**Clustered and Nonclustered Indexes Described**

**SQL Server 2014**

[Other Versions](javascript:;)

http://i.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png?v=635533092500576233

* [SQL Server 2012](http://msdn.microsoft.com/en-us/library/ms190457(d=printer,v=sql.110).aspx)
* [SQL Server 2008 R2](http://msdn.microsoft.com/en-us/library/ms190457(d=printer,v=sql.105).aspx)
* [SQL Server 2008](http://msdn.microsoft.com/en-us/library/ms190457(d=printer,v=sql.100).aspx)
* [SQL Server 2005](http://msdn.microsoft.com/en-us/library/ms190457(d=printer,v=sql.90).aspx)

An index is an on-disk structure associated with a table or view that speeds retrieval of rows from the table or view. An index contains keys built from one or more columns in the table or view. These keys are stored in a structure (B-tree) that enables SQL Server to find the row or rows associated with the key values quickly and efficiently.

A table or view can contain the following types of indexes:

* Clustered
  + Clustered indexes sort and store the data rows in the table or view based on their key values. These are the columns included in the index definition. There can be only one clustered index per table, because the data rows themselves can be sorted in only one order.
  + The only time the data rows in a table are stored in sorted order is when the table contains a clustered index. When a table has a clustered index, the table is called a clustered table. If a table has no clustered index, its data rows are stored in an unordered structure called a heap.
* Nonclustered
  + Nonclustered indexes have a structure separate from the data rows. A nonclustered index contains the nonclustered index key values and each key value entry has a pointer to the data row that contains the key value.
  + The pointer from an index row in a nonclustered index to a data row is called a row locator. The structure of the row locator depends on whether the data pages are stored in a heap or a clustered table. For a heap, a row locator is a pointer to the row. For a clustered table, the row locator is the clustered index key.
  + You can add nonkey columns to the leaf level of the nonclustered index to by-pass existing index key limits, 900 bytes and 16 key columns, and execute fully covered, indexed, queries. For more information, see [Create Indexes with Included Columns](http://msdn.microsoft.com/en-us/library/ms190806.aspx).

Both clustered and nonclustered indexes can be unique. This means no two rows can have the same value for the index key. Otherwise, the index is not unique and multiple rows can share the same key value. For more information, see [Create Unique Indexes](http://msdn.microsoft.com/en-us/library/ms187019.aspx).

Indexes are automatically maintained for a table or view whenever the table data is modified.

See [Indexes](http://msdn.microsoft.com/en-us/library/ms175049.aspx) for additional types of special purpose indexes.

[**Indexes and Constraints**](javascript:void(0))

Indexes are automatically created when PRIMARY KEY and UNIQUE constraints are defined on table columns. For example, when you create a table and identify a particular column to be the primary key, the Database Engine automatically creates a PRIMARY KEY constraint and index on that column. For more information, see [Create Primary Keys](http://msdn.microsoft.com/en-us/library/ms189039.aspx) and [Create Unique Constraints](http://msdn.microsoft.com/en-us/library/ms190024.aspx).

[**How Indexes Are Used by the Query Optimizer**](javascript:void(0))

Well-designed indexes can reduce disk I/O operations and consume fewer system resources therefore improving query performance. Indexes can be helpful for a variety of queries that contain SELECT, UPDATE, DELETE, or MERGE statements. Consider the query SELECT Title, HireDate FROM HumanResources.Employee WHERE EmployeeID = 250 in the AdventureWorks2012 database. When this query is executed, the query optimizer evaluates each available method for retrieving the data and selects the most efficient method. The method may be a table scan, or may be scanning one or more indexes if they exist.

When performing a table scan, the query optimizer reads all the rows in the table, and extracts the rows that meet the criteria of the query. A table scan generates many disk I/O operations and can be resource intensive. However, a table scan could be the most efficient method if, for example, the result set of the query is a high percentage of rows from the table.

When the query optimizer uses an index, it searches the index key columns, finds the storage location of the rows needed by the query and extracts the matching rows from that location. Generally, searching the index is much faster than searching the table because unlike a table, an index frequently contains very few columns per row and the rows are in sorted order.

The query optimizer typically selects the most efficient method when executing queries. However, if no indexes are available, the query optimizer must use a table scan. Your task is to design and create indexes that are best suited to your environment so that the query optimizer has a selection of efficient indexes from which to select. SQL Server provides the [Database Engine Tuning Advisor](http://msdn.microsoft.com/en-us/library/hh231122.aspx) to help with the analysis of your database environment and in the selection of appropriate indexes.