**Nonclustered Index Structures**

SQL Server 2008 Books Online (November 2009)

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 layer of a nonclustered index is made up of index pages instead of data pages.

Nonclustered indexes can be defined on a table or view with a clustered index or a heap. Each index row in the nonclustered index contains the nonclustered key value and a row locator. This locator points to the data row in the clustered index or heap having the key value.

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:

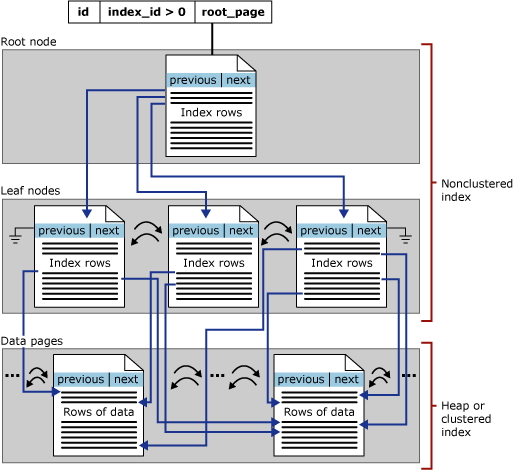
* If the table is a heap, which means it does not have a clustered index, the row locator is a pointer to the row. The pointer is built from the file identifier (ID), page number, and number of the row on the page. The whole pointer is known as a Row ID (RID).
* If the table has a clustered index, or the index is on an indexed view, the row locator is the clustered index key for the row. If the clustered index is not a unique index, SQL Server makes any duplicate keys unique by adding an internally generated value called a **uniqueifier**. This four-byte value is not visible to users. It is only added when required to make the clustered key unique for use in nonclustered indexes. SQL Server retrieves the data row by searching the clustered index using the clustered index key stored in the leaf row of the nonclustered index.

Nonclustered indexes have one row in [sys.partitions](http://msdn.microsoft.com/en-us/library/ms175012.aspx) 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](http://msdn.microsoft.com/en-us/library/ms189051.aspx). The page collections for the B-tree are anchored by **root\_page** pointers in the **sys.system\_internals\_allocation\_units** system view.

|  |
| --- |
| **ms177484.note(en-us,SQL.100).gifImportant:** |
| The **sys.system\_internals\_allocation\_units** system view is reserved for Microsoft SQL Server internal use only. Future compatibility is not guaranteed. |

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



http://i.msdn.microsoft.com/Global/Images/clear.gif Included Column Indexes

The functionality of nonclustered indexes can be extended by adding included columns, called nonkey columns, to the leaf level of the index. While the key columns are stored at all levels of the nonclustered index, nonkey columns are stored only at the leaf level. For more information, see [Index with Included Columns](http://msdn.microsoft.com/en-us/library/ms190806.aspx).