**Clustered and Nonclustered Indexes Described**

**SQL Server 2012**

[Other Versions](javascript:;)

http://i.technet.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [SQL Server 2008 R2](http://technet.microsoft.com/en-us/library/ms190457(d=printer,v=sql.105).aspx)
* [SQL Server 2008](http://technet.microsoft.com/en-us/library/ms190457(d=printer,v=sql.100).aspx)
* [SQL Server 2005](http://technet.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://technet.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://technet.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://technet.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://technet.microsoft.com/en-us/library/ms189039.aspx) and [Create Unique Constraints](http://technet.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://technet.microsoft.com/en-us/library/hh231122.aspx) to help with the analysis of your database environment and in the selection of appropriate indexes.

**Indexes**

**SQL Server 2012**

[Other Versions](javascript:;)

http://i.technet.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [SQL Server 2008 R2](http://technet.microsoft.com/en-us/library/ms175049(d=printer,v=sql.105).aspx)
* [SQL Server 2008](http://technet.microsoft.com/en-us/library/ms175049(d=printer,v=sql.100).aspx)
* [SQL Server 2005](http://technet.microsoft.com/en-us/library/ms175049(d=printer,v=sql.90).aspx)

The following table lists the types of indexes available in SQL Server and provides links to additional information.

|  |  |  |
| --- | --- | --- |
| **Index type** | **Description** | **Additional information** |
| Clustered | A clustered index sorts and stores the data rows of the table or view in order based on the clustered index key. The clustered index is implemented as a B-tree index structure that supports fast retrieval of the rows, based on their clustered index key values. | [Clustered and Nonclustered Indexes Described](http://technet.microsoft.com/en-us/library/ms190457.aspx)  [Create Clustered Indexes](http://technet.microsoft.com/en-us/library/ms186342.aspx) |
| Nonclustered | A nonclustered index can be defined on a table or view with a clustered index or on 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 rows in the index are stored in the order of the index key values, but the data rows are not guaranteed to be in any particular order unless a clustered index is created on the table. | [Clustered and Nonclustered Indexes Described](http://technet.microsoft.com/en-us/library/ms190457.aspx)  [Create Nonclustered Indexes](http://technet.microsoft.com/en-us/library/ms189280.aspx) |
| Unique | A unique index ensures that the index key contains no duplicate values and therefore every row in the table or view is in some way unique.  Uniqueness can be a property of both clustered and nonclustered indexes. | [Create Unique Indexes](http://technet.microsoft.com/en-us/library/ms187019.aspx) |
| Columnstore | An xVelocity memory optimized columnstore index based on vertical partitioning of the data by columns, stored as large objects (LOB). | [Columnstore Indexes](http://technet.microsoft.com/en-us/library/gg492088.aspx) |
| Index with included columns | A nonclustered index that is extended to include nonkey columns in addition to the key columns. | [Create Indexes with Included Columns](http://technet.microsoft.com/en-us/library/ms190806.aspx) |
| Index on computed columns | An index on a column that is derived from the value of one or more other columns, or certain deterministic inputs. | [Indexes on Computed Columns](http://technet.microsoft.com/en-us/library/ms189292.aspx) |
| Filtered | An optimized nonclustered index, especially suited to cover queries that select from a well-defined subset of data. It uses a filter predicate to index a portion of rows in the table. A well-designed filtered index can improve query performance, reduce index maintenance costs, and reduce index storage costs compared with full-table indexes. | [Create Filtered Indexes](http://technet.microsoft.com/en-us/library/cc280372.aspx) |
| Spatial | A spatial index provides the ability to perform certain operations more efficiently on spatial objects (spatial data) in a column of the **geometry** data type. The spatial index reduces the number of objects on which relatively costly spatial operations need to be applied. | [Spatial Indexes Overview](http://technet.microsoft.com/en-us/library/bb895265.aspx) |
| XML | A shredded, and persisted, representation of the XML binary large objects (BLOBs) in the xml data type column. | [XML Indexes (SQL Server)](http://technet.microsoft.com/en-us/library/ms191497.aspx) |
| Full-text | A special type of token-based functional index that is built and maintained by the Microsoft Full-Text Engine for SQL Server. It provides efficient support for sophisticated word searches in character string data. | [Populate Full-Text Indexes](http://technet.microsoft.com/en-us/library/ms142575.aspx) |