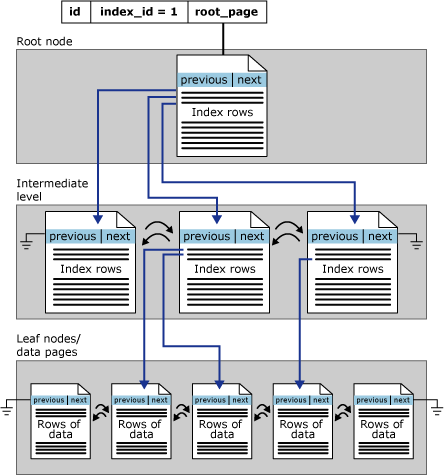
For a clustered index, the **root\_page** column in **sys.system\_internals\_allocation\_units** points to the top of the clustered index for a specific partition. SQL Server moves down the index to find the row corresponding to a clustered index key. To find a range of keys, SQL Server moves through the index to find the starting key value in the range and then scans through the data pages using the previous or next pointers. To find the first page in the chain of data pages, SQL Server follows the leftmost pointers from the root node of the index.

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



Nonclustered indexes have one row in sys.partitions with **index\_id** >0 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. For more information about allocation units, see Table and Index Organization. The page collections for the B-tree are anchored by **root\_page** pointers in the **sys.system\_internals\_allocation\_units** system view.

The following illustration shows the structure of a nonclustered index in a single partition.

