

Contents

[Overview](#)

[Create](#)

[Modify a Partition Function](#)

[Modify a Partition Scheme](#)

[Manage Partition Wizard F1 Help](#)

Partitioned Tables and Indexes

7/2/2018 • 10 minutes to read • [Edit Online](#)

THIS TOPIC APPLIES TO:  SQL Server  Azure SQL Database  Azure SQL Data Warehouse  Parallel Data Warehouse

SQL Server supports table and index partitioning. The data of partitioned tables and indexes is divided into units that can be spread across more than one filegroup in a database. The data is partitioned horizontally, so that groups of rows are mapped into individual partitions. All partitions of a single index or table must reside in the same database. The table or index is treated as a single logical entity when queries or updates are performed on the data. Prior to SQL Server 2016 (13.x) SP1, partitioned tables and indexes were not available in every edition of SQL Server. For a list of features that are supported by the editions of SQL Server, see [Editions and Supported Features for SQL Server 2016](#).

IMPORTANT

SQL Server 2017 supports up to 15,000 partitions by default. In versions earlier than SQL Server 2012 (11.x), the number of partitions was limited to 1,000 by default. On x86-based systems, creating a table or index with more than 1000 partitions is possible, but is not supported.

Benefits of Partitioning

Partitioning large tables or indexes can have the following manageability and performance benefits.

- You can transfer or access subsets of data quickly and efficiently, while maintaining the integrity of a data collection. For example, an operation such as loading data from an OLTP to an OLAP system takes only seconds, instead of the minutes and hours the operation takes when the data is not partitioned.
- You can perform maintenance operations on one or more partitions more quickly. The operations are more efficient because they target only these data subsets, instead of the whole table. For example, you can choose to compress data in one or more partitions or rebuild one or more partitions of an index.
- You may improve query performance, based on the types of queries you frequently run and on your hardware configuration. For example, the query optimizer can process equi-join queries between two or more partitioned tables faster when the partitioning columns in the tables are the same, because the partitions themselves can be joined.

When SQL Server performs data sorting for I/O operations, it sorts the data first by partition. SQL Server accesses one drive at a time, and this might reduce performance. To improve data sorting performance, stripe the data files of your partitions across more than one disk by setting up a RAID. In this way, although SQL Server still sorts data by partition, it can access all the drives of each partition at the same time.

In addition, you can improve performance by enabling lock escalation at the partition level instead of a whole table. This can reduce lock contention on the table.

Components and Concepts

The following terms are applicable to table and index partitioning.

Partition function

A database object that defines how the rows of a table or index are mapped to a set of partitions based on the values of certain column, called a partitioning column. That is, the partition function defines the number of

partitions that the table will have and how the boundaries of the partitions are defined. For example, given a table that contains sales order data, you may want to partition the table into twelve (monthly) partitions based on a **datetime** column such as a sales date.

Partition scheme

A database object that maps the partitions of a partition function to a set of filegroups. The primary reason for placing your partitions on separate filegroups is to make sure that you can independently perform backup operations on partitions. This is because you can perform backups on individual filegroups.

Partitioning column

The column of a table or index that a partition function uses to partition the table or index. Computed columns that participate in a partition function must be explicitly marked **PERSISTED**. All data types that are valid for use as index columns can be used as a partitioning column, except **timestamp**. The **ntext**, **text**, **image**, **xml**, **varchar(max)**, **nvarchar(max)**, or **varbinary(max)** data types cannot be specified. Also, Microsoft .NET Framework common language runtime (CLR) user-defined type and alias data type columns cannot be specified.

Aligned index

An index that is built on the same partition scheme as its corresponding table. When a table and its indexes are in alignment, SQL Server can switch partitions quickly and efficiently while maintaining the partition structure of both the table and its indexes. An index does not have to participate in the same named partition function to be aligned with its base table. However, the partition function of the index and the base table must be essentially the same, in that 1) the arguments of the partition functions have the same data type, 2) they define the same number of partitions, and 3) they define the same boundary values for partitions.

Nonaligned index

An index partitioned independently from its corresponding table. That is, the index has a different partition scheme or is placed on a separate filegroup from the base table. Designing a nonaligned partitioned index can be useful in the following cases:

- The base table has not been partitioned.
- The index key is unique and it does not contain the partitioning column of the table.
- You want the base table to participate in colocated joins with more tables using different join columns.

Partition elimination The process by which the query optimizer accesses only the relevant partitions to satisfy the filter criteria of the query.

Performance Guidelines

The new, higher limit of 15,000 partitions affects memory, partitioned index operations, DBCC commands, and queries. This section describes the performance implications of increasing the number of partitions above 1,000 and provides workarounds as needed. With the limit on the maximum number of partitions being increased to 15,000, you can store data for a longer time. However, you should retain data only for as long as it is needed and maintain a balance between performance and number of partitions.

Processor Cores and Number of Partitions Guidelines

To maximize performance with parallel operations, we recommend that you use the same number of partitions as processor cores, up to a maximum of 64 (which is the maximum number of parallel processors that SQL Server can utilize).

Memory Usage and Guidelines

We recommend that you use at least 16 GB of RAM if a large number of partitions are in use. If the system does not have enough memory, Data Manipulation Language (DML) statements, Data Definition Language (DDL) statements and other operations can fail due to insufficient memory. Systems with 16 GB of RAM that run many memory-intensive processes may run out of memory on operations that run on a large number of partitions.

Therefore, the more memory you have over 16 GB, the less likely you are to encounter performance and memory issues.

Memory limitations can affect the performance or ability of SQL Server to build a partitioned index. This is especially the case when the index is not aligned with its base table or is not aligned with its clustered index, if the table already has a clustered index applied to it.

Partitioned Index Operations

Memory limitations can affect the performance or ability of SQL Server to build a partitioned index. This is especially the case with nonaligned indexes. Creating and rebuilding nonaligned indexes on a table with more than 1,000 partitions is possible, but is not supported. Doing so may cause degraded performance or excessive memory consumption during these operations.

Creating and rebuilding aligned indexes could take longer to execute as the number of partitions increases. We recommend that you do not run multiple create and rebuild index commands at the same time as you may run into performance and memory issues.

When SQL Server performs sorting to build partitioned indexes, it first builds one sort table for each partition. It then builds the sort tables either in the respective filegroup of each partition or in **tempdb**, if the `SORT_IN_TEMPDB` index option is specified. Each sort table requires a minimum amount of memory to build. When you are building a partitioned index that is aligned with its base table, sort tables are built one at a time, using less memory. However, when you are building a nonaligned partitioned index, the sort tables are built at the same time. As a result, there must be sufficient memory to handle these concurrent sorts. The larger the number of partitions, the more memory required. The minimum size for each sort table, for each partition, is 40 pages, with 8 kilobytes per page. For example, a nonaligned partitioned index with 100 partitions requires sufficient memory to serially sort 4,000 (40 * 100) pages at the same time. If this memory is available, the build operation will succeed, but performance may suffer. If this memory is not available, the build operation will fail. Alternatively, an aligned partitioned index with 100 partitions requires only sufficient memory to sort 40 pages, because the sorts are not performed at the same time.

For both aligned and nonaligned indexes, the memory requirement can be greater if SQL Server is applying degrees of parallelism to the build operation on a multiprocessor computer. This is because the greater the degrees of parallelism, the greater the memory requirement. For example, if SQL Server sets degrees of parallelism to 4, a nonaligned partitioned index with 100 partitions requires sufficient memory for four processors to sort 4,000 pages at the same time, or 16,000 pages. If the partitioned index is aligned, the memory requirement is reduced to four processors sorting 40 pages, or 160 (4 * 40) pages. You can use the `MAXDOP` index option to manually reduce the degrees of parallelism.

DBCC Commands

With a larger number of partitions, DBCC commands could take longer to execute as the number of partitions increases.

Queries

Queries that use partition elimination could have comparable or improved performance with larger number of partitions. Queries that do not use partition elimination could take longer to execute as the number of partitions increases.

For example, assume a table has 100 million rows and columns `A`, `B`, and `C`. In scenario 1, the table is divided into 1000 partitions on column `A`. In scenario 2, the table is divided into 10,000 partitions on column `A`. A query on the table that has a `WHERE` clause filtering on column `A` will perform partition elimination and scan one partition. That same query may run faster in scenario 2 as there are fewer rows to scan in a partition. A query that has a `WHERE` clause filtering on column `B` will scan all partitions. The query may run faster in scenario 1 than in scenario 2 as there are fewer partitions to scan.

Queries that use operators such as `TOP` or `MAX/MIN` on columns other than the partitioning column may

experience reduced performance with partitioning because all partitions must be evaluated.

Behavior Changes in Statistics Computation During Partitioned Index Operations

Beginning with SQL Server 2012 (11.x), statistics are not created by scanning all the rows in the table when a partitioned index is created or rebuilt. Instead, the query optimizer uses the default sampling algorithm to generate statistics. After upgrading a database with partitioned indexes, you may notice a difference in the histogram data for these indexes. This change in behavior may not affect query performance. To obtain statistics on partitioned indexes by scanning all the rows in the table, use `CREATE STATISTICS` or `UPDATE STATISTICS` with the `FULLSCAN` clause.

Related Tasks

| Tasks | Topic |
|--|---|
| Describes how to create partition functions and partition schemes and then apply these to a table and index. | Create Partitioned Tables and Indexes |
| | |




Related Content

You may find the following white papers on partitioned table and index strategies and implementations useful.

- [Partitioned Table and Index Strategies Using SQL Server 2008](#)
- [How to Implement an Automatic Sliding Window](#)
- [Bulk Loading into a Partitioned Table](#)
- [Project REAL: Data Lifecycle -- Partitioning](#)
- [Query Processing Enhancements on Partitioned Tables and Indexes](#)
- [Top 10 Best Practices for Building a Large Scale Relational Data Warehouse](#)

Create Partitioned Tables and Indexes

7/2/2018 • 15 minutes to read • [Edit Online](#)

THIS TOPIC APPLIES TO:  SQL Server  Azure SQL Database  Azure SQL Data Warehouse  Parallel Data Warehouse

You can create a partitioned table or index in SQL Server 2017 by using SQL Server Management Studio or Transact-SQL. The data in partitioned tables and indexes is horizontally divided into units that can be spread across more than one filegroup in a database. Partitioning can make large tables and indexes more manageable and scalable.

Creating a partitioned table or index typically happens in four parts:

1. Create a filegroup or filegroups and corresponding files that will hold the partitions specified by the partition scheme.
2. Create a partition function that maps the rows of a table or index into partitions based on the values of a specified column.
3. Create a partition scheme that maps the partitions of a partitioned table or index to the new filegroups.
4. Create or modify a table or index and specify the partition scheme as the storage location.

In This Topic

- **Before you begin:**

- [Limitations and Restrictions](#)

- [Security](#)

- **To create a partitioned table or index, using:**

- [SQL Server Management Studio](#)

- [Transact-SQL](#)

Before You Begin

Limitations and Restrictions

- The scope of a partition function and scheme is limited to the database in which they have been created. Within the database, partition functions reside in a separate namespace from other functions.
- If any rows within a partition function have partitioning columns with null values, these rows are allocated to the left-most partition. However, if NULL is specified as a boundary value and RIGHT is indicated, the left-most partition remains empty and NULL values are placed in the second partition.

Security

Permissions

Creating a partitioned table requires CREATE TABLE permission in the database and ALTER permission on the schema in which the table is being created. Creating a partitioned index requires ALTER permission on the table or view where the index is being created. Creating either a partitioned table or index requires any one of the following additional permissions:

- ALTER ANY DATASPACE permission. This permission defaults to members of the **sysadmin** fixed server role and the **db_owner** and **db_ddladmin** fixed database roles.

- CONTROL or ALTER permission on the database in which the partition function and partition scheme are being created.
- CONTROL SERVER or ALTER ANY DATABASE permission on the server of the database in which the partition function and partition scheme are being created.

Using SQL Server Management Studio

Follow the steps in this procedure to create one or more filegroups, corresponding files, and a table. You will reference these objects in the next procedure when you create the partitioned table.

To create new filegroups for a partitioned table

1. In Object Explorer, right-click the database in which you want to create a partitioned table and select **Properties**.
2. In the **Database Properties – database_name** dialog box, under **Select a page**, select **Filegroups**.
3. Under **Rows**, click **Add**. In the new row, enter the filegroup name.

WARNING

You must always have one extra filegroup in addition to the number of filegroups specified for the boundary values when you are creating partitions.

4. Continue adding rows until you have created all of the filegroups for the partitioned table.
5. Click **OK**.
6. Under **Select a page**, select **Files**.
7. Under **Rows**, click **Add**. In the new row, enter a filename and select a filegroup.
8. Continue adding rows until you have created at least one file for each filegroup.
9. Expand the **Tables** folder and create a table as you normally would. For more information, see [Create Tables \(Database Engine\)](#). Alternatively, you can specify an existing table in the next procedure.

To create a partitioned table

1. Right-click the table that you wish to partition, point to **Storage**, and then click **Create Partition....**
2. In the **Create Partition Wizard**, on the **Welcome to the Create Partition Wizard** page, click **Next**.
3. On the **Select a Partitioning Column** page, in the **Available partitioning columns** grid, select the column on which you want to partition your table. Only columns with data types that can be used to partition data will be displayed in the **Available partitioning columns** grid. If you select a computed column as the partitioning column, the column must be designated as a persisted column.

The choices you have for the partitioning column and the values range are determined primarily by the extent to which your data can be grouped in a logical way. For example, you may choose to divide your data into logical groupings by months or quarters of a year. The queries you plan to make against your data will determine whether this logical grouping is adequate for managing your table partitions. All data types are valid for use as partitioning columns, except **text**, **ntext**, **image**, **xml**, **timestamp**, **varchar(max)**, **nvarchar(max)**, **varbinary(max)**, alias data types, or CLR user-defined data types.

The following additional options are available on this page:

Collocate this table to the selected partitioned table

Allows you to select a partitioned table that contains related data to join with this table on the partitioning column. Tables with partitions joined on the partitioning columns are typically queried more efficiently.

Storage-align non-unique indexes and unique indexes with an indexed partition column

Aligns all indexes of the table that are partitioned with the same partition scheme. When a table and its indexes are aligned, you can move partitions in and out of partitioned tables more effectively, because your data is partitioned with the same algorithm.

After selecting the partitioning column and any other options, click **Next**.

4. On the **Select a Partition Function** page, under **Select partition function**, click either **New partition function** or **Existing partition function**. If you choose **New partition function**, enter the name of the function. If you choose **Existing partition function**, select the name of the function you'd like to use from the list. The **Existing partition function** option will not be available if there are no other partition functions on the database.

After completing this page, click **Next**.

5. On the **Select a Partition Scheme** page, under **Select partition scheme**, click either **New partition scheme** or **Existing partition scheme**. If you choose **New partition scheme**, enter the name of the scheme. If you choose **Existing partition scheme**, select the name of the scheme you'd like to use from the list. The **Existing partition scheme** option will not be available if there are no other partition schemes on the database.

After completing this page, click **Next**.

6. On the **Map Partitions** page, under **Range**, select either **Left boundary** or **Right boundary** to specify whether to include the highest or lowest bounding value within each filegroup you create. You must always enter one extra filegroup in addition to the number of filegroups specified for the boundary values when you are creating partitions.

In the **Select filegroups and specify boundary values** grid, under **Filegroup**, select the filegroup into which you want to partition your data. Under **Boundary**, enter the boundary value for each filegroup. If boundary value is left empty, the partition function maps the whole table or index into a single partition using the partition function name.

The following additional options are available on this page:

Set Boundaries...

Opens the **Set Boundary Values** dialog box to select the boundary values and date ranges you want for your partitions. This option is only available when you have selected a partitioning column that contains one of the following data types: **date**, **datetime**, **smalldatetime**, **datetime2**, or **datetimeoffset**.

Estimate storage

Estimates rowcount, required space, and available space for storage for each filegroup specified for the partitions. These values are displayed in the grid as read-only values.

The **Set Boundary Values** dialog box allows for the following additional options:

Start date

Selects the starting date for the range values of your partitions.

End date

Selects the ending date for the range values of your partitions. If you selected **Left boundary** on the **Map Partitions** page, this date will be the last value for each filegroup/partition. If you selected **Right boundary** on the **Map Partitions** page, this date will be the first value in the next-to-last filegroup.

Date range

Selects the date granularity or range value increment you want for each partition.

After completing this page, click **Next**.

7. In the **Select an Output Option** page, specify how you want to complete your partitioned table. Select **Create Script** to create a SQL script based the previous pages in the wizard. Select **Run immediately** to create the new partitioned table after completing all remaining pages in the wizard. Select **Schedule** to create the new partitioned table at a predetermined time in the future.

If you select **Create script**, the following options are available under **Script options**:

Script to file

Generates the script as a .sql file. Enter a file name and location in the **File name** box or click **Browse** to open the **Script File Location** dialog box. From **Save As**, select **Unicode text** or **ANSI text**.

Script to Clipboard

Saves the script to the Clipboard.

Script to New Query Window

Generates the script to a new Query Editor window. This is the default selection.

If you select **Schedule**, click **Change schedule**.

- a. In the **New Job Schedule** dialog box, in the **Name** box, enter the job schedule's name.
- b. On the **Schedule type** list, select the type of schedule:
 - **Start automatically when SQL Server Agent starts**
 - **Start whenever the CPUs become idle**
 - **Recurring**. Select this option if your new partitioned table updates with new information on a regular basis.
 - **One time**. This is the default selection.
- c. Select or clear the **Enabled** check box to enable or disable the schedule.
- d. If you select **Recurring**:
 - a. Under **Frequency**, on the **Occurs** list, specify the frequency of occurrence:
 - If you select **Daily**, in the **Recurs every** box, enter how often the job schedule repeats in days.
 - If you select **Weekly**, in the **Recurs every** box, enter how often the job schedule repeats in weeks. Select the day or days of the week on which the job schedule is run.
 - If you select **Monthly**, select either **Day** or **The**.
 - If you select **Day**, enter both the date of the month you want the job schedule to run and how often the job schedule repeats in months. For example, if you want the job schedule to run on the 15th day of the month every other month, select **Day** and enter "15" in the first box and "2" in the second box. Please note that the largest number allowed in the second box is "99".
 - If you select **The**, select the specific day of the week within the month that you want the job schedule to run and how often the job schedule repeats in months. For example, if you want the job schedule to run on the last weekday of the month every other month, select **Day**, select **last** from the first list and **weekday** from the second list, and then enter "2" in the last box. You can also select **first**, **second**, **third**, or **fourth**, as well as specific weekdays (for example: Sunday or Wednesday) from the first two lists. Please note that the largest number allowed in the last box is "99".

b. Under **Daily frequency**, specify how often the job schedule repeats on the day the job schedule runs:

- If you select **Occurs once at**, enter the specific time of day when the job schedule should run in the **Occurs once at** box. Enter the hour, minute, and second of the day, as well as AM or PM.
- If you select **Occurs every**, specify how often the job schedule runs during the day chosen under **Frequency**. For example, if you want the job schedule to repeat every 2 hours during the day that the job schedule is run, select **Occurs every**, enter "2" in the first box, and then select **hour(s)** from the list. From this list you can also select **minute(s)** and **second(s)**. Please note that the largest number allowed in the first box is "100".

In the **Starting at** box, enter the time that the job schedule should start running. In the **Ending at** box, enter the time that the job schedule should stop repeating. Enter the hour, minute, and second of the day, as well as AM or PM.

- c. Under **Duration**, in **Start date**, enter the date that you want the job schedule to start running. Select **End date** or **No end date** to indicate when the job schedule should stop running. If you select **End date**, enter the date that you want to job schedule to stop running.
- e. If you select **One Time**, under **One-time occurrence**, in the **Date** box, enter the date that the job schedule will be run. In the **Time** box, enter the time that the job schedule will be run. Enter the hour, minute, and second of the day, as well as AM or PM.
- f. Under **Summary**, in **Description**, verify that all job schedule settings are correct.
- g. Click **OK**.

After completing this page, click **Next**.

8. On the **Review Summary** page, under **Review your selections**, expand all available options to verify that all partition settings are correct. If everything is as expected, click **Finish**.
9. On the **Create Partition Wizard Progress** page, monitor status information about the actions of the Create Partition Wizard. Depending on the options that you selected in the wizard, the progress page might contain one or more actions. The top box displays the overall status of the wizard and the number of status, error, and warning messages that the wizard has received.

The following options are available on the **Create Partition Wizard Progress** page:

Details

Provides the action, status, and any messages that are returned from action taken by the wizard.

Action

Specifies the type and name of each action.

Status

Indicates whether the wizard action as a whole returned the value of **Success** or **Failure**.

Message

Provides any error or warning messages that are returned from the process.

Report

Creates a report that contains the results of the Create Partition Wizard. The options are **View Report**, **Save Report to File**, **Copy Report to Clipboard**, and **Send Report as Email**.

View Report

Opens the **View Report** dialog box, which contains a text report of the progress of the Create Partition Wizard.

Save Report to File

Opens the **Save Report As** dialog box.

Copy Report to Clipboard

Copies the results of the wizard's progress report to the Clipboard.

Send Report as Email

Copies the results of the wizard's progress report into an email message.

When complete, click **Close**.

The Create Partition Wizard creates the partition function and scheme and then applies the partitioning to the specified table. To verify the table partitioning, in Object Explorer, right-click the table and select **Properties**. Click the **Storage** page. The page displays information such as the name of the partition function and scheme and the number of partitions.

Using Transact-SQL

To create a partitioned table

1. In **Object Explorer**, connect to an instance of Database Engine.
2. On the Standard bar, click **New Query**.
3. Copy and paste the following example into the query window and click **Execute**. The example creates new filegroups, a partition function, and a partition scheme. A new table is created with the partition scheme specified as the storage location.

```
USE AdventureWorks2012;
GO
-- Adds four new filegroups to the AdventureWorks2012 database
ALTER DATABASE AdventureWorks2012
ADD FILEGROUP test1fg;
GO
ALTER DATABASE AdventureWorks2012
ADD FILEGROUP test2fg;
GO
ALTER DATABASE AdventureWorks2012
ADD FILEGROUP test3fg;
GO
ALTER DATABASE AdventureWorks2012
ADD FILEGROUP test4fg;

-- Adds one file for each filegroup.
ALTER DATABASE AdventureWorks2012
ADD FILE
(
    NAME = test1dat1,
    FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\DATA\t1dat1.ndf',
    SIZE = 5MB,
    MAXSIZE = 100MB,
    FILEGROWTH = 5MB
)
TO FILEGROUP test1fg;
ALTER DATABASE AdventureWorks2012
ADD FILE
(
    NAME = test2dat2,
    FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\DATA\t2dat2.ndf',
    SIZE = 5MB,
    MAXSIZE = 100MB,
    FILEGROWTH = 5MB
)
```

```

        FILEGROWTH = 5MB
    )
    TO FILEGROUP test2fg;
GO
ALTER DATABASE AdventureWorks2012
ADD FILE
(
    NAME = test3dat3,
    FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\DATA\t3dat3.ndf',
    SIZE = 5MB,
    MAXSIZE = 100MB,
    FILEGROWTH = 5MB
)
    TO FILEGROUP test3fg;
GO
ALTER DATABASE AdventureWorks2012
ADD FILE
(
    NAME = test4dat4,
    FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\DATA\t4dat4.ndf',
    SIZE = 5MB,
    MAXSIZE = 100MB,
    FILEGROWTH = 5MB
)
    TO FILEGROUP test4fg;
GO
-- Creates a partition function called myRangePF1 that will partition a table into four partitions
CREATE PARTITION FUNCTION myRangePF1 (int)
    AS RANGE LEFT FOR VALUES (1, 100, 1000) ;
GO
-- Creates a partition scheme called myRangePS1 that applies myRangePF1 to the four filegroups created
above
CREATE PARTITION SCHEME myRangePS1
    AS PARTITION myRangePF1
    TO (test1fg, test2fg, test3fg, test4fg) ;
GO
-- Creates a partitioned table called PartitionTable that uses myRangePS1 to partition col1
CREATE TABLE PartitionTable (col1 int PRIMARY KEY, col2 char(10))
    ON myRangePS1 (col1) ;
GO

```

To determine if a table is partitioned

1. The following query returns one or more rows if the table `PartitionTable` is partitioned. If the table is not partitioned, no rows are returned.

```

SELECT *
FROM sys.tables AS t
JOIN sys.indexes AS i
    ON t.[object_id] = i.[object_id]
    AND i.[type] IN (0,1)
JOIN sys.partition_schemes ps
    ON i.data_space_id = ps.data_space_id
WHERE t.name = 'PartitionTable';
GO

```

To determine the boundary values for a partitioned table

1. The following query returns the boundary values for each partition in the `PartitionTable` table.

```

SELECT t.name AS TableName, i.name AS IndexName, p.partition_number, p.partition_id, i.data_space_id,
f.function_id, f.type_desc, r.boundary_id, r.value AS BoundaryValue
FROM sys.tables AS t
JOIN sys.indexes AS i
    ON t.object_id = i.object_id
JOIN sys.partitions AS p
    ON i.object_id = p.object_id AND i.index_id = p.index_id
JOIN sys.partition_schemes AS s
    ON i.data_space_id = s.data_space_id
JOIN sys.partition_functions AS f
    ON s.function_id = f.function_id
LEFT JOIN sys.partition_range_values AS r
    ON f.function_id = r.function_id and r.boundary_id = p.partition_number
WHERE t.name = 'PartitionTable' AND i.type <= 1
ORDER BY p.partition_number;

```

To determine the partition column for a partitioned table

1. The following query returns the name of the partitioning column for table. `PartitionTable`.

```

SELECT
    t.[object_id] AS ObjectID
    , t.name AS TableName
    , ic.column_id AS PartitioningColumnID
    , c.name AS PartitioningColumnName
FROM sys.tables AS t
JOIN sys.indexes AS i
    ON t.[object_id] = i.[object_id]
    AND i.[type] <= 1 -- clustered index or a heap
JOIN sys.partition_schemes AS ps
    ON ps.data_space_id = i.data_space_id
JOIN sys.index_columns AS ic
    ON ic.[object_id] = i.[object_id]
    AND ic.index_id = i.index_id
    AND ic.partition_ordinal >= 1 -- because 0 = non-partitioning column
JOIN sys.columns AS c
    ON t.[object_id] = c.[object_id]
    AND ic.column_id = c.column_id
WHERE t.name = 'PartitionTable' ;
GO

```

For more information, see:

- [ALTER DATABASE File and Filegroup Options \(Transact-SQL\)](#)
- [CREATE PARTITION FUNCTION \(Transact-SQL\)](#)
- [CREATE PARTITION SCHEME \(Transact-SQL\)](#)
- [CREATE TABLE \(Transact-SQL\)](#)

Modify a Partition Function

7/2/2018 • 4 minutes to read • [Edit Online](#)

THIS TOPIC APPLIES TO:  SQL Server  Azure SQL Database  Azure SQL Data Warehouse  Parallel Data Warehouse

You can change the way a table or index is partitioned in SQL Server 2017 by adding or subtracting the number of partitions specified, in increments of 1, in the partition function of the partitioned table or index by using Transact-SQL. When you add a partition, you do so by "splitting" an existing partition into two partitions and redefining the boundaries of the new partitions. When you drop a partition, you do so by "merging" the boundaries of two partitions into one. This last action repopulates one partition and leaves the other partition unassigned.

Caution

More than one table or index can use the same partition function. When you modify a partition function, you affect all of them in a single transaction. Check the partition function's dependencies before modifying it.

In This Topic

- **Before you begin:**

- [Limitations and Restrictions](#)

- [Security](#)

- **To modify a partition function, using:**

- [SQL Server Management Studio](#)

- [Transact-SQL](#)

Before You Begin

Limitations and Restrictions

- ALTER PARTITION FUNCTION can only be used for splitting one partition into two, or for merging two partitions into one. To change the way a table or index is partitioned (from 10 partitions to 5, for example), you can use any one of the following options:
 - Create a new partitioned table with the desired partition function, and then insert the data from the old table into the new table by using either an INSERT INTO ... SELECT FROM Transact-SQL statement or the **Manage Partition Wizard** in SQL Server Management Studio.
 - Create a partitioned clustered index on a heap.

NOTE

Dropping a partitioned clustered index results in a partitioned heap.

- Drop and rebuild an existing partitioned index by using the Transact-SQL CREATE INDEX statement with the DROP EXISTING = ON clause.
 - Perform a sequence of ALTER PARTITION FUNCTION statements.
- SQL Server does not provide replication support for modifying a partition function. If you want to make changes to a partition function in the publication database, you must do this manually in the subscription database.

- All filegroups that are affected by ALTER PARTITION FUNCTION must be online.

Security

Permissions

Any one of the following permissions can be used to execute ALTER PARTITION FUNCTION:

- ALTER ANY DATASPACE permission. This permission defaults to members of the **sysadmin** fixed server role and the **db_owner** and **db_ddladmin** fixed database roles.
- CONTROL or ALTER permission on the database in which the partition function was created.
- CONTROL SERVER or ALTER ANY DATABASE permission on the server of the database in which the partition function was created.

Using SQL Server Management Studio

To modify a partition function:

This specific action cannot be performed using SQL Server Management Studio. In order to modify a partition function, you must first delete the function and then create a new one with the desired properties using the Create Partition Wizard. For more information, see

To delete a partition function

1. Expand the database where you want to delete the partition function and then expand the **Storage** folder.
2. Expand the **Partition Functions** folder.
3. Right-click the partition function you want to delete and select **Delete**.
4. In the **Delete Object** dialog box, ensure that the correct partition function is selected, and then click **OK**.

Using Transact-SQL

To split a single partition into two partitions

1. In **Object Explorer**, connect to an instance of Database Engine.
2. On the Standard bar, click **New Query**.
3. Copy and paste the following example into the query window and click **Execute**.

```
-- Look for a previous version of the partition function "myRangePF1" and deletes it if it is found.
IF EXISTS (SELECT * FROM sys.partition_functions
           WHERE name = 'myRangePF1')
    DROP PARTITION FUNCTION myRangePF1;
GO

-- Create a new partition function called "myRangePF1" that partitions a table into four partitions.
CREATE PARTITION FUNCTION myRangePF1 (int)
AS RANGE LEFT FOR VALUES ( 1, 100, 1000 );
GO

--Split the partition between boundary_values 100 and 1000
--to create two partitions between boundary_values 100 and 500
--and between boundary_values 500 and 1000.
ALTER PARTITION FUNCTION myRangePF1 ( )
SPLIT RANGE (500);
```

To merge two partitions into one partition





1. In **Object Explorer**, connect to an instance of Database Engine.
2. On the Standard bar, click **New Query**.
3. Copy and paste the following example into the query window and click **Execute**.

```
-- Look for a previous version of the partition function "myRangePF1" and deletes it if it is found.
IF EXISTS (SELECT * FROM sys.partition_functions
           WHERE name = 'myRangePF1')
    DROP PARTITION FUNCTION myRangePF1;
GO
-- Create a new partition function called "myRangePF1" that partitions a table into four partitions.
CREATE PARTITION FUNCTION myRangePF1 (int)
AS RANGE LEFT FOR VALUES ( 1, 100, 1000 );
GO
--Merge the partitions between boundary_values 1 and 100
--and between boundary_values 100 and 1000 to create one partition
--between boundary_values 1 and 1000.
ALTER PARTITION FUNCTION myRangePF1 ( )
MERGE RANGE (100);
```

For more information, see [ALTER PARTITION FUNCTION \(Transact-SQL\)](#).

Modify a Partition Scheme

7/2/2018 • 3 minutes to read • [Edit Online](#)

THIS TOPIC APPLIES TO:  SQL Server  Azure SQL Database  Azure SQL Data Warehouse  Parallel Data Warehouse

You can modify a partition scheme in SQL Server 2017 by designating a filegroup to hold the next partition that is added to a partitioned table using SQL Server Management Studio or Transact-SQL. You do this by assigning the NEXT USED property to a filegroup. You can assign the NEXT USED property to an empty filegroup or to one that already holds a partition. In other words, a filegroup can hold more than one partition.

In This Topic

- **Before you begin:**

- [Limitations and Restrictions](#)

- [Security](#)

- **To create a partitioned table or index, using:**

- [SQL Server Management Studio](#)

- [Transact-SQL](#)

Before You Begin

Limitations and Restrictions

Any filegroup affected by ALTER PARTITION SCHEME must be online.

Security

Permissions

The following permissions can be used to execute ALTER PARTITION SCHEME:

- ALTER ANY DATASPACE permission. This permission defaults to members of the **sysadmin** fixed server role and the **db_owner** and **db_ddladmin** fixed database roles.
- CONTROL or ALTER permission on the database in which the partition scheme was created.
- CONTROL SERVER or ALTER ANY DATABASE permission on the server of the database in which the partition scheme was created.

Using SQL Server Management Studio

To modify a partition scheme:

This specific action cannot be performed using SQL Server Management Studio. In order to modify a partition scheme, you must first delete the scheme and then create a new one with the desired properties using the Create Partition Wizard. For more information, see [Create Partitioned Tables and Indexes Using SQL Server Management Studio](#) under **Create Partitioned Tables and Indexes**.

To delete a partition scheme

1. Click the plus sign to expand the database where you want to delete the partition scheme.
2. Click the plus sign to expand the **Storage** folder.

3. Click the plus sign to expand the **Partition Schemes** folder.
4. Right-click the partition scheme you want to delete and select **Delete**.
5. In the **Delete Object** dialog box, ensure that the correct partition scheme is selected, and then click **OK**.

Using Transact-SQL

To modify a partition scheme

1. In **Object Explorer**, connect to an instance of Database Engine.
2. On the Standard bar, click **New Query**.
3. Copy and paste the following example into the query window and click **Execute**.

```

USE AdventureWorks2012;
GO
-- add five new filegroups to the AdventureWorks2012 database
ALTER DATABASE AdventureWorks2012
ADD FILEGROUP test1fg;
GO
ALTER DATABASE AdventureWorks2012
ADD FILEGROUP test2fg;
GO
ALTER DATABASE AdventureWorks2012
ADD FILEGROUP test3fg;
GO
ALTER DATABASE AdventureWorks2012
ADD FILEGROUP test4fg;
GO
ALTER DATABASE AdventureWorks2012
ADD FILEGROUP test5fg;
GO
-- if the "myRangePF1" partition function and the "myRangePS1" partition scheme exist,
-- drop them from the AdventureWorks2012 database
IF EXISTS (SELECT * FROM sys.partition_functions
WHERE name = 'myRangePF1')
DROP PARTITION FUNCTION myRangePF1;
GO
IF EXISTS (SELECT * FROM sys.partition_schemes
WHERE name = 'myRangePS1')
DROP PARTITION SCHEME myRangePS1;
GO
-- create the new partition function "myRangePF1" with four partition groups
CREATE PARTITION FUNCTION myRangePF1 (int)
AS RANGE LEFT FOR VALUES ( 1, 100, 1000 );
GO
-- create the new partition scheme "myRangePS1" that will use
-- the "myRangePF1" partition function with five file groups.
-- The last filegroup, "test5fg," will be kept empty but marked
-- as the next used filegroup in the partition scheme.
CREATE PARTITION SCHEME myRangePS1
AS PARTITION myRangePF1
TO (test1fg, test2fg, test3fg, test4fg, test5fg);
GO
--Split "myRangePS1" between boundary_values 100 and 1000
--to create two partitions between boundary_values 100 and 500
--and between boundary_values 500 and 1000.
ALTER PARTITION FUNCTION myRangePF1 ( )
SPLIT RANGE (500);
GO
-- Allow the "myRangePS1" partition scheme to use the filegroup "test5fg"
-- for the partition with boundary_values of 100 and 500
ALTER PARTITION SCHEME myRangePS1
NEXT USED test5fg;
GO

```

For more information, see [ALTER PARTITION SCHEME \(Transact-SQL\)](#).

Manage Partition Wizard F1 Help

7/2/2018 • 8 minutes to read • [Edit Online](#)

THIS TOPIC APPLIES TO:  SQL Server  Azure SQL Database  Azure SQL Data Warehouse  Parallel Data Warehouse

Use the **Manage Partition Wizard** to manage and modify existing partitioned tables through partition switching or the implementation of a sliding window scenario. This wizard can ease the management of your partitions and simplify the regular migration of data in and out of your tables.

To start the Manage Partition Wizard

- In SQL Server Management Studio, select the database, right-click the table on which you want to create partitions, point to **Storage**, and then click **Manage Partition**.

Note If **Manage Partition** is unavailable, you may have selected a table that does not contain partitions. Click **Create Partition** on the **Storage** submenu and use the **Create Partition Wizard** to create partitions in your table.

For general information about partitions and indexes, see [Partitioned Tables and Indexes](#).

This section provides the information that is required to manage, modify, and implement partitions using the **Manage Partition Wizard**.

In This Section

The following sections provide help on the pages of the **Manage Partition Wizard**.

[Manage Partition Wizard \(Select Partition Action Page\)](#)

[Manage Partition Wizard \(Switch In Page\)](#)

[Manage Partition Wizard \(Switch Out Page\)](#)

[Manage Partition Wizard \(Select Staging Table Options Page\)](#)

[Manage Partition Wizard \(Select Output Option Page\)](#)

[Manage Partition Wizard \(New Job Schedule Page\)](#)

[Manage Partition Wizard \(Summary Page\)](#)

[Manage Partition Wizard \(Progress Page\)](#)

Select Partition Action Page

Use the **Select Partition Action** page to choose the action you want to perform on your partition.

Create a Staging Table

Partition switching is a common partitioning task if you have a partitioned table that you migrate data into and out of on a regular basis; for example, you have a partitioned table that stores current quarterly data, and you must move in new data and archive older data at the end of each quarter.

The wizard designs the staging table with the same partitioning column, table and column structure, and indexes, and stores the new table in the filegroup where your source partition is located.

To create a staging table to switch in or switch out partition data, select **Create a staging table for partition**

switching.

Sliding Window Scenario

To manage your partitions in a sliding-window scenario, select **Manage partitioned data in a sliding window scenario**.

UIElement List

Create a staging table for partition switching

Creates a staging table for the data you are switching in or switching out of the existing partitioned table.

Switch out partition

Provides options when removing a partition from the table.

Switch in partition

Provides options when adding a partition to the table.

Manage partitioned data in a sliding window scenario

Appends an empty partition to the existing table that can be used for switching in data. The wizard currently supports switching into the last partition and switching out the first partition.

 [In This Section](#)

Select Partition Switching-In Options Page

Use the **Select Partition Switching-In options** page to select the staging table you are switching into the partitioned table.

UIElement List

Show All Partitions

Select to show all partitions, including the partitions currently in the partitioned table.

Partition grid

Displays the partition name, **Left boundary**, **Right boundary**, **Filegroup**, and **Row count** of the partitions you selected.

Switch in table

Select the staging table that contains the partition that you want to add to your partitioned table. You must create this staging table before you switch-in partitions with the **Manage PartitionsWizard**.

 [In This Section](#)

Select Partition Switching-Out Options Page

Use the **Select Partition Switching-Out options** page to select the partition and the staging table to hold the partitioned data that you are switching out of the partitioned table.

UIElement List

Partition grid

Displays the partition name, **Left boundary**, **Right boundary**, **Filegroup**, and **Row count** of the partitions you selected.

Switch out table

Choose a new table or an existing table to switch-out your data to.

New

Enter a new name for the staging table you want to use for the partition to switch out of the current source table.

Existing

Select an existing staging table you want to use for the partition you want to switch out of the current source table. If the existing table contains data, this data will be overwritten with the data you are switching out.

 [In This Section](#)

Select the Staging Table Options Page

Use the **Select the Staging Table Options** page to create the staging table you want to use for switching your partitioned data.

Staging tables must reside in the same filegroup of the selected partition where the source table is located. The staging table must mirror the design of both the source table and the destination table.

You can also create the same indexes in the staging table that exist in the source partition. The staging table automatically contains a constraint based on the elements of the source partition. This constraint is typically generated from the boundary value of the source partition.

UIElement List

Staging table name

Create a name for the staging table or accept the default name displayed in the edit box.

Switch partition

Select the source partition that you want to switch out of the current table.

New boundary value

Select or enter the boundary value you want for the partition in the staging table.

Filegroup

Select a filegroup for the new table.

 [In This Section](#)

Select Output Option Page

Use the **Select Output Option** page to specify how you want to complete the modifications to your partitions.

Create Script

When the wizard finishes, it creates a script in Query Editor to modify partitions in the table. Select **Create Script** if you want to review the script, and then execute the script manually.

Script to file

Generate the script to a .sql file. Specify either **Unicode** or **ANSI text**. To specify a name and location for the file, click **Browse**.

Script to Clipboard

Save the script to the Clipboard.

Script to New Query Window

Generate the script to a Query Editor window. If no editor window is open, a new editor window opens as the target for the script.

Run Immediately

Run immediately

Have the wizard finish modifications to the partitions when you click **Next** or **Finish**.

Schedule

Select to modify the table partitions at a scheduled date and time.

Change schedule

Opens the **New Job Schedule** dialog box, where you can select, change, or view the properties of the scheduled job.

 [In This Section](#)

New Job Schedule Page

Use the **New Job Schedule** page to view and change the properties of the schedule.

Options

Select the type of schedule you want for the SQL Server Agent job.

Name

Type a new name for the schedule.

Jobs in schedule

View the existing jobs that use the schedule.

Schedule type

Select the type of schedule.

Enabled

Enable or disable the schedule.

Recurring Schedule Types Options

Select the frequency of the scheduled job.

Occurs

Select the interval at which the schedule recurs.

Recurs every

Select the number of days or weeks between recurrences of the schedule. This option is not available for monthly schedules.

Monday

Set the job to occur on a Monday. Only available for weekly schedules.

Tuesday

Set the job to occur on a Tuesday. Only available for weekly schedules.

Wednesday

Set the job to occur on a Wednesday. Only available for weekly schedules.

Thursday

Set the job to occur on a Thursday. Only available for weekly schedules.

Friday

Set the job to occur on a Friday. Only available for weekly schedules.

Saturday

Set the job to occur on a Saturday. Only available for weekly schedules.

Sunday

Set the job to occur on a Sunday. Only available for weekly schedules.

Day

Select the day of the month the schedule occurs. Only available for monthly schedules.

of every

Select the number of months between occurrences of the schedule. Only available for monthly schedules.

The

Specify a schedule for a specific day of the week on a specific week within the month. Only available for monthly schedules.

Occurs once at

Set the time for a job to occur daily.

Occurs every

Set the number of hours or minutes between occurrences.

Start date

Set the date when this schedule will become effective.

End date

Set the date when the schedule will no longer be effective.

No end date

Specify that the schedule will remain effective indefinitely.

One Time Schedule Types Options

If you schedule a job to run once, you must select a date and time in the future.

Date

Select the date for the job to run.

Time

Select the time for the job to run.

 [In This Section](#)

Summary Page

Use the **Summary** page to review the options that you have selected on the previous pages.

UIElement List

Review your selections

Displays the selections you have made for each page of the wizard. Click a node to expand and view your previously selected options.

 [In This Section](#)

Progress Page

Use the **Progress** page to monitor status information about the actions of the **Manage Partition Wizard**.

Depending on the options that you selected in the wizard, the **Progress** page might contain one or more actions.

The top box displays the overall status of the wizard and the number of status, error, and warning messages that the wizard has received.

Options

Details

Provides the action, status, and any messages that are returned from action taken by the wizard.

Action

Specifies the type and name of each action.

Status

Indicates whether the wizard action as a whole returned the value of **Success** or **Failure**.

Message

Provides any error or warning messages that are returned from the process.

Stop

Stop the action of the wizard.

Report

Create a report that contains the results of the **Manage Partition Wizard**. The options are:

- **View Report**
- **Save Report to File**
- **Copy Report to Clipboard**
- **Send Report as Email**

View Report

Open the **View Report** dialog box. This dialog box contains a text report of the progress of the **Manage Partition Wizard**.

Close

Close the wizard.

 [In This Section](#)

See Also

[Partitioned Tables and Indexes](#)