

REBUILD INDEXES ONLINE WITH SQL SERVER 2005

Takeaway: Maintaining indexes is one of the key concerns for any database administrator. **SQL Server 2005** introduces a new feature that enhances the database administrator's ability to maintain indexes.

Indexes are specialized data structures that operate on tables (*and sometimes views*) in the database engine used to aid in the searching for and sorting of data. **Indexes** are vital to the database engine returning results quickly. As data is modified in the underlying tables that the indexes operate on, the indexes become fragmented. Fragmentation is when the logical ordering of an index does not match the physical ordering of the underlying table or view. As the indexes become more and more fragmented, query times can begin to suffer. The **remedy** to this situation is to either reorganize or rebuild the index in **SQL Server 2005**.

(**Note:** This feature is only available in the **Enterprise Edition** of the product.)

REORGANIZE VS. REBUILD

Fragmented indexes can be "**unfragmented**" in two ways: They can be reorganized or rebuilt. Reorganizing an index causes the reordering of the data inside of the outermost data pages and will compact the index. No additional data is added to the index for the reorganization, so the index may remain somewhat fragmented. The operation doesn't take a lot of system resources and may occur while outside processes are accessing the table that the index operates on, so it is said to be an "online" operation.

Rebuilding an index essentially drops the desired index and creates a new one. Any fragmentation that was in the older index is removed, and the logical ordering of the new index matches the physical ordering. Due to the fact that the index is removed and recreated, outside processes are not able to access the table and performance can suffer. In fact, other processes will not be able to lock the table at all while the index rebuild is occurring. This is a major hindrance of rebuilding indexes.

ONLINE INDEX REBUILD

SQL Server 2005 introduces the ability to rebuild your indexes in an online fashion so that other processes are able to access the table while the rebuild is occurring. Because you can access the indexes during the rebuild, you are not limited to only rebuilding indexes during off-peak hours.

To accomplish this, the database engine takes some special actions to rebuild the index and to allow access to the index at the same time. The original index will remain available to users for reading data and data modification. Row versioning is used to allow for transactional consistency. During the rebuild, a new index is created that mimics the old index. Any data modifications that alter the original index will also be applied to this index by **SQL Server** during the rebuild. This new index is not read from at all — it is write-only.

It is essential that you have enough available disk space to accommodate the data for the two concurrent indexes during the online rebuild. While the rebuild is taking place, **SQL Server** uses a mapping index to determine records to modify in the new index when modifications occur in the original index. Once the rebuild process has finished, any queries or data modifications occur to the new index, and the original index is dropped.

EXAMPLE

The process to rebuild an index online is not much different than the typical rebuild process; however, there are a few ways to accomplish the rebuild. One way is to simply drop the index using a **DROP INDEX** statement followed by a **CREATE INDEX** statement. *Rebuilding indexes* in this fashion leaves the table without an index until the index is completely created. For this reason (and a number of other reasons), dropping the index and recreating it is not recommended.

The **CREATE INDEX** statement can still be used to rebuild an index if the **DROP_EXISTING** option is used. This feature allows the definition of the specified index to change, and it allows the **DBA** to change the location of the index to another filegroup or partition.

The **ALTER INDEX** statement allows for the rebuilding of the clustered and all nonclustered indexes on the table. The drawback with this statement is that you cannot change the index definition. Both of these statements have options that will build the index online.

The following statement will rebuild the clustered index (which is on the **SaleID** column) on the **SalesHistory** table. The existing index will be dropped in the process, but it will be available during the operation because the **ONLINE** option is specified.

```
CREATE CLUSTERED INDEX cl_SalesHistory_SaleID ON SalesHistory(SaleID) WITH(DROP_EXISTING
= ON, ONLINE = ON)
```

This statement is very similar to the one used above, but it changes the actual index definition to include an additional column. The index is still rebuilt in the same fashion.

```
CREATE CLUSTERED INDEX cl_SalesHistory_SaleID ON SalesHistory(SaleID ASC, SaleDate ASC)
WITH(DROP_EXISTING = ON, ONLINE = ON)
```

Using the **ALTER INDEX** statement, I can rebuild all indexes on a specified table. The **ONLINE** syntax remains the same as the **CREATE INDEX** statement. This new syntax replaces the **DBCC DBREINDEX** statement used in previous versions of **SQL Server**.

```
ALTER INDEX ALL ON SalesHistory REBUILD WITH(ONLINE = ON)
```

This statement rebuilds the clustered index on the **SalesHistory** table. The **ONLINE** option is omitted, which means that the table will not be accessible during the rebuild operation.

```
ALTER INDEX cl_SalesHistory_SaleID ON SalesHistory REBUILD
```

This statement is the same statement as above, but the rebuild will be performed online, so operations can continue to be performed on the table.

```
ALTER INDEX cl_SalesHistory_SaleID ON SalesHistory REBUILD WITH (ONLINE = ON)
```

CONSIDERATIONS

The ability to rebuild indexes in **SQL Server 2005** is a fantastic new option. If you rebuild your indexes online, you must ensure that you have enough available disk space to hold the index that is being created along with the preexisting index. After the rebuild operation, the old index will be dropped.

Also, rebuilding indexes online takes a significant amount more time and resources than just rebuilding the index. This is usually a pretty good tradeoff since the table will remain available during the operation.