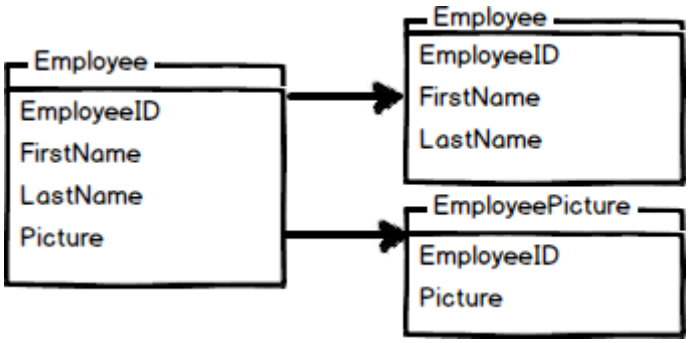
¿Qué es el particionamiento de tablas de bases de datos?

Particionar es el proceso donde tablas muy grandes son divididas en múltiples partes más pequeñas. Al separar una tabla grande en tablas individuales más pequeñas, las consultas que acceden sólo a una fracción de los datos pueden correr más rápido porque hay menos datos que escanear. El objetivo principal de particionar es ayudar en el mantenimiento de tablas grandes y reducir el tiempo de respuesta general para leer y cargar datos para operaciones SQL particulares.

Particionamiento Vertical en tablas SQL Server

El particionamiento vertical de tablas es principalmente usado para incrementar el desempeño de SQL Server especialmente en casos cuando una consulta retorna todas las columnas de una tabla que contiene un número de columnas de texto muy amplio o BLOB. En este caso, para reducir los tiempos de acceso, las columnas BLOB pueden ser divididas a su propia tabla. Otro ejemplo es restringir el acceso a datos sensibles, por ejemplo, contraseñas, información salarial, etc. La partición vertical divide una tabla en dos o más tablas que contienen diferentes columnas:



Ejemplo de particionamiento vertical

Un ejemplo de particionamiento vertical puede ser una tabla grande con reportes para empleados que contiene información básica, como el nombre del reporte, el id, el número de reporte y una gran columna con la descripción del reporte. Asumiremos que un ~95% de los usuarios están buscando en la parte del nombre del reporte, el número, etc., y que sólo un ~5% de las peticiones están abriendo el campo de las descripciones de los reportes y viendo la descripción. Asumamos que todas esas búsquedas conducirán a los escaneos del índice agrupado y dado que los escaneos del índice leen todas las filas en la tabla el costo de la consulta es proporcional al número total de filas en la tabla y nuestro objetivo es minimizar el número de operaciones IO y reducir el costo de la búsqueda.

Veamos el ejemplo de la tabla `EmployeeReports` table:

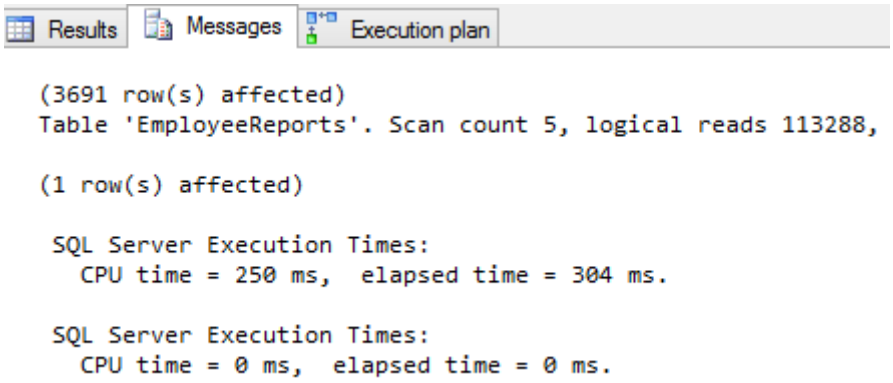
```
CREATE TABLE EmployeeReports
(
```

```
ReportID int IDENTITY (1,1) NOT NULL,

ReportName varchar (100),

ReportNumber varchar (20),

ReportDescription varchar (max)

CONSTRAINT EReport_PK PRIMARY KEY CLUSTERED (ReportID)

)


DECLARE @i int

SET @i = 1


BEGIN TRAN

WHILE @i<100000

BEGIN

INSERT INTO EmployeeReports

(

ReportName,

ReportNumber,

ReportDescription

)

VALUES

(

'ReportName',

CONVERT (varchar (20), @i),

REPLICATE ('Report', 1000)

)

SET @i=@i+1

END

COMMIT TRAN

GO
```

Si corremos una consulta SQL para recuperar los datos de `ReportID`, `ReportName`, `ReportNumber` de la tabla `EmployeeReports` el resultado establece que la cuenta del escaneo es 5 y representa un número de veces que la tabla ha

sido accedida durante la consulta, y que teníamos 113288 lecturas lógicas que representan el número total de accesos de páginas para procesar la consulta:

```
SET STATISTICS IO ON

SET STATISTICS TIME ON

SELECT er.ReportID, er.ReportName, er.ReportNumber

FROM dbo.EmployeeReports er

WHERE er.ReportNumber LIKE '%33%'

SET STATISTICS IO OFF

SET STATISTICS TIME OFF
```



Como se indicó, cada página es leída desde el caché de datos, sea o no necesario traer la página desde el disco al caché para cualquier lectura dada. Para reducir el costo de la consulta cambiaremos el esquema de base de datos de SQL Server y dividiremos la tabla `EmployeeReports` verticalmente.

A continuación crearemos la tabla `ReportsDesc` y moveremos la columna `ReportDescription` y la tabla`ReportsData` y moveremos todos los datos de la tabla `EmployeeReports` excepto la columna`ReportDescription`.

```
CREATE TABLE ReportsDesc

( ReportID int FOREIGN KEY REFERENCES EmployeeReports (ReportID),

 ReportDescription varchar(max)

 CONSTRAINT PK_ReportDesc PRIMARY KEY CLUSTERED (ReportID)

 )


CREATE TABLE ReportsData

(

ReportID int NOT NULL,
```

```
ReportName varchar (100),
ReportNumber varchar (20),

CONSTRAINT DReport_PK PRIMARY KEY CLUSTERED (ReportID)
)
INSERT INTO dbo.ReportsData
(
   ReportID,
   ReportName,
   ReportNumber
)
SELECT er.ReportID,
er.ReportName,
er.ReportNumber
FROM dbo.EmployeeReports er
```

La misma consulta de búsqueda ahora dará diferentes resultados:

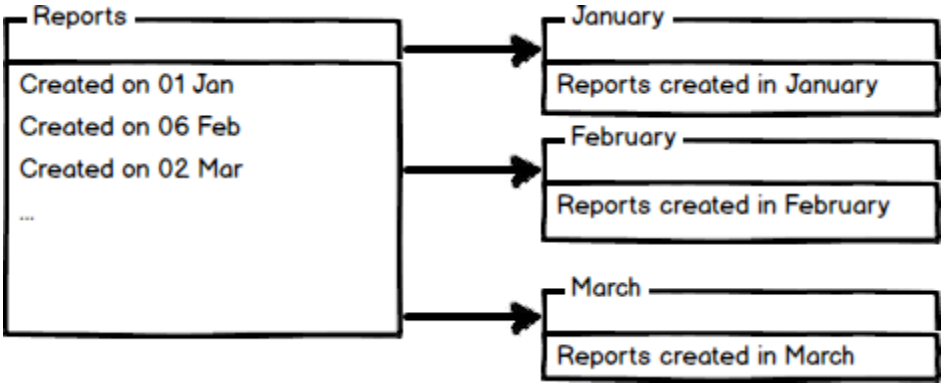
```
SET STATISTICS IO ON
SET STATISTICS TIME ON
SELECT er.ReportID, er.ReportName, er.ReportNumber
FROM ReportsData er
WHERE er.ReportNumber LIKE '%33%'
SET STATISTICS IO OFF
SET STATISTICS TIME OFF
```

```
(3691 row(s) affected)
Table 'ReportsData'. Scan count 1, logical reads 421,

(1 row(s) affected)

 SQL Server Execution Times:
   CPU time = 31 ms,  elapsed time = 104 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
```

La partición vertical en tablas SQL Server puede no ser el método correcto en cada caso. Sin embargo, si usted tiene, por ejemplo, una tabla con muchos datos que no se accede igualitariamente, tablas con datos a los que desea restringir el acceso, o escaneos que retornan muchos datos, la partición vertical puede ayudar.

Particionamiento Horizontal en tablas SQL Server

El particionamiento horizontal divide una tabla en múltiples tablas que contienen el mismo número de columnas, pero menos filas. Por ejemplo, si una tabla contiene un gran número de filas que representan reportes mensuales podría ser particionada horizontalmente en tablas por años, con cada tabla representando todos los reportes para un año específico. De esta manera las consultas que requieren datos para un año específico sólo referenciarán la tabla apropiada. Las tablas deberían ser particionadas en una manera que las consultas referencian tan pocas tablas como sea posible.



Las tablas son particionadas horizontalmente basadas en una columna que será usada para particionar y los rangos asociados a cada partición. La columna de particionamiento es usualmente una columna de fecha pero todos los tipos de datos que son válidos para usarse como columnas de índice pueden ser usados como columna de partición, excepto columnas timestamp. Los siguientes tipos de datos no pueden ser especificados: ntext, text, image, xml, varchar(max), nvarchar(max), o varbinary(max), el tipo definido por el usuario Microsoft .NET Framework common language runtime (CLR), columnas de tipo de datos de alias.

Hay dos enfoques diferentes que podríamos usar para lograr la partición de la tabla. El primero es crear una nueva tabla particionada y simplemente copiar los datos desde su tabla existente en la nueva tabla y renombrarla. El segundo enfoque es particionar una tabla existente reconstruyendo o creando un índice agrupado en la tabla.

Ejemplo de particionamiento horizontal con la creación de una nueva tabla particionada

SQL Server 2005 introdujo una característica incorporada de particionamiento para particionar horizontalmente una tabla con hasta 1000 particiones en SQL Server 2008 y 15000 particiones en SQL Server 2012, y el emplazamiento de los datos es manejado automáticamente por SQL Server. Esta característica está disponible sólo en la Edición Enterprise de SQL Server.

Para crear una tabla particionada para almacenar reportes mensuales primero crearemos grupos de archivos adicionales. Un grupo de archivos es una unidad de almacenamiento lógica. Cada base de datos tiene un grupo de archivos que contiene el archivo de datos

primario (.mdf). Un grupo de archivos adicional y definido por el usuario puede ser creado para contener archivos secundarios (.ndf). Nosotros crearemos 12 grupos de archivos por cada mes:

```
ALTER DATABASE PartitioningDB
ADD FILEGROUP January
GO
ALTER DATABASE PartitioningDB
ADD FILEGROUP February
GO
ALTER DATABASE PartitioningDB
ADD FILEGROUP March
GO
ALTER DATABASE PartitioningDB
ADD FILEGROUP April
GO
ALTER DATABASE PartitioningDB
ADD FILEGROUP May
GO
ALTER DATABASE PartitioningDB
ADD FILEGROUP June
GO
ALTER DATABASE PartitioningDB
ADD FILEGROUP July
GO
ALTER DATABASE PartitioningDB
ADD FILEGROUP Avgust
GO
ALTER DATABASE PartitioningDB
ADD FILEGROUP September
GO
ALTER DATABASE PartitioningDB
ADD FILEGROUP October
GO
```

```
ALTER DATABASE PartitioningDB

ADD FILEGROUP November

GO

ALTER DATABASE PartitioningDB

ADD FILEGROUP December

GO
```

Para verificar los grupos de archivos creados y disponibles en la base de datos actual ejecute la siguiente consulta:

```
SELECT name AS AvailableFilegroups

FROM sys.filegroups

WHERE type = 'FG'
```

	AvailableFilegroups
1	PRIMARY
2	January
3	February
4	March
5	April
6	May
7	June
8	July
9	Avgust
10	September
11	October
12	November
13	December

Cuando son creados grupos de archivos añadiremos el archivo .ndf a cada grupo de archivos:

```
ALTER DATABASE [PartitioningDB]

   ADD FILE

   (

   NAME = [PartJan],

   FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL11.LENOVO\MSSQL\DATA\PartitioningDB.nd

f',
```

```
    SIZE = 3072 KB,
    MAXSIZE = UNLIMITED,
    FILEGROWTH = 1024 KB
  ) TO FILEGROUP [January]
```

De la misma manera los archivos a todos los grupos de archivos creados especificando el nombre lógico del archivo y el nombre del archivo del sistema operativo (físico) para cada grupo de archivos. Por ejemplo:

```
ALTER DATABASE [PartitioningDB]
  ADD FILE
  (
  NAME = [PartFeb],
  FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL11.LENOVO\MSSQL\DATA\PartitioningDB2.n
df',
    SIZE = 3072 KB,
    MAXSIZE = UNLIMITED,
    FILEGROWTH = 1024 KB
  ) TO FILEGROUP [February]
```

Para verificar archivos creado añadidos a los grupos de archivos ejecute la siguiente consulta:

```
SELECT
name as [FileName],
physical_name as [FilePath]
FROM sys.database_files
where type_desc = 'ROWS'
GO
```

	FileName	FilePath
1	PartitioningDB	C:\Program Files\Microsoft SQL Server\MSSQL11.LENOVO\MSSQL\DATA\PartitioningDB.mdf
2	PartJan	C:\Program Files\Microsoft SQL Server\MSSQL11.LENOVO\MSSQL\DATA\PartitioningDB.ndf
3	PartFeb	C:\Program Files\Microsoft SQL Server\MSSQL11.LENOVO\MSSQL\DATA\PartitioningDB2.ndf
4	PartMar	C:\Program Files\Microsoft SQL Server\MSSQL11.LENOVO\MSSQL\DATA\PartitioningDB3.ndf
5	PartApr	C:\Program Files\Microsoft SQL Server\MSSQL11.LENOVO\MSSQL\DATA\PartitioningDB4.ndf
6	PartMay	C:\Program Files\Microsoft SQL Server\MSSQL11.LENOVO\MSSQL\DATA\PartitioningDB5.ndf
7	PartJul	C:\Program Files\Microsoft SQL Server\MSSQL11.LENOVO\MSSQL\DATA\PartitioningDB7.ndf
8	PartAvg	C:\Program Files\Microsoft SQL Server\MSSQL11.LENOVO\MSSQL\DATA\PartitioningDB8.ndf
9	PartSep	C:\Program Files\Microsoft SQL Server\MSSQL11.LENOVO\MSSQL\DATA\PartitioningDB9.ndf
10	PartOct	C:\Program Files\Microsoft SQL Server\MSSQL11.LENOVO\MSSQL\DATA\PartitioningDB10.ndf
11	PartNov	C:\Program Files\Microsoft SQL Server\MSSQL11.LENOVO\MSSQL\DATA\PartitioningDB11.ndf
12	PartDec	C:\Program Files\Microsoft SQL Server\MSSQL11.LENOVO\MSSQL\DATA\PartitioningDB12.ndf
13	PartJune	C:\Program Files\Microsoft SQL Server\MSSQL11.LENOVO\MSSQL\DATA\PartitioningDB6.ndf

Después de crear grupos de archivos adicionales para almacenar datos crearemos una función de partición. Una función de partición es una función que mapea las filas de una tabla particionada en particiones basada en los valores de una columna de partición. En este ejemplo crearemos una función de partición que particiona una tabla en 12 particiones, una por cada mes de los valores de un año en una columna de fecha:

```
CREATE PARTITION FUNCTION [PartitioningByMonth] (datetime)
AS RANGE RIGHT FOR VALUES ('20140201', '20140301', '20140401',
        '20140501', '20140601', '20140701', '20140801',
        '20140901', '20141001', '20141101', '20141201');
```

Para mapear las particiones de una tabla particionada a grupos de archivos y determinar el número y dominio de las particiones de una tabla crearemos un esquema de partición:

```
CREATE PARTITION SCHEME PartitionBymonth
AS PARTITION PartitioningBymonth
TO (January, February, March,
   April, May, June, July,
   Avgust, September, October,
   November, December);
```

Ahora vamos a crear la tabla usando el esquema de partición `PartitionBymonth`, y la llenaremos con datos de prueba:

```
CREATE TABLE Reports
(ReportDate datetime PRIMARY KEY,
MonthlyReport varchar(max))
ON PartitionBymonth (ReportDate);
```

```
GO

INSERT INTO Reports (ReportDate,MonthlyReport)
SELECT '20140105', 'ReportJanuary' UNION ALL
SELECT '20140205', 'ReportFebryary' UNION ALL
SELECT '20140308', 'ReportMarch' UNION ALL
SELECT '20140409', 'ReportApril' UNION ALL
SELECT '20140509', 'ReportMay' UNION ALL
SELECT '20140609', 'ReportJune' UNION ALL
SELECT '20140709', 'ReportJuly' UNION ALL
SELECT '20140809', 'ReportAugust' UNION ALL
SELECT '20140909', 'ReportSeptember' UNION ALL
SELECT '20141009', 'ReportOctober' UNION ALL
SELECT '20141109', 'ReportNovember' UNION ALL
SELECT '20141209', 'ReportDecember'
```

Ahora verificaremos las filas en las diferentes particiones:

```
SELECT
p.partition_number AS PartitionNumber,
f.name AS PartitionFilegroup,
p.rows AS NumberOfRows
FROM sys.partitions p
JOIN sys.destination_data_spaces dds ON p.partition_number = dds.destination_id
JOIN sys.filegroups f ON dds.data_space_id = f.data_space_id
WHERE OBJECT_NAME(OBJECT_ID) = 'Reports'
```

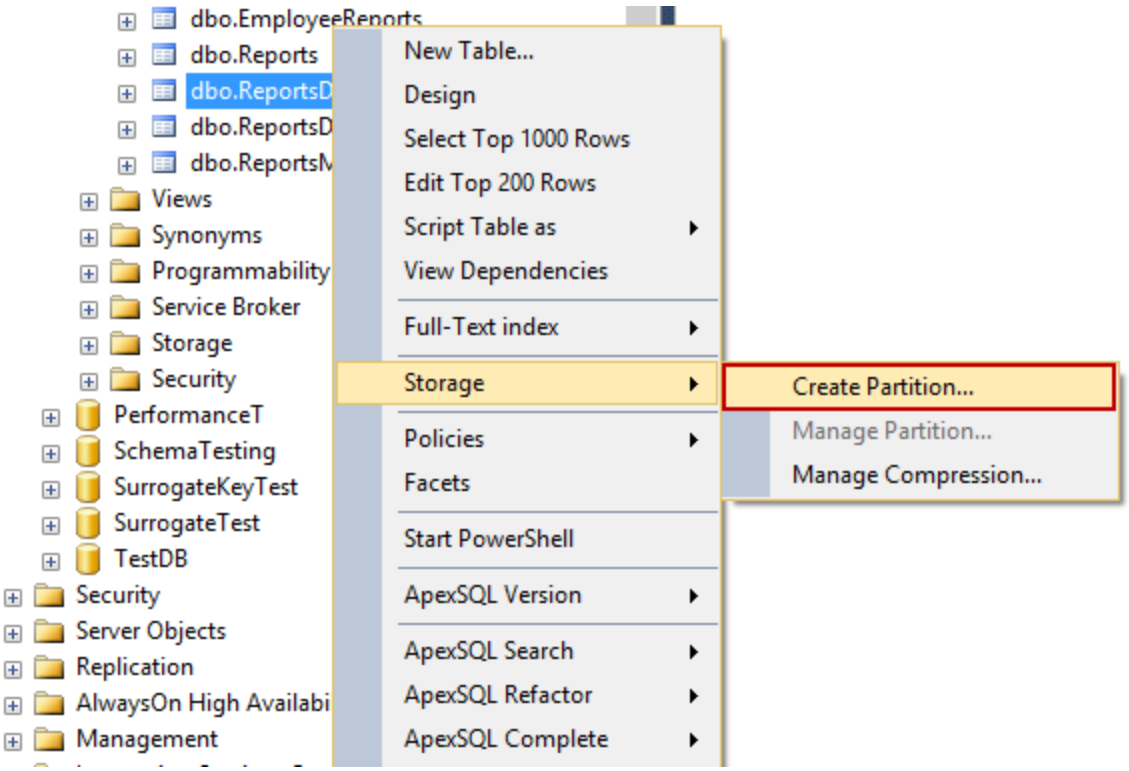
PartitionNumber	PartitionFilegroup	NumberOfRows
1	January	1
2	February	1
3	March	1
4	April	1
5	May	1
6	June	1
7	July	1
8	Avgust	1
9	September	1
10	October	1
11	November	1
12	December	1

Ahora simplemente copie los datos desde su tabla y renombre una tabla particionada.

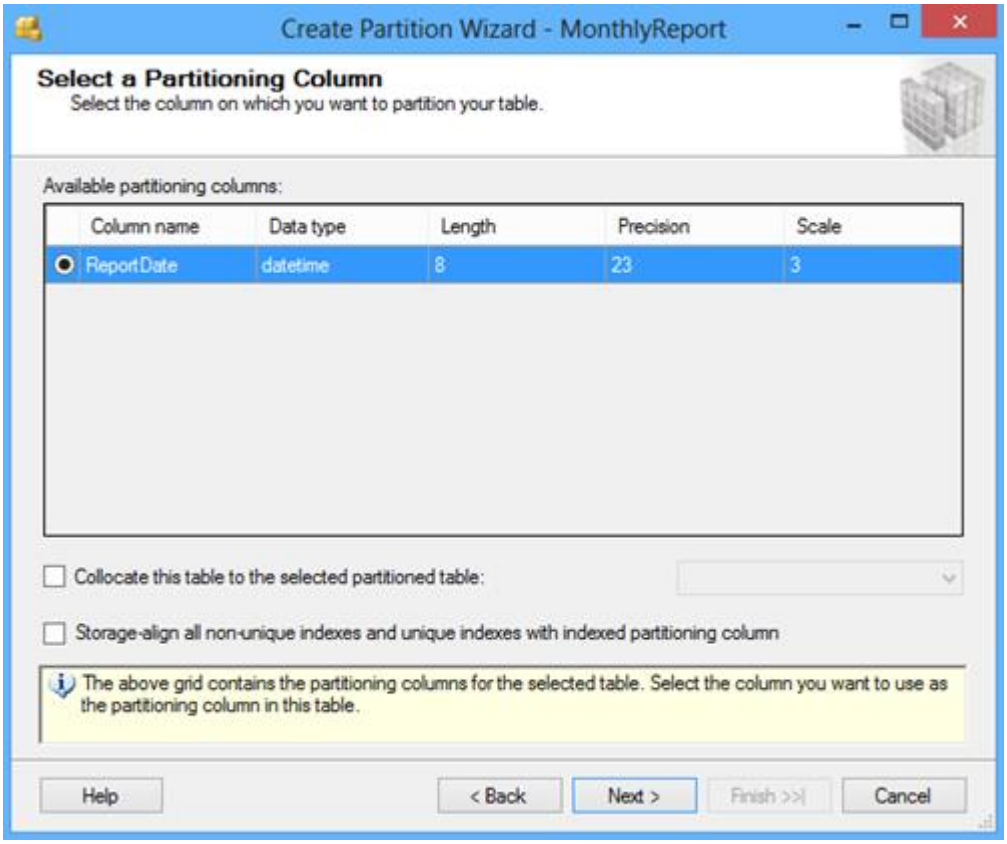
Particionar una tabla usando el asistente de Partición de SQL Server Management Studio

SQL Server 2008 introdujo un asistente de particionamiento de tablas en SQL Server Management Studio.

Haga clic derecho en una tabla en el panel **Object Explorer** y en el menú contextual **Storage** elija el comando **Create Partition**:

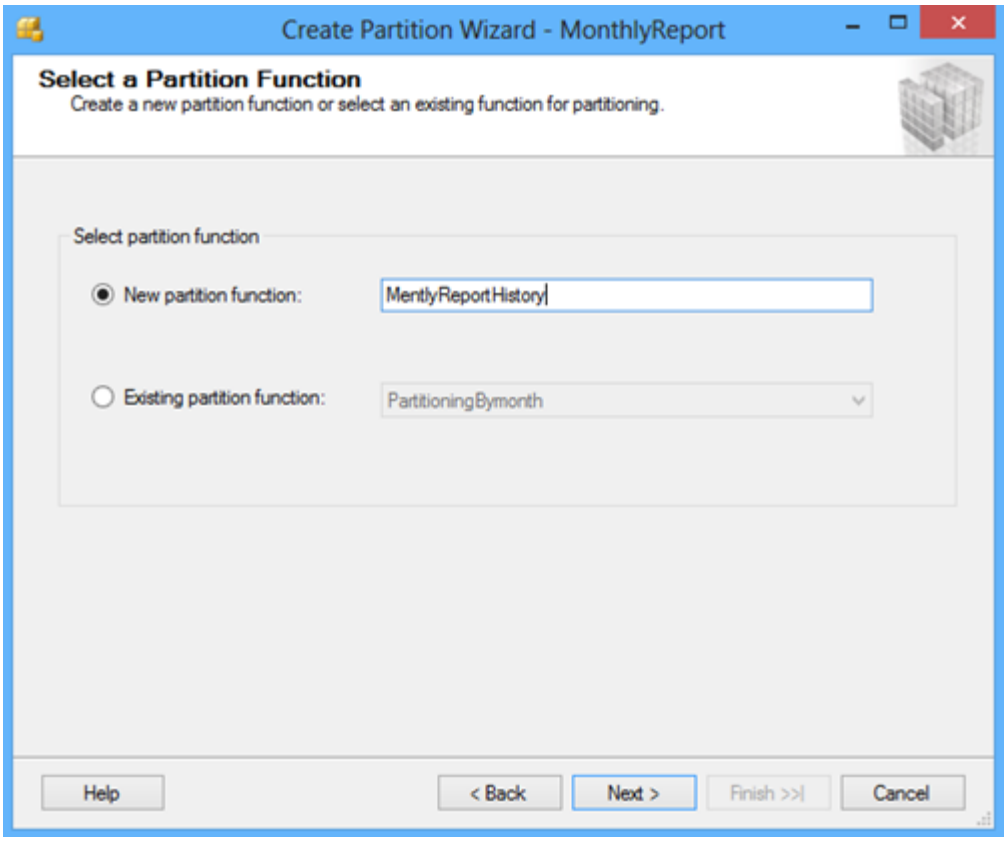


En la ventana **Select a Partitioning Column**, seleccione una columna que será usada para particionar una tabla desde las columnas de partición disponibles:

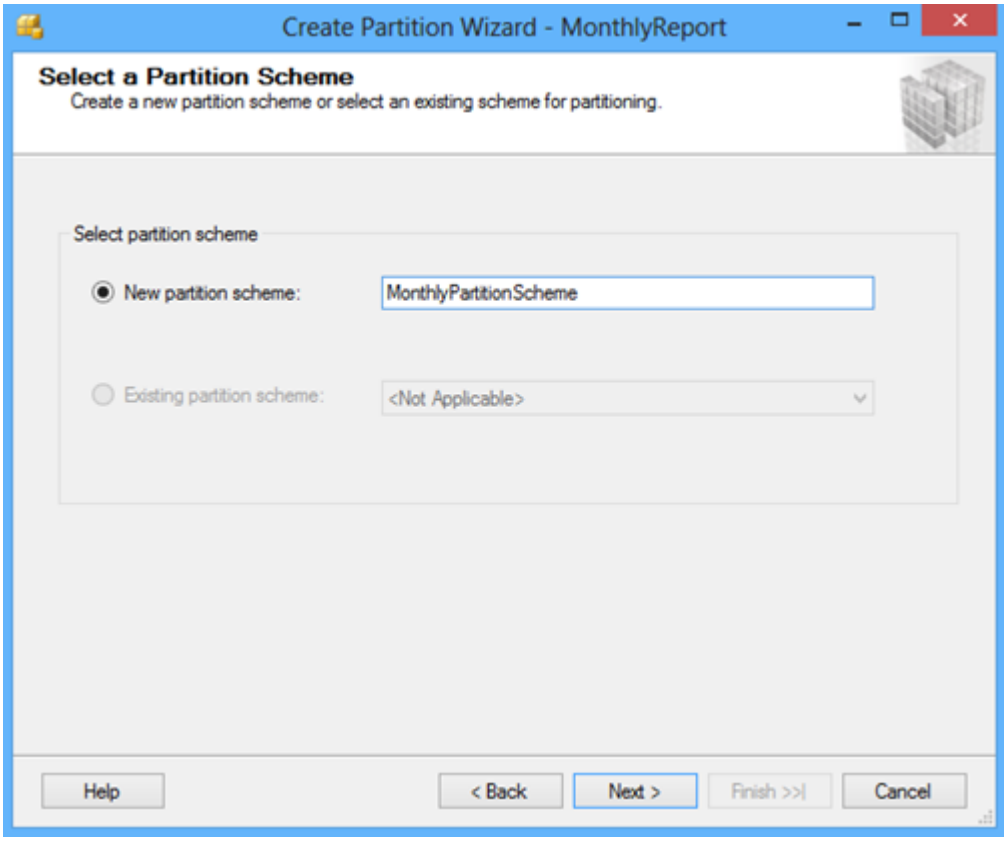


Otras opciones en el diálogo **Create Partition Wizard** incluyen la opción **Collocate this table to the selected partition table**, usada para mostrar datos relacionados para unirlos con la columna particionada y la opción **Storage Align Non Unique Indexes** and **Unique Indexes with an Indexed Partition Column**, que alinea todos los índices de la tabla particionada con el mismo esquema de partición.

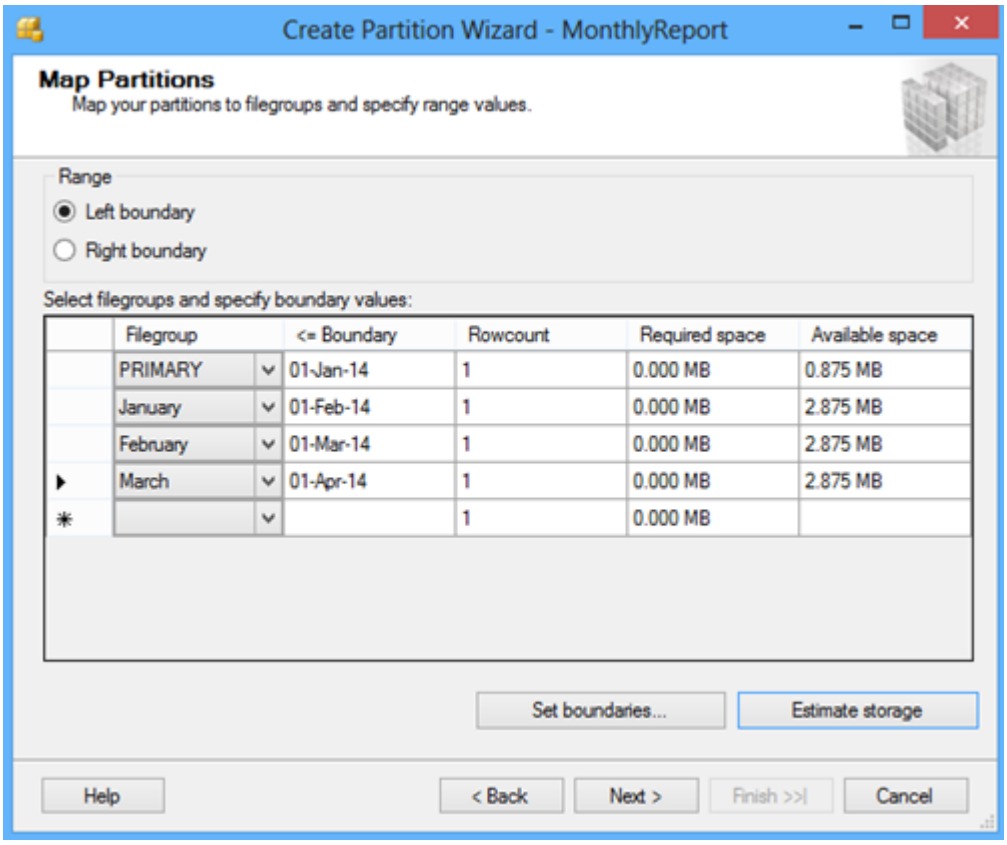
Después de seleccionar una columna para particionar haga clic en el botón Next. En la ventana **Select a Partition Function** ingrese el nombre de una función de partición para mapear las filas de la tabla o índice en particiones basadas en los valores de la columna ReportDate, o seleccione la función de partición existente:



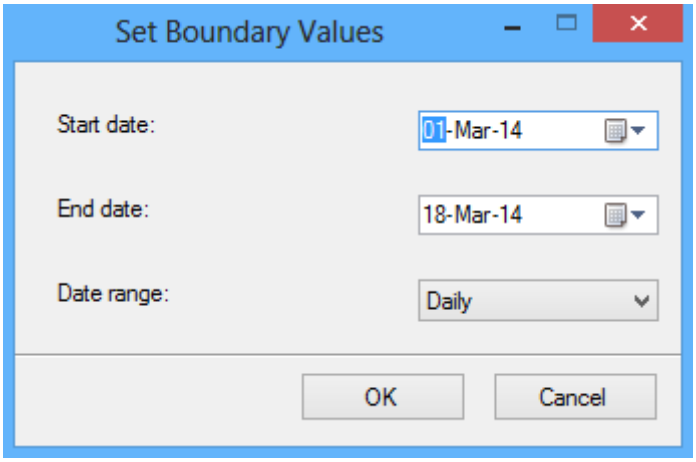
Haga clic en el botón **Next** y en la ventana **Select a Partition Scheme** cree el esquema de partición para mapear las particiones de la tabla MonthlyReport a diferentes grupos de archivos:



Haga clic en el botón **Next** y en la ventana **Map Partition**s elija el rango de particionamiento y seleccione los grupos de archivos disponibles y los límites del rango. El límite izquierdo está basado en Valor <= Límite y el límite derecho está basado en Valor < Límite.

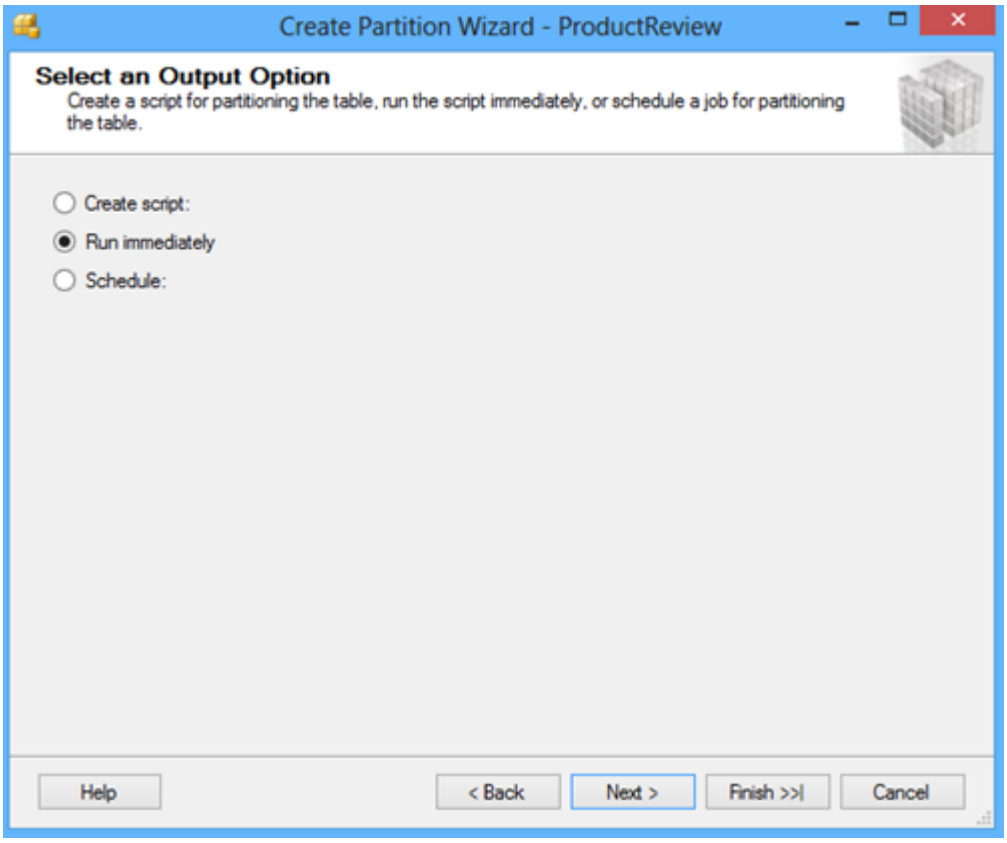


Haciendo clic en el botón **Set boundaries** usted puede personalizar el rango de datos y establecer las fechas de inicio y final para cada partición:

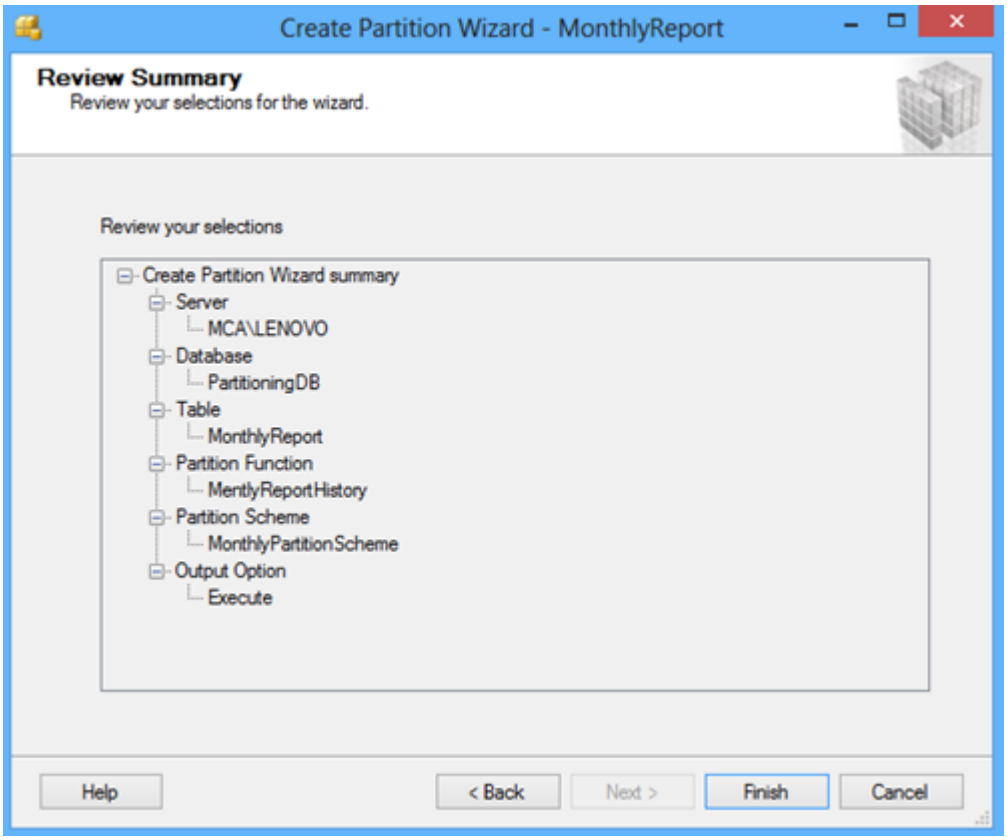


La opción **Estimate storage** determina las columnas Rowcont, Required (espacio requerido) y Available (espacio disponible), que muestran un estimado acerca del espacio requerido y el espacio disponible basado en el número de registros en la tabla.

La siguiente pantalla del asistente ofrece elegir la opción para ejecutar el script inmediatamente por el asistente para crear objetos y una tabla de partición, o crear un script y grabarlo. Un horario para ejecutar el script para realizar las operaciones automáticamente puede ser también especificado:



La siguiente pantalla del asistente muestra una revisión de las selecciones hechas en el asistente:



Haga clic en el botón **Finish** para completar el proceso: