Nonclustered Index Internals

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

Module Overview

Nonclustered index overview

Case study

- Building the leaf level
- Building the tree structure

Purpose of clustering key columns in nonclustered indexes

Nonclustered Index Overview

Not required, but critical to achieving best performance

Maximum of 999 per table since SQL Server 2008

Separate from base table, with links to base table rows

Logical order maintained through a doubly-linked list

Requires ongoing and automated maintenance

Fastest index for range queries if query covered!

Employee Table Case Study

Nonclustered unique constraint for SSN

- Investigate nonclustered index leaf rows
- Physically order leaf level
- Investigate nonclustered index non-leaf level rows
- Complete the B-tree

Investigate Nonclustered Leaf Rows

Step 1: Build a new structure for nonclustered index leaf rows

Leaf-level row = nonclustered index column(s) + data row lookup ID + row overhead

Data row lookup ID = fixed RID (if heap) or clustering key

Row overhead = TagA byte + null block (min of 3) + additional overhead as needed

= TagA (1 byte) + SSN (11 bytes) + EmployeeID (4 bytes) + null block (3 bytes)

= 19 bytes/entry + 2 bytes in the slot array

$$\frac{\text{8,096 bytes / page}}{\text{19 bytes/entry + 2 bytes in slot array}} = \textbf{385 index entries per leaf level page}$$

$$\frac{\text{80,000 rows}}{\text{385 rows per page}} = \textbf{208 leaf level pages}$$

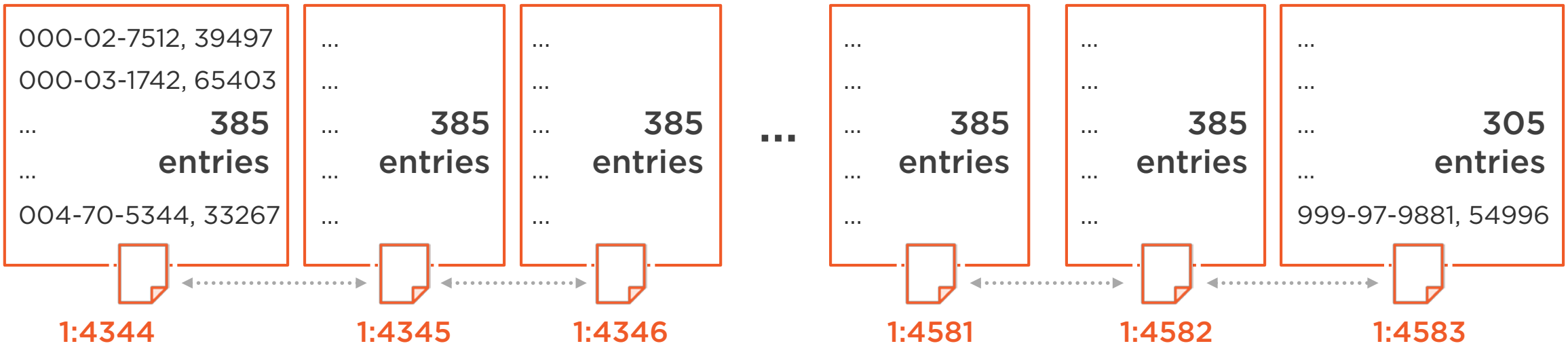
Nonclustered Unique Constraint on SSN

Step 1: **Physically order nonclustered leaf level**

SQL Server will duplicate SSN and EmployeeID for EVERY ROW and order it by index definition (ascending by default)

Note: every INSERT/DELETE will need to touch each nonclustered index to keep them up-to-date

80,000 x SSN, EmployeeID Pairs = 208 Pages



Investigate Nonclustered Non-leaf Rows

Step 2: **Complete the B-tree**

Non-leaf level row = nonclustered index column(s) + data row lookup ID* + pointer + row overhead

*Data row lookup ID only included when nonclustered index is nonunique

Row overhead = TagA byte + null block (min of 3, but only when nullable columns)

= TagA (1 byte) + SSN (11 bytes) + pointer (6 bytes)

= 18 bytes/entry + 2 bytes in the slot array

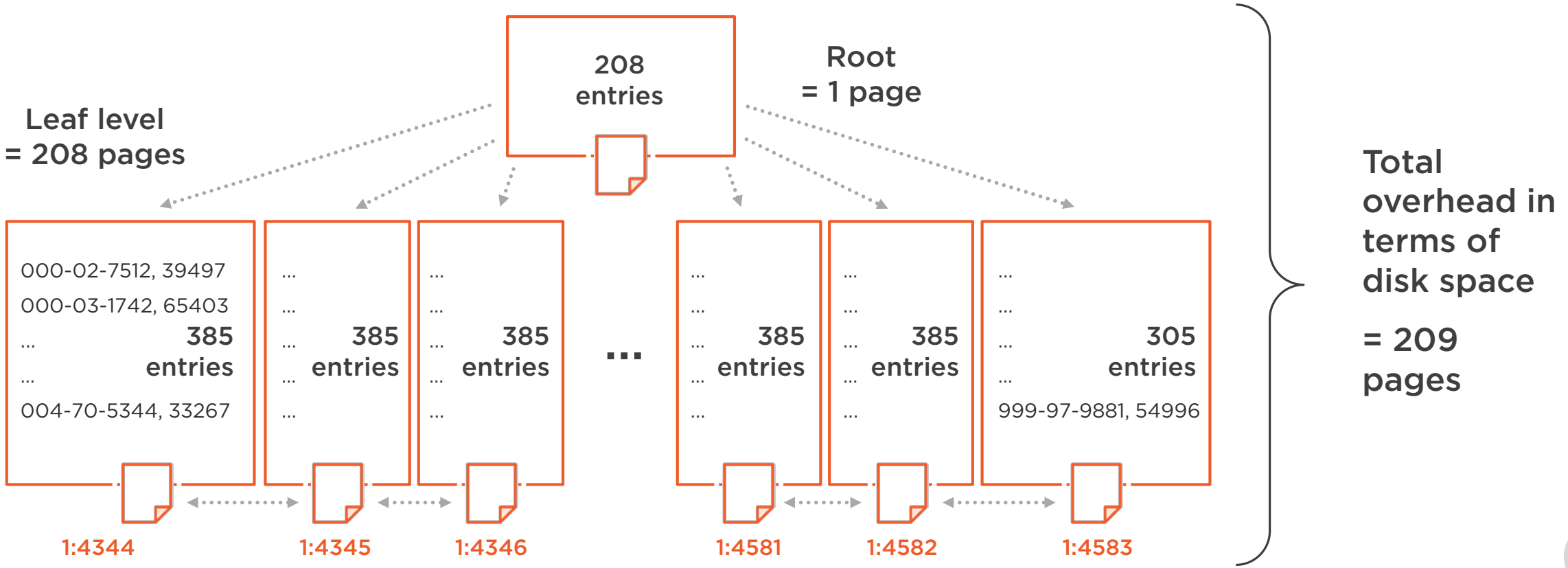
$$\frac{\text{8,096 bytes / page}}{\text{18 bytes/entry + 2 bytes in slot array}} = \text{404 index entries per non-leaf level page}$$

$$\frac{\text{208 rows}}{\text{404 rows per page}} = \text{1 non-leaf level page = root page}$$

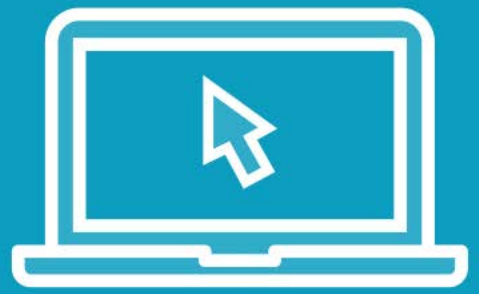
Nonclustered Unique Constraint on SSN

Step 2: **Complete the B-tree**

Continuing up to a root of 1 page

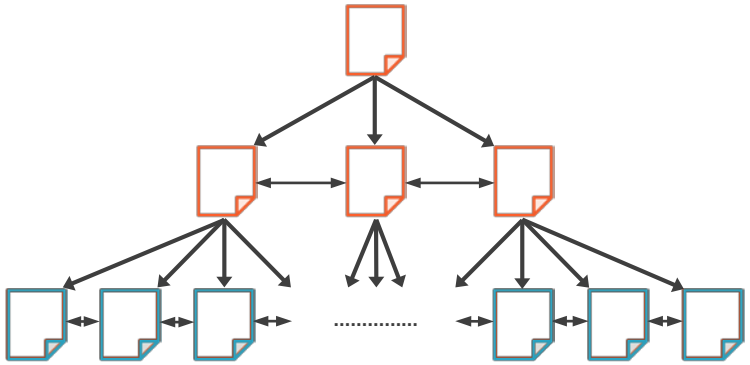


Demo



Nonclustered index internals

Lookup ID in Nonclustered Indexes



Always in the leaf level of the index

Only added up-the-tree when the nonclustered index is non-unique

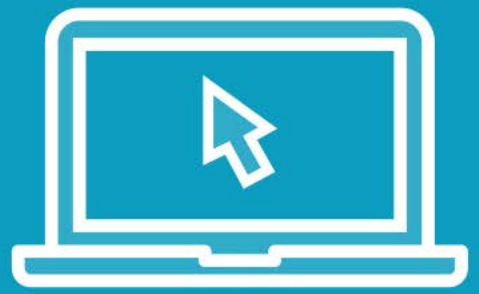
- Nonclustered indexes require the "lookup" key in the b-tree when? (http://bit.ly/2rX9i5u)

For a clustered table, the lookup ID is the clustering key

For a heap, the lookup ID is the 8-byte RID

- 2 bytes for the File ID
- 4 bytes for the Page ID
- 2 bytes for the Slot ID

Demo



Clustering key columns in nonclustered indexes

Clustering Key Columns Where?

What if:

```
CREATE UNIQUE CLUSTERED INDEX [IXCL] ON [tname] ([c6], [c8], [c2]);
CREATE NONCLUSTERED INDEX [NC1] ON [tname] ([c5], [c2], [c4]);
```

Nonclustered index leaf level: c5, c2, c4, c6, c8

Nonclustered index key and non-leaf levels: c5, c2, c4, c6, c8

What if:

```
CREATE UNIQUE NONCLUSTERED INDEX [NC2] ON [tname] ([c5], [c2], [c4]);
```

Nonclustered index leaf level: c5, c2, c4, c6, c8

Nonclustered index key and non-leaf levels: c5, c2, c4

Clustering key columns are added only ONCE to your nonclustered indexes

Where they are added is based on whether nonclustered is unique or not

What We Covered

Nonclustered index overview

Case study

- Building the leaf level
- Building the tree structure

Purpose of clustering key columns in nonclustered indexes