fact, probably well before that... as 4,000 sequential reads are better than even fewer random reads



The Tipping Point

If bookmark lookups are necessary... then a nonclustered is used ONLY when it's selective enough

SQL Server looks at table size to compare the cost of bookmark lookups (which are random) to cost of a table scan (which can be performed sequentially)

Estimating the tipping point:

- A nonclustered index can be used when the cost of the random I/O is low enough (under 1/4 of the number of pages in the table)
- A nonclustered index is NOT selective enough when the cost of the random I/O is too high (more than 1/3 of the number of pages in the table)

Employee Scenario

Imagine an Employee table with 80,000 rows at 20 rows/page = 4,000 total pages:

Marking these points ($^1/_4$ and $^1/_3$) we see the boundary where tipping point is found

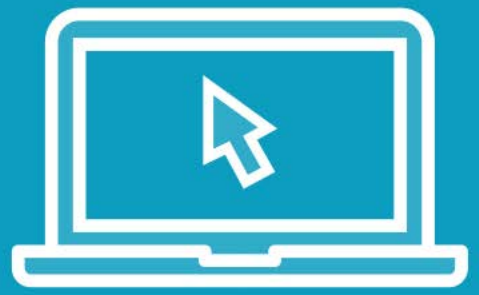
Since that represents random reads, translate that into rows (lookups):

$^1/_4$ mark is 1,000/80,000 = 1.25% and $^1/_3$ mark is 1,333/80,000 = 1.66%

If query against this table will do FEWER than 1,000 reads, SQL Server WILL use a nonclustered index because the lookups are inexpensive compared to a table scan

If query will do MORE than 1,333 reads, SQL Server will NOT use a nonclustered index because it's too expensive and instead, SQL Server will perform a table scan

Demo



The Tipping Point

The Tipping Point Varies...

Table by Table and Based on PAGES

Roughly **$^1/_4$ to $^1/_3$ of pages**

Why it is not an exact percentage?

- CPUs/cores, disk affinity...
- Not actual disk speed (e.g. solid state)
- Not based on what's in cache
- The query: data types, predicates, etc...

More info on my blog

- https://www.sqlskills.com/blogs/kimberly/category/the-tipping-point/

The Tipping Point Varies...

Table by Table and Based on PAGES

Tipping point query #1:

- Table with 1 million rows over 50,000 pages (= 20 rows/page)
  - 50,000 / 4 = 12,500 (1.25%)
  - 50,000 / 3 = 16,666 (1.66%)
- Queries requesting less than 1.25% of the data will use the nonclustered index (query ***IS*** selective enough)
- Queries requesting more than 1.66% will perform a table scan (query ***IS NOT*** selective enough)

The Tipping Point Varies...

Table by Table and Based on PAGES

Tipping point query #2:

- Table with 1 million rows over 10,000 pages (= 100 rows/page)
  - 10,000 / 4 = 2,500 (0.25%)
  - 10,000 / 3 = 3,333 (0.33%)
- Queries requesting less than 0.25% of the data will use the nonclustered index (query ***IS*** selective enough)
- Queries requesting more than 0.33% will perform a table scan (query ***IS NOT*** selective enough)

The Tipping Point Varies...

Table by Table and Based on PAGES

Tipping point query #3:

- Table with 1 million rows over 100,000 pages (= 10 rows/page)
  - 100,000 / 4 = 25,000 (2.5%)
  - 100,000 / 3 = 33,333 (3.33%)
- Queries requesting less than 2.5% of the data will use the nonclustered index (query ***IS*** selective enough)
- Queries requesting more than 3.33% will perform a table scan (query ***IS NOT*** selective enough)

Generalizing the Tipping Point

	TABLE			Selective-Enough		Scan	
Scenario	Rows	Pages	Row/Page	1/4	Percentage	1/3	Percentage
Scenario 3	1,000,000	100,000	10	25,000	2.500%	33,333.33	3.333%
Employee	80,000	4,000	20	1,000	1.250%	1,333.33	1.667%
EmployeeHeap	80,000	4,000	20	1,000	1.250%	1,333.33	1.667%
Scenario 1	1,000,000	50,000	20	12,500	1.250%	16,666.67	1.667%
Scenario 2	1,000,000	10,000	100	2,500	0.250%	3,333.33	0.333%

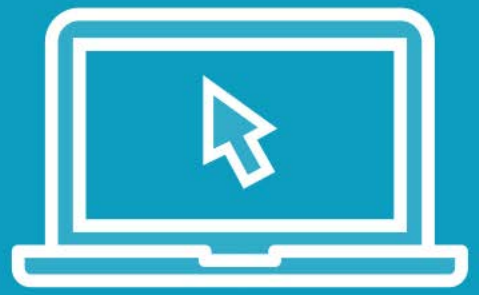
Generalizing the Tipping Point

	TABLE			Selective-Enough		Scan	
Scenario	Rows	Pages	Row/Page	1/4	Percentage	1/3	Percentage
Scenario 3	1,000,000	100,000	10	25,000	2.500%	33,333.33	3.333%
Employee	80,000	4,000	20	1,000	1.250%	1,333.33	1.667%
EmployeeHeap	80,000	4,000	20	1,000	1.250%	1,333.33	1.667%
Scenario 1	1,000,000	50,000	20	12,500	1.250%	16,666.67	1.667%
Scenario 2	1,000,000	10,000	100	2,500	0.250%	3,333.33	0.333%

Generalizing the Tipping Point

	TABLE			Selective-Enough		Scan	
Scenario	Rows	Pages	Row/Page	1/4	Percentage	1/3	Percentage
Scenario 3	1,000,000	100,000	10	25,000	2.500%	33,333.33	3.333%
Employee	80,000	4,000	20	1,000	1.250%	1,333.33	1.667%
EmployeeHeap	80,000	4,000	20	1,000	1.250%	1,333.33	1.667%
Scenario 1	1,000,000	50,000	20	12,500	1.250%	16,666.67	1.667%
Scenario 2	1,000,000	10,000	100	2,500	0.250%	3,333.33	0.333%

Demo



"That doesn't happen on big tables..."

Summary: The Tipping Point

Use formula as an estimate – more concept than hard number

- It's not exact but the idea is incredibly important!

Narrow indexes have very FEW uses! (they often tip)

If query has to do a bookmark lookup then it must be EXTREMELY HIGHLY selective in order to use the index

If query is not highly selective but critical and you want consistent results/plans, you must consider other options for optimization

- Adding an index solely because a column is listed in a WHERE clause is NOT enough to make that index useful

Covering? Covering is always good – there is NO TIPPING POINT!

Summary: Lookups Are Expensive!

How can we get rid of the bookmark lookup?

Joins that reduce set before lookups are needed

Index intersection (using multiple indexes)

Covering, in its many forms

You have to be careful not to over-index!

What We Covered

- What is the tipping point?
- When is a query selective enough?
- How the tipping point varies
- Bookmark lookups are expensive