SQL Server: Indexes

techonthenet.com/sql_server/indexes.php

Learn how to **create, rename and drop indexes** in SQL Server with syntax and examples.

What is an Index in SQL Server?

An index is a performance-tuning method of allowing faster retrieval of records. An index creates an entry for each value that appears in the indexed columns.

Create an Index

Syntax

The syntax for creating an index in SQL Server (Transact-SQL) is:

```
CREATE [ UNIQUE ] [ CLUSTERED | NONCLUSTERED ] INDEX index_name
  ON table_name ( column1 [ASC | DESC ], ...  column_n [ ASC | DESC ] )
  [ INCLUDE ( column1, ... column_n ) ]
  [ WHERE condition ]
  [ WITH ( PAD_INDEX = { ON | OFF }
         | FILLFACTOR = fillfactor
         | SORT_IN_TEMPDB = { ON | OFF }
         | IGNORE_DUP_KEY = { ON | OFF }
         | STATISTICS_NORECOMPUTE = { ON | OFF }
         | STATISTICS_INCREMENTAL = { ON | OFF }
         | DROP_EXISTING = { ON | OFF }
         | ONLINE = { ON | OFF }
         | ALLOW_ROW_LOCKS = { ON | OFF }
         | ALLOW_PAGE_LOCKS = { ON | OFF }
         | MAXDOP = max_degree
         | DATA_COMPRESSION = { NONE | PAGE | ROW }
            [ ON PARTITIONS ( { number | range } ]
  [ ON partition_scheme ( column )
  | ON filegroup
  | ON default_filegroup ]
  [ FILESTREAM_ON { filegroup | partition_scheme };
```

UNIQUE
Optional. Indicates that the combination of values in the indexed columns must be unique.

CLUSTERED
Optional. Indicates that the logical order determines the physical order of the rows in the table.

NONCLUSTERED
Optional. Indicates that the logical order does not determine the physical order of the rows in the table.

index_name

The name of the index to create.

table_name
The name of the table or view on which the index is to be created.

column1, ... column_n
The columns to base the index.

ASC | DESC
The sort order for each of the columns.

INCLUDE (column1, ... column_n)
Optional. The columns that are not key columns to add to the leaf level of the nonclustered index.

WHERE condition
Optional. The condition to determine which rows to include in the index.

ON partition_scheme (column)
Optional. Indicates that the partition schema determines the filegroups in which the partitions will be mapped.

ON filegroup
Optional. Indicates that the index will be created on the specified filegroup.

ON default_filegroup
Optional. Indicates the default filegroup.

FILESTREAM_ON { filegroup | partition_scheme }
Optional. Indicates where to place the FILESTREAM data for a clustered index.

Index Example

Let's look at an example of how to create an index in SQL Server (Transact-SQL).

For example:

```
CREATE INDEX contacts_idx
  ON contacts (last_name);
```

In this example, we've created an index on the *contacts* table called *contacts_idx*. It consists of only one field - the *last_name* field.

We could also create an index with more than one field as in the example below:

```
CREATE INDEX contacts_idx
  ON contacts (last_name, first_name);
```

In this example, we've created an index on the *contacts* table called *contacts_idx* but this time, it consists of the *last_name* and *first_name* fields.

Since we have not specified ASC | DESC to each of the columns, the index is created with each of the fields in ascending order. We could modify our example and change the sort orders to descending as follows:

```
CREATE INDEX contacts_idx
  ON contacts (last_name DESC, first_name DESC);
```

This CREATE INDEX example will create the contacts_idx index with the last_name sorted in descending order and the first_name sorted in descending order.

UNIQUE Index Example

Next, let's look at an example of how to create a unique index in SQL Server (Transact-SQL).

For example:

```
CREATE UNIQUE INDEX contacts_uidx
  ON contacts (last_name, first_name);
```

This example would create an index called *contacts_uidx* on that *contacts* table that consists of the *last_name* and *first_name* fields, but also ensures that the there are only unique combinations of the two fields.

You could modify this example further to make the unique index also clustered so that the physical order of the rows in the table is determined by the logical order of the index.

For example:

```
CREATE UNIQUE CLUSTERED INDEX contacts_uidx
  ON contacts (last_name, first_name);
```

This example creates an index called *contacts_uidx* that is a *unique* index based on the *last_name* and *first_name* fields and the index is also clustered which changes the physical order of the rows in the table.

Rename an Index

Syntax

The syntax for renaming an index in SQL Server (Transact-SQL) is:

```
sp_rename 'table_name.old_index_name', 'new_index_name', 'INDEX';
```

table_name
The name of the table where the index has been created.

old_index_name
The name of the index that you wish to rename.

new_index_name
The new name for the index.

Example

```
sp_rename 'contacts.contacts_idx', 'contacts_index_cname', 'INDEX';
```

In this example, we're renaming the index on the contacts table called *contacts_idx* to *contacts_index_cname*.

Drop an Index

Syntax

The syntax for dropping an index in SQL Server is:

```
DROP INDEX table_name.index_name;
```

table_name
The name of the table where the index has been created.

index_name
The name of the index to drop.

Example

Let's look at example of how to drop an index in SQL Server (Transact-SQL).

For example:

```
DROP INDEX contacts.contacts_idx;
```

In this example, we're dropping an index called supplier_idx.