Article - Microsoft SQL Server Clustered Index Design

This is one of the most interesting documentation pages I have read and I really thought I would share it with you. You can read the entire design guide with this link and I also included the pdf of the design. I took a snippet of the two most important parts that I found interesting from this design guide, pay close attention to the images and how clustered index is persisted compared to non-clustered index in the b-tree architecture. Keep in mind that this is Microsoft way of solving things and it's not necessarily the way to go. Try to challenge how is this done and come up with a different approach.

Enjoy
-Hussein

Clustered Index Architecture

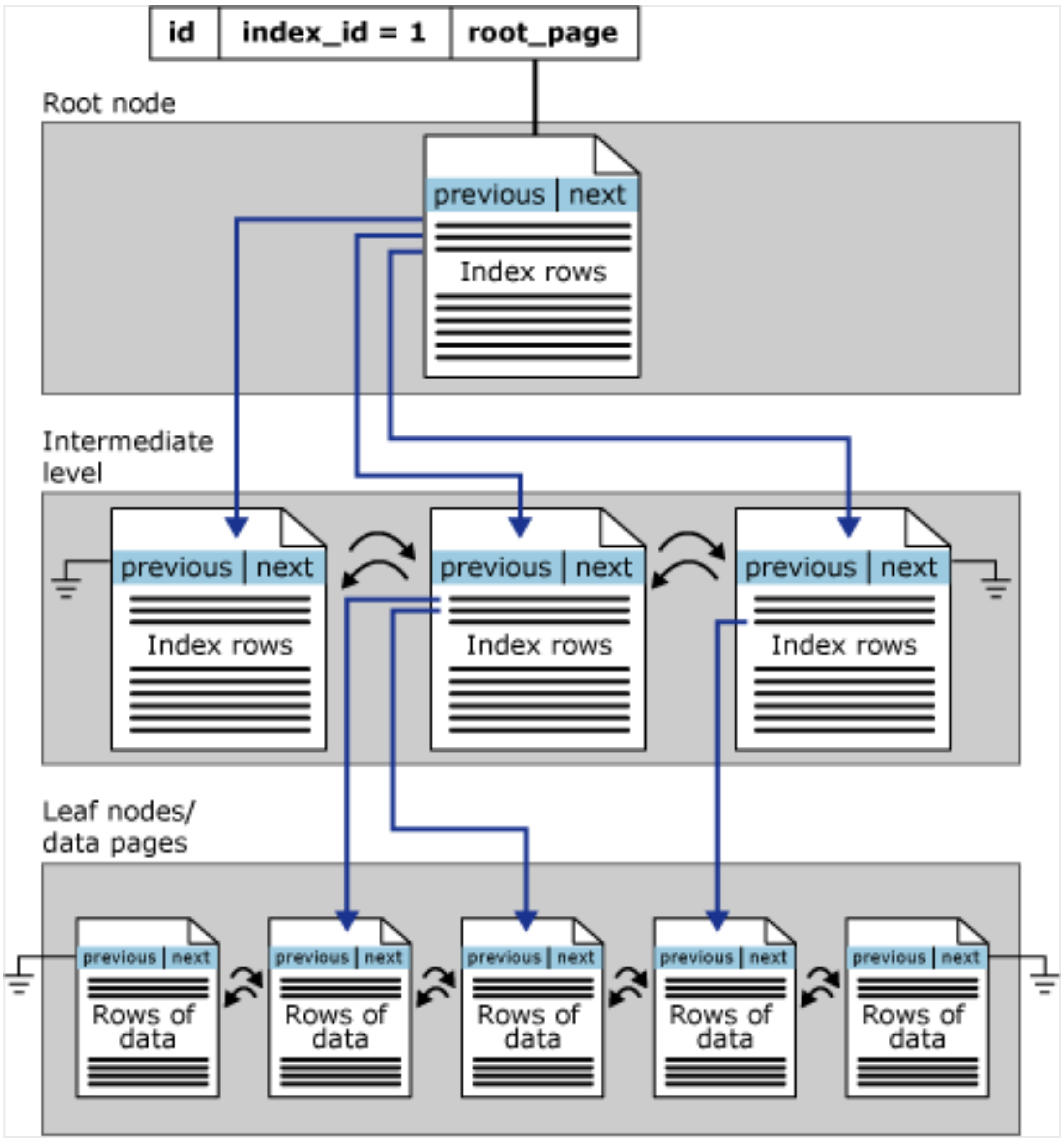
In SQL Server, indexes are organized as B-Trees. Each page in an index B-tree is called an index node. The top node of the B-tree is called the root node. The bottom nodes in the index are called the leaf nodes. Any index levels between the root and the leaf nodes are collectively known as intermediate levels. In a clustered index, the leaf nodes contain the data pages of the underlying table. The root and intermediate level nodes contain index pages holding index rows. Each index row contains a key value and a pointer to either an intermediate level page in the B-tree, or a data row in the leaf level of the index. The pages in each level of the index are linked in a doubly-linked list.

Clustered indexes have one row in sys.partitions, with **index_id** = 1 for each partition used by the index. By default, a clustered index has a single partition. When a clustered index has multiple partitions, each partition has a B-tree structure that contains the data for that specific partition. For example, if a clustered index has four partitions, there are four B-tree structures; one in each partition.

Depending on the data types in the clustered index, each clustered index structure will have one or more allocation units in which to store and manage the data for a specific partition. At a minimum, each clustered index will have one IN_ROW_DATA allocation unit per partition. The clustered index will also have one *LOB_DATA* allocation unit per partition if it contains large object (LOB) columns. It will also have one *ROW_OVERFLOW_DATA* allocation unit per partition if it contains variable length columns that exceed the 8,060 byte row size limit.

The pages in the data chain and the rows in them are ordered on the value of the clustered index key. All inserts are made at the point where the key value in the inserted row fits in the ordering sequence among existing rows.

This illustration shows the structure of a clustered index in a single partition.



Nonclustered Index Architecture

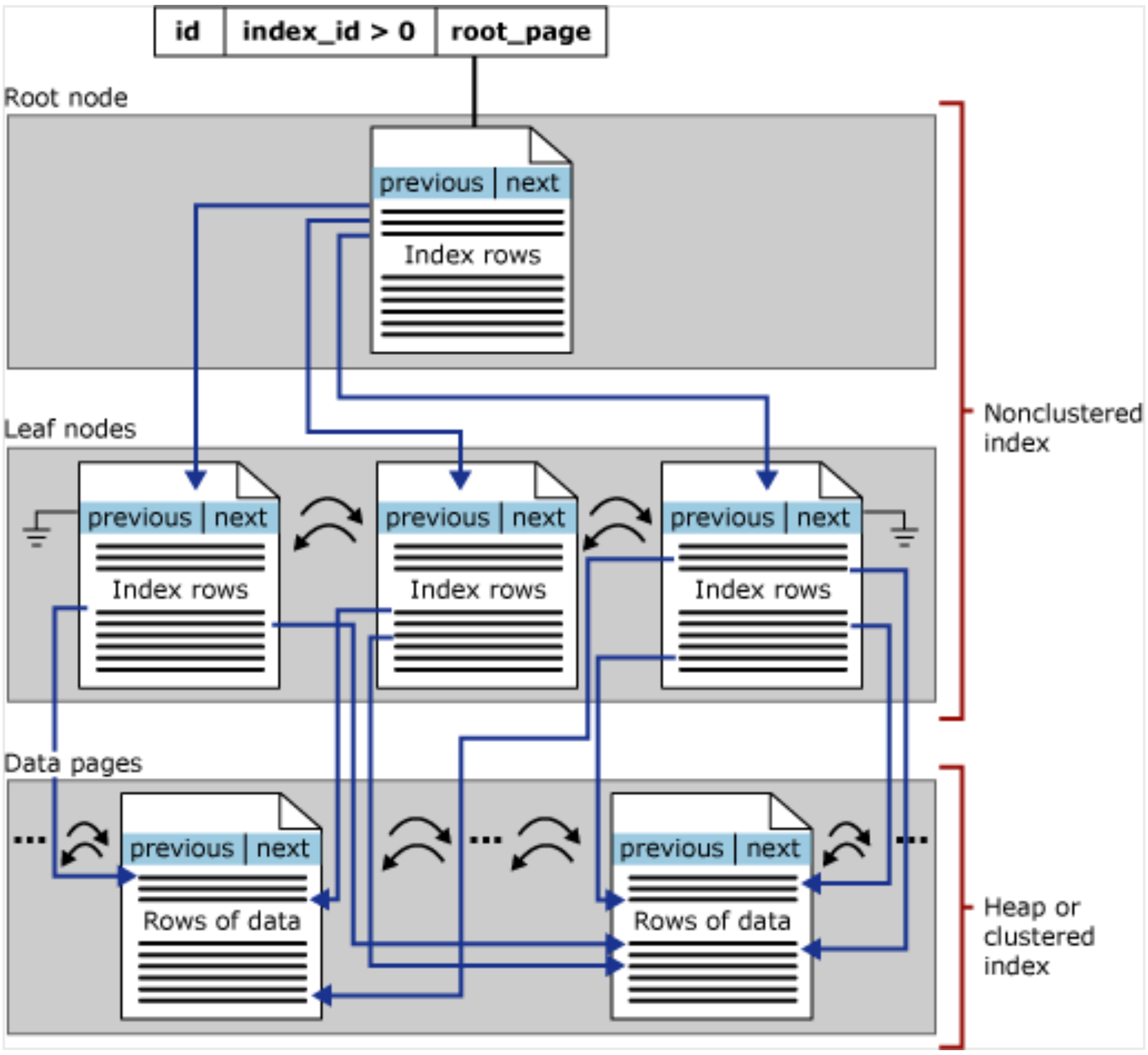
Nonclustered indexes have the same B-tree structure as clustered indexes, except for the following significant differences:

- The data rows of the underlying table are not sorted and stored in order based on their nonclustered keys.
- The leaf level of a nonclustered index is made up of index pages instead of data pages.

The row locators in nonclustered index rows are either a pointer to a row or are a clustered index key for a row, as described in the following:

- If the table is a heap, which means it does not have a clustered index, the row locator is a pointer to the row. The pointer is built from the file identifier (ID), page number, and number of the row on the page. The whole pointer is known as a Row ID (RID).
- If the table has a clustered index, or the index is on an indexed view, the row locator is the clustered index key for the row.

Nonclustered indexes have one row in sys.partitions with **index_id** > 1 for each partition used by the index. By default, a nonclustered index has a single partition. When a nonclustered index has multiple partitions, each partition has a B-tree structure that contains the index rows for that specific partition. For example, if a nonclustered index has four partitions, there are four B-tree structures, with one in each partition.
Depending on the data types in the nonclustered index, each nonclustered index structure will have one or more allocation units in which to store and manage the data for a specific partition. At a minimum, each nonclustered index will have one *IN_ROW_DATA* allocation unit per partition that stores the index B-tree pages. The nonclustered index will also have one *LOB_DATA* allocation unit per partition if it contains large object (LOB) columns. Additionally, it will have one *ROW_OVERFLOW_DATA* allocation unit per partition if it contains variable length columns that exceed the 8,060 byte row size limit.
The following illustration shows the structure of a nonclustered index in a single partition.


id
index_id > 0
root_page
Root node
previous | next
Index rows
Leaf nodes
Nonclustered index
Data pages
Rows of data
Heap or clustered index