Create Partitioned Tables and Indexes

SQL Server 2016 and later

Applies To: SQL Server 2016

You can create a partitioned table or index in SQL Server 2016 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

- 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**:

  i. 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".

  ii. 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.

  iii. 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
)
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)

Community Additions