**Nonclustered Index Design Guidelines**

**SQL Server 2008 R2**

[Other Versions](javascript:;)

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

* [SQL Server 2008](http://msdn.microsoft.com/en-us/library/ms179325(d=printer,v=sql.100).aspx)
* [SQL Server 2005](http://msdn.microsoft.com/en-us/library/ms179325(d=printer,v=sql.90).aspx)

A nonclustered index contains the index key values and row locators that point to the storage location of the table data. For more information about nonclustered index architecture, see [Nonclustered Index Structures](http://msdn.microsoft.com/en-us/library/ms177484(v=sql.105).aspx).

You can create multiple nonclustered indexes on a table or indexed view. Generally, nonclustered indexes should be designed to improve the performance of frequently used queries that are not covered by the clustered index.

Similar to the way you use an index in a book, the query optimizer searches for a data value by searching the nonclustered index to find the location of the data value in the table and then retrieves the data directly from that location. This makes nonclustered indexes the optimal choice for exact match queries because the index contains entries describing the exact location in the table of the data values being searched for in the queries. For example, to query the Person.Person table for people that have a certain last name, the query optimizer might use the nonclustered index IX\_Person\_LastName\_FirstName\_MiddleName; this has LastName as one of its key columns. The query optimizer can quickly find all entries in the index that match the specified LastName. Each index entry points to the exact page and row in the table, or clustered index, in which the corresponding data can be found. After the query optimizer finds all entries in the index, it can go directly to the exact page and row to retrieve the data.

[Database Considerations](javascript:void(0))

Consider the characteristics of the database when designing nonclustered indexes.

* Databases or tables with low update requirements, but large volumes of data can benefit from many nonclustered indexes to improve query performance. Consider creating filtered indexes for well-defined subsets of data to improve query performance, reduce index storage costs, and reduce index maintenance costs compared with full-table nonclustered indexes.

Decision Support System applications and databases that contain primarily read-only data can benefit from many nonclustered indexes. The query optimizer has more indexes to choose from to determine the fastest access method, and the low update characteristics of the database mean index maintenance will not impede performance.

* Online Transaction Processing applications and databases that contain heavily updated tables should avoid over-indexing. Additionally, indexes should be narrow, that is, with as few columns as possible.

Large numbers of indexes on a table affect the performance of INSERT, UPDATE, DELETE, and MERGE statements because all indexes must be adjusted appropriately as data in the table changes.

[Query Considerations](javascript:void(0))

Before you create nonclustered indexes, you should understand how your data will be accessed. Consider using a nonclustered index for queries that have the following attributes:

* Use JOIN or GROUP BY clauses.

Create multiple nonclustered indexes on columns involved in join and grouping operations, and a clustered index on any foreign key columns.

* Queries that do not return large result sets.

Create filtered indexes to cover queries that return a well-defined subset of rows from a large table.

* Contain columns frequently involved in search conditions of a query, such as WHERE clause, that return exact matches.

[Column Considerations](javascript:void(0))

Consider columns that have one or more of these attributes:

* Cover the query.

Performance gains are achieved when the index contains all columns in the query. The query optimizer can locate all the column values within the index; table or clustered index data is not accessed resulting in fewer disk I/O operations. Use index with included columns to add covering columns instead of creating a wide index key. For more information, see [Index with Included Columns](http://msdn.microsoft.com/en-us/library/ms190806(v=sql.105).aspx).

If the table has a clustered index, the column or columns defined in the clustered index are automatically appended to the end of each nonclustered index on the table. This can produce a covered query without specifying the clustered index columns in the definition of the nonclustered index. For example, if a table has a clustered index on column C, a nonclustered index on columns B and A will have as its key values columns B, A, and C.

* Lots of distinct values, such as a combination of last name and first name, if a clustered index is used for other columns.

If there are very few distinct values, such as only 1 and 0, most queries will not use the index because a table scan is generally more efficient. For this type of data, consider creating a filtered index on a distinct value that only occurs in a small number of rows. For example, if most of the values are 0, the query optimizer might use a filtered index for the data rows that contain 1.

[Index Options](javascript:void(0))

There are several index options that can be specified when you create a nonclustered index. You should give special consideration to the following options:

* FILLFACTOR
* ONLINE

For more information, see [Setting Index Options](http://msdn.microsoft.com/en-us/library/ms188677(v=sql.105).aspx).

**Setting Index Options**

**SQL Server 2008 R2**

[Other Versions](javascript:;)

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

* [SQL Server 2008](http://msdn.microsoft.com/en-us/library/ms188677(d=printer,v=sql.100).aspx)
* [SQL Server 2005](http://msdn.microsoft.com/en-us/library/ms188677(d=printer,v=sql.90).aspx)
* [SQL Server 2012](http://msdn.microsoft.com/en-us/library/ms188677(d=printer,v=sql.110).aspx)

When you design, create, or modify an index, there are several index options to consider. These options can be specified when you first create an index or when you rebuild an index. Additionally, you can set some index options at any time by using the SET clause of the ALTER INDEX statement.

|  |  |  |  |
| --- | --- | --- | --- |
| **Index option** | **Description** | **Setting stored in metadata** | **Related topic** |
| PAD\_INDEX | Sets the percentage of free space in the intermediate level pages during index creation. | Yes | [Fill Factor](http://msdn.microsoft.com/en-us/library/ms177459(v=sql.105).aspx) |
| FILLFACTOR | Sets the percentage of free space in the leaf level of each index page during index creation. | Yes | [Fill Factor](http://msdn.microsoft.com/en-us/library/ms177459(v=sql.105).aspx) |
| SORT\_IN\_TEMPDB | Determines where the intermediate sort results, generated during index creation, are stored.  When ON, the sort results are stored in [tempdb](http://msdn.microsoft.com/en-us/library/ms190768(v=sql.105).aspx). When OFF, the sort results are stored in the filegroup or partition scheme in which the resulting index is stored.  Note**Note**  If a sort operation is not required, or if the sort can be performed in memory, SORT\_IN\_TEMPDB is ignored. | No | [tempdb and Index Creation](http://msdn.microsoft.com/en-us/library/ms188281(v=sql.105).aspx) |
| IGNORE\_DUP\_KEY | Specifies the error response when an insert operation attempts to insert duplicate key values into a unique index. The IGNORE\_DUP\_KEY option applies only to insert operations after the index is created or rebuilt. The default is OFF. | Yes | [CREATE INDEX](http://msdn.microsoft.com/en-us/library/ms188783(v=sql.105).aspx) |
| STATISTICS\_NORECOMPUTE | Specifies whether out-of-date index statistics should be automatically recomputed. | Yes | [Index Statistics](http://msdn.microsoft.com/en-us/library/ms190397(v=sql.105).aspx) |
| DROP\_EXISTING | Indicates the existing index should be dropped and recreated. | No | [Reorganizing and Rebuilding Indexes](http://msdn.microsoft.com/en-us/library/ms189858(v=sql.105).aspx) |
| ONLINE | Determines whether concurrent user access to the underlying table or clustered index data and any associated nonclustered indexes is allowed during index operations.  Note**Note**  Online index operations are available only in SQL Server Enterprise, Developer, and Evaluation editions. | No | [Performing Index Operations Online](http://msdn.microsoft.com/en-us/library/ms177442(v=sql.105).aspx). |
| ALLOW\_ROW\_LOCKS | Determines whether row locks are used in accessing index data. | Yes | [CREATE INDEX](http://msdn.microsoft.com/en-us/library/ms188783(v=sql.105).aspx)  [Customizing Locking for an Index](http://msdn.microsoft.com/en-us/library/ms189076(v=sql.105).aspx) |
| ALLOW\_PAGE\_LOCKS | Determines whether page locks are used in accessing index data. | Yes | [CREATE INDEX](http://msdn.microsoft.com/en-us/library/ms188783(v=sql.105).aspx)  [Customizing Locking for an Index](http://msdn.microsoft.com/en-us/library/ms189076(v=sql.105).aspx) |
| MAXDOP | Sets the maximum number of processors the query processor can use to execute a single index statement. Fewer processors may be used depending on the current system workload.  Note**Note**  Parallel index operations are available only in SQL Server Enterprise, Developer, and Evaluation editions. | No | [Modifying Indexes](http://msdn.microsoft.com/en-us/library/ms189329(v=sql.105).aspx) |
| DATA\_COMPRESSION | Specifies the data compression option for the specified table, partition number or range of partitions. Options are NONE, ROW, and PAGE. | Yes | [Creating Compressed Tables and Indexes](http://msdn.microsoft.com/en-us/library/cc280449(v=sql.105).aspx) |

To set options on an index

* [ALTER INDEX (Transact-SQL)](http://msdn.microsoft.com/en-us/library/ms188388(v=sql.105).aspx)

[Setting Options Without Rebuilding](javascript:void(0))

By using the SET clause in the ALTER INDEX statement, you can set the following index options without rebuilding the index:

* ALLOW\_PAGE\_LOCKS
* ALLOW\_ROW\_LOCKS
* IGNORE\_DUP\_KEY
* STATISTICS\_NORECOMPUTE

These options are immediately applied to the index. Other index options, such as FILLFACTOR and ONLINE, can be specified only when an index is created or rebuilt.

[Viewing Index Option Settings](javascript:void(0))

Not all index options values are stored in metadata. Those values that are stored in metadata can be viewed in the appropriate catalog views. To examine the current option settings on existing indexes, use the [sys.indexes](http://msdn.microsoft.com/en-us/library/ms173760(v=sql.105).aspx) catalog view . To examine the current value for STATISTICS\_NORECOMPUTE, use the [sys.stats](http://msdn.microsoft.com/en-us/library/ms177623(v=sql.105).aspx) catalog view. For more information, see [Viewing Index Information](http://msdn.microsoft.com/en-us/library/ms188280(v=sql.105).aspx).

[Examples](javascript:void(0))

The following example sets the ALLOW\_ROW\_LOCKS and IGNORE\_DUP\_KEY options for the AK\_Product\_ProductNumber index on the Production.Product table.

USE AdventureWorks2008R2;

GO

--Verify the current values for these options.

SELECT allow\_row\_locks, ignore\_dup\_key

FROM sys.indexes

WHERE name = N'AK\_Product\_ProductNumber';

GO

--Set the ALLOW\_ROW\_LOCKS option OFF and the IGNORE\_DUP\_KEY option ON.

ALTER INDEX AK\_Product\_ProductNumber

ON Production.Product

SET (ALLOW\_ROW\_LOCKS = OFF, IGNORE\_DUP\_KEY = ON);

GO

--Verify the new values for these options.

SELECT allow\_row\_locks, ignore\_dup\_key

FROM sys.indexes

WHERE name = N'AK\_Product\_ProductNumber';

GO