Covering Queries

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

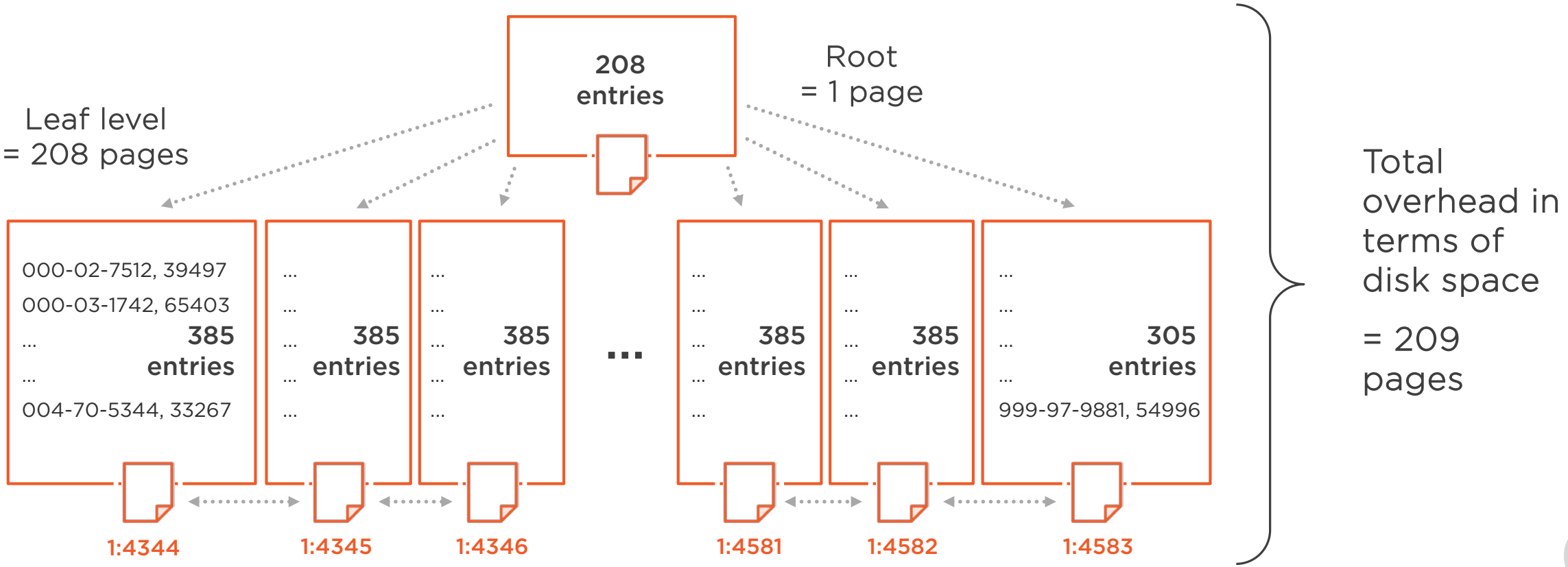
Module Overview

- Nonclustered index seeks and scans
- Selectivity determines access pattern
- Understanding covering
- How is covering possible
- Improving low-selectivity queries
- Covering cleverly, correctly, and concisely!

Nonclustered Index: Unique Key SSN

Think back to our nonclustered index structure

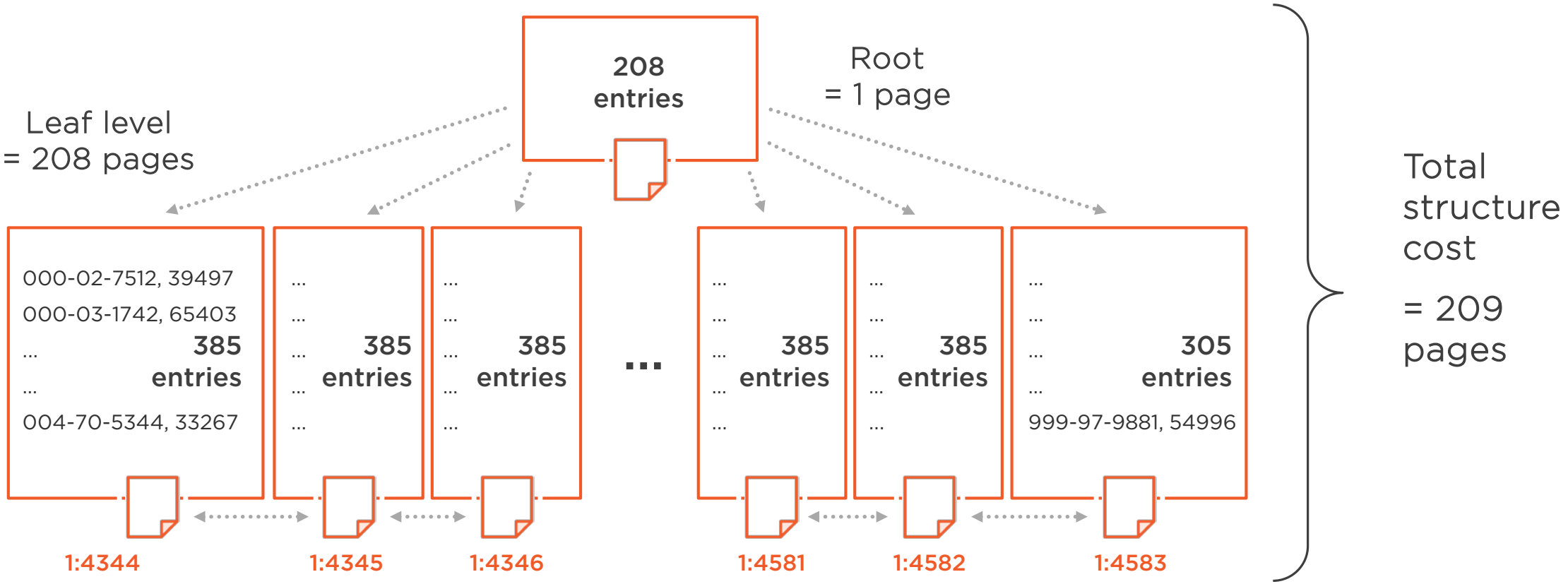
Leaf level contains nonclustered key column(s), in indexed order



What if You Didn't Know?

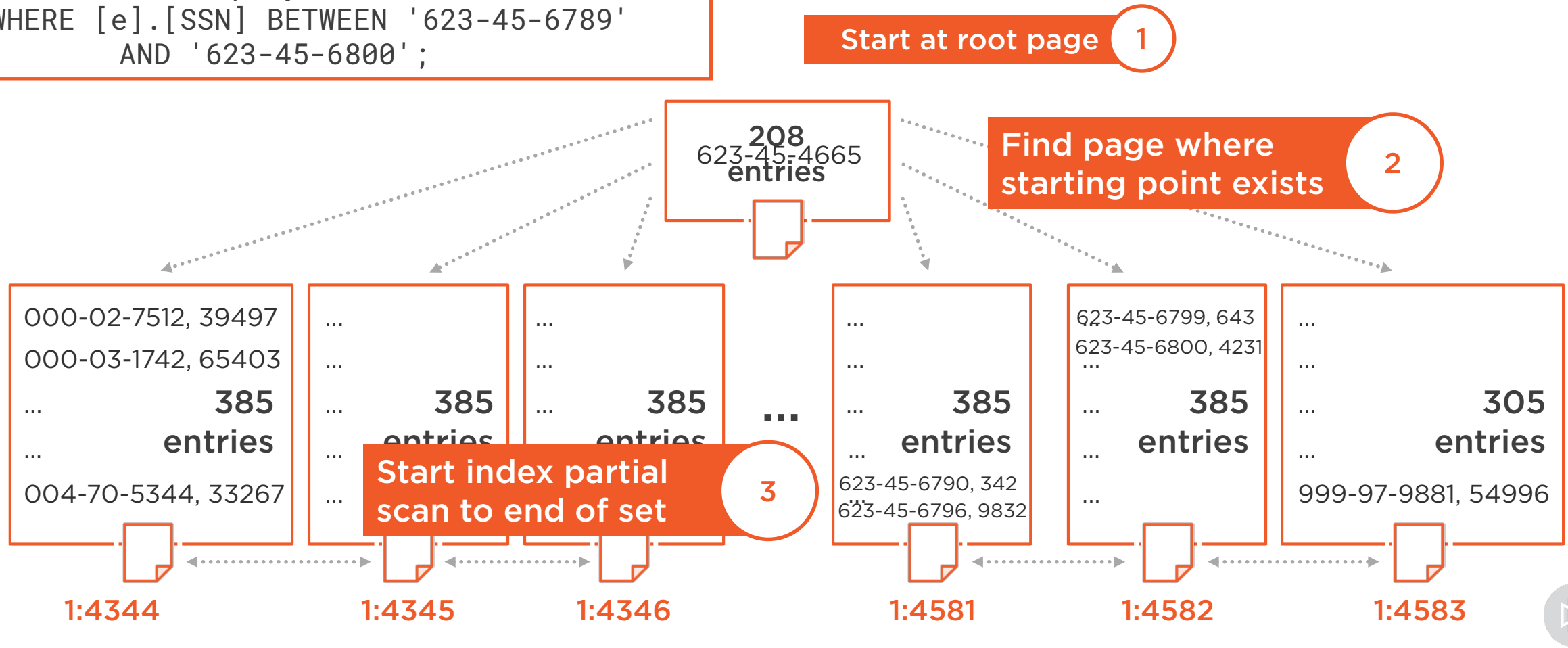
Could this structure be anything else?

What about table with only two columns (EmployeeID and SSN) that's clustered on SSN?

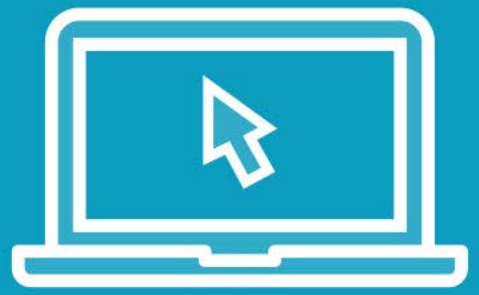


Fairly Obvious Index Access

```
SELECT [e].[EmployeeID], [e].[SSN]
FROM [dbo].[Employee] AS [e]
WHERE [e].[SSN] BETWEEN '623-45-6789'
        AND '623-45-6800';
```



Demo



Nonclustered, covering, seek

Less Obvious Index Access

```
SELECT [e].[EmployeeID], [e].[SSN]
FROM [dbo].[Employee] AS [e]
WHERE [e].[EmployeeID] < 10000;
```

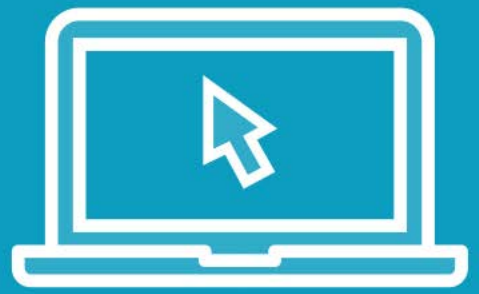
Clustered index on EmployeeID = ~500 reads

- Seekable with partial scan
- Table has 80,000 rows at 20 rows per page, so 4,000 pages
- 10,000 is $^1/_8$ of 80,000, so this SEEKABLE query will cost $^1/_8$ of 4,000 pages

Nonclustered index on SSN, EmployeeID = ~208 reads

- Not seekable, must scan
- Leaf level has 80,000 rows at 385 rows per page, so 208 pages
- If index is not seekable, then SQL Server must scan all 208 pages

Demo



Nonclustered, covering, scan

The Best Index Varies…

What if number of rows WHERE EmployeeID < 10000 was not 9,999?

Would optimizer make a different choice between the two indexes?

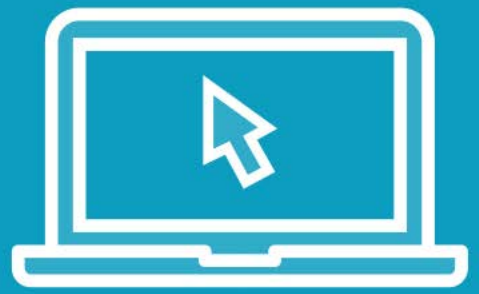
- When 9,999 rows:
  - Seekable clustered index = ~500 reads
  - Nonclustered covering index scan = ~208 reads
- If only 1,000 rows, for example:
  - Seekable clustered index = ~50 reads
  - Nonclustered covering index scan = ~208 reads

Because the covering index isn't seekable, the best index varies!

If query is critical, consider covering with a nonclustered index that is seekable:

```
CREATE INDEX [NCCoveringSeekableIX]
ON [dbo].[Employee] ([EmployeeID], [SSN])
```

Demo



Which index is best varies

What Is Covering?

Only applies to nonclustered indexes

- Clustered “covers” all requests but it’s what we’re trying to avoid!

Using JUST a nonclustered index to access the data a query requests

All columns requested in query are somewhere in index regardless of:

- Where they are in the query
- Where they are in the index

However, column order does matter…

- If an index is seekable, might be able to drastically reduce reads
- If an index is not seekable and can only be scanned, still might have significant savings, depending on how much narrower index is than base table…

How Is Covering Possible?

Unfiltered NC indexes contain a row for every row in base table

Each NC is a “mini clustered table” of just the data you need!

This structure looks and acts just like a table

We can seek into a NC index; we can scan a NC index

We can AVOID bookmark lookups!

There are all sorts of ways we can leverage this!

Improve Critical Queries with Low Selectivity

```
SELECT [e].[LastName], [e].[FirstName], [e].[Phone]
FROM [dbo].[Employee] AS [e]
WHERE [e].[LastName] LIKE '[S-Z]%';
  -- 13,632 in S-Z range (NOT selective)
```

Options to Access Data

Table scan

Nonclustered on LastName with bookmark lookups for every row

Nonclustered index: LastName, FirstName, PhoneNo

Nonclustered index: FirstName, LastName, PhoneNo

You CANNOT Cover Everything!

Theoretically you can cover anything and everything

- *Just because you can, doesn't mean you should!*

Over-indexing can be worse than under-indexing... as unused indexes are nothing but costly overhead

- *Do not just automatically put "an index on every column"!!*

Too many indexes cost you:

- During data modifications
- During regular index maintenance
- Wasted space in memory
- Wasted space on disk (including in backups)

Covering: Cleverly, Correctly, and Concisely!

Scalability != “add more indexes”

Get a good feel for your workload

Know your data and workload, plus how SQL Server works

Test a wide variety of different values and scenarios

Using covering sparingly

Use index consolidation techniques

Methods for Covering

Nonclustered indexes

Using INCLUDE

Using filtered indexes

Using indexed views (more for relational DW)

Using columnstore indexes (more for relational DW)

What We Covered

- Nonclustered index seeks and scans
- Selectivity determines access pattern
- Understanding covering
- How is covering possible
- Improving low-selectivity queries
- Covering cleverly, correctly, and concisely!