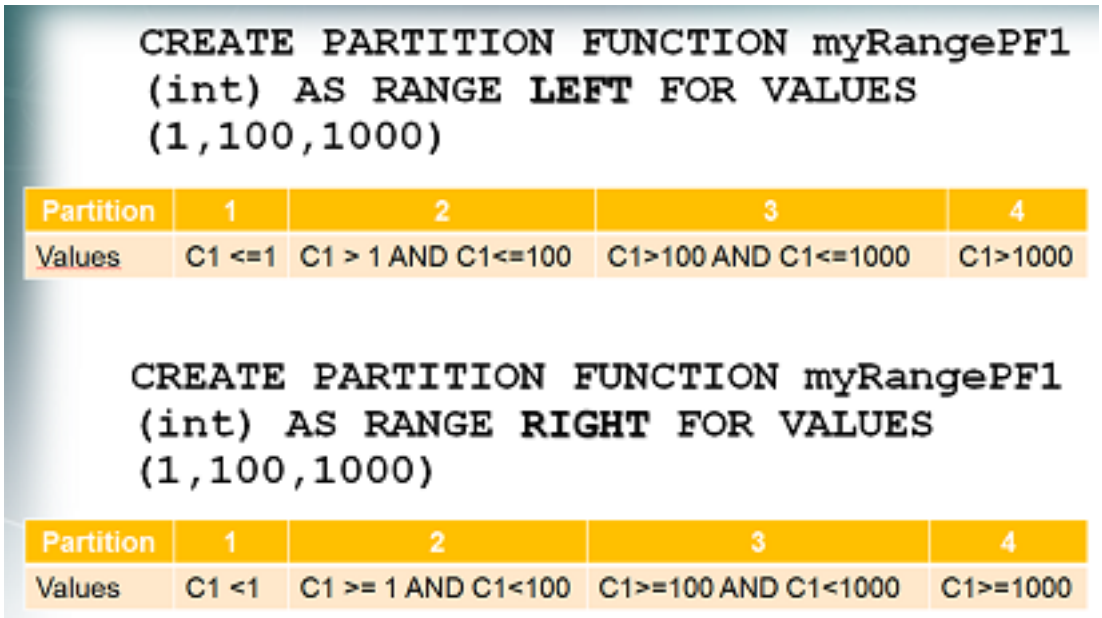
عنوان: **استفاده از قابلیت پارتیشن بندی در آرشیو جداول بانک‌های اطلاعاتی SQL Server**
نویسنده: محمد رجبی
تاریخ: ۱۳۹۳/۰۴/۰۸ ۱۴:۲۵
آدرس: www.dotnettips.info
گروه‌ها: SQL Server, Partitioning, Data Archiving

1- مقدمه پارتیشن بندی در بانک اطلاعاتی SQL Server، از ویژگی‌هایی است که از نسخه 2005، به این محصول اضافه شده است. بکارگیری این قابلیت که با Split کردن، محتوای یک جدول و قرار دادن آنها در چندین فایل، برای جداول حجیم، به ویژه جداولی که داده‌های آن حاوی مقادیر تاریخچه‌ای است، بسیار سودمند است.
سادگی در مدیریت داده‌ها و شاخص‌های موجود یک جدول (از قبیل اندازه فضای ذخیره سازی و استراتژی جدید Back up گیری)، اجرای سریعتر کوئری هایی که روی یک محدوده از داده‌ها کار می‌کنند و سهولت در آرشیو داده‌های قدیمی یک جدول، از قابلیت‌هایی است که استفاده از این ویژگی بوجود می‌آورد.
محدوده استفاده از این ویژگی روی یک بانک اطلاعاتی و در یک Instance است. بنابراین مباحث مرتبط با معماری Scalability را پوشش نمی‌دهد و صرفاً Solution ایی است که در یک Instance بانک اطلاعاتی استفاده می‌شود.

2- Data File و Filegroup هر بانک اطلاعاتی در حالت پیش فرض، شامل یک فایل داده‌ای (MDF.) و یک فایل ثبت تراکنشی (LDF.) می‌باشد. می‌توان جهت ذخیره سطرهای داده‌ای از فایل‌های بیشتری تحت نام فایل‌های ثانویه (NDF.) استفاده نمود. به همان طریق که در فایل سیستم، فایل‌ها به پوشه‌ها تخصیص داده می‌شوند، می‌توان Data File را به Filegroup تخصیص داد. چنانچه چندین Data File به یک Filegroup تخصیص داده شوند، داده‌ها در تمامی Data Fileها به طریق Round-Robin توزیع می‌شوند.

3- Partition Function مطابق با مقادیر تعریف شده در بدنه دستور، محدوده داده‌ای (پارتیشن‌ها) با استفاده از Partition Function ایجاد می‌شود. با در نظر گرفتن ستونی که به عنوان Partition Key انتخاب شده، این تابع یک Data Type را به عنوان ورودی دریافت می‌کند. در هنگام تعریف محدوده برای پارتیشن‌ها، به منظور مشخص کردن محدوده هر پارتیشن از Right و Left استفاده می‌شود.
Left نمایش دهنده‌ی حد بالای هر محدوده است و به طور مشابه، Right برای مشخص کردن حد پائین آن محدوده استفاده می‌شود. به منظور درک بهتر، به شکل زیر توجه نمائید:



همانطور که مشاهده می‌شود، همواره نیاز به یک Filegroup اضافه‌تری از آنچه مورد نظرتان در تعریف تابع است، می‌باشد. بنابراین اگر Function دارای n مقدار باشد، به n+1 مقدار برای Filegroup نیاز است.
همچنین هیچ محدودیتی برای اولین و آخرین بازه در نظر گرفته نمی‌شود. بنابراین جهت محدود کردن مقادیری که در این بازه‌ها قرار می‌گیرند، می‌توان از Check Constraint استفاده نمود.

3-1- Right or Left

یک سوال متداول اینکه از کدام مورد استفاده شود؟ در پاسخ باید گفت، به چگونگی تعریف پارتیشن هایتان وابسته است. مطابق شکل، تنها تفاوت این دو، در نقاط مرزی هر یک از پارتیشن‌ها می‌باشد. در بیشتر اوقات هنگام کار با داده‌های عددی می‌توان از Left استفاده نمود و بطور مشابه هنگامیکه نوع داده‌ها از جنس زمان است، می‌توان از Right استفاده کرد.

4- Partition Schema

گام بعدی پس از ایجاد Partition Function، تعریف Partition Schema است، که به منظور قرار گرفتن هر یک از پارتیشن‌های تعریف شده توسط Function در Filegroupهای مناسب آن استفاده می‌شود.

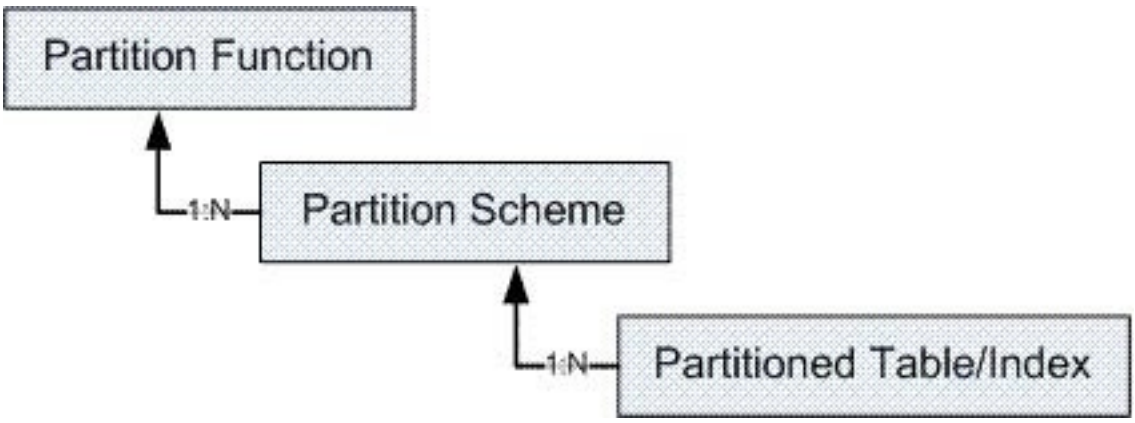
5- Partition Table

گام پایانی ایجاد یک جدول، استفاده از Partition Schema است، که داده‌ها را با توجه به رویه درون Partition Function مورد استفاده، ذخیره می‌کند. همانطور که می‌دانید هنگام ایجاد یک جدول، می‌توان مکان ذخیره شدن آنرا مشخص نمود.

```
Create Table <name> (…) ON …
```

دستور بعد از بخش ON، مشخص کننده مکان ذخیره جدول می‌باشد.

در هنگام ایجاد یک جدول، معمولاً جدول در Filegroup پیش فرض که PRIMARY است، قرار می‌گیرد. می‌توان با نوشتن نام Partition Schema و همچنین Partition Key که پیشتر ذکر آن رفت، بعد از بخش ON، برای جدول مشخص نمائیم که داده‌های آن به چه ترتیبی ذخیره شوند. ارتباط این سه به شرح زیر است:



توجه شود زمانیکه یک Primary Key Constraint به یک جدول اضافه می‌شود، یک Unique Clustered Index نیز همراه با آن ساخته می‌شود. چنانچه Primary Key شامل یک Clustered Index باشد، جدول با استفاده از این ستون (ستون‌های) شاخص ذخیره خواهد شد، در حالیکه اگر Primary Key شامل یک Non Clustered Index باشد، یک ساختار ذخیره-سازی اضافی ایجاد خواهد شد که داده‌های جدول در آن قرار خواهند گرفت.

6- Index & Data Alignment

به عنوان یک Best Practice هنگام ایجاد یک Partition Table به منظور پارتیشن بندی، از ساختار Aligned Index استفاده شود. بدین ترتیب که تعریف Index، شامل Partition Key (ستونی که معیاری برای پارتیشن بندی است) باشد. چنانچه این عمل انجام شود، داده‌های ذخیره شده مرتبط با هر پارتیشن متناظر با همان شاخص، در فایل داده‌ای (.NDF) ذخیره خواهند شد. از این رو چنانچه کوئری درخواست شده از جدول روی یک محدوده باشد

```
Where [OrderDate] Between …
```

تنها از شاخص متناظر با این داده استفاده می‌شود. بدین ترتیب بکارگیری آن برای Execution Plan بسیار سودمند خواهد بود. همچنین می‌توان استراتژی بازیافت سودمندی با Back up گیری از Filegroup ایجاد کرد. هنگامی که Indexها به صورت Aligned هستند می‌توان در کسری از ثانیه، محتوای یک Partition را به یک جدول دیگر منتقل نمود (تنها با تغییر در Meta Data آن).

بدین ترتیب برای بهرمندی از این مزایا، استفاده از Aligned Index توصیه شده است.

7- Operations

از نیازمندی‌های متداول در پارتیشنینگ می‌توان به افزودن، حذف پارتیشن‌ها و جابجایی محتوای یک پارتیشن که برای عملیات آرشیو استفاده می‌شود، اشاره کرد.

7-1- Split Partition

به منظور ایجاد یک محدوده جدید به پارتیشن‌ها استفاده می‌شود. یک نکته مهم مادامی که عملیات انتقال داده‌ها به پارتیشن جدید انجام می‌گیرد، روی جدول یک قفل انحصاری قرار می‌گیرد و بدین ترتیب عملیات ممکن است زمانبر باشد.
به عنوان یک Best Practice همواره یک Partition خالی را Split نمائید و پس از آن اقدام به بارگذاری داده در آن نمائید.
به یاد داشته باشید پیش از انجام عملیات splitting روی Partition Function با تغییر در Partition Schema (و بکارگیری Next Used) مشخص نمائید چه محدوده‌ای در این Filegroup جدید قرار خواهد گرفت.

7-2- Merge Partition

به منظور ادغام پارتیشن‌ها استفاده می‌شود، چنانچه پارتیشن خالی نیست، برای عملیات ادغام مسائل Performance به علت اینکه در طول عملیات از Lock (قفل انحصاری) استفاده می‌شود، در نظر گرفته شود.

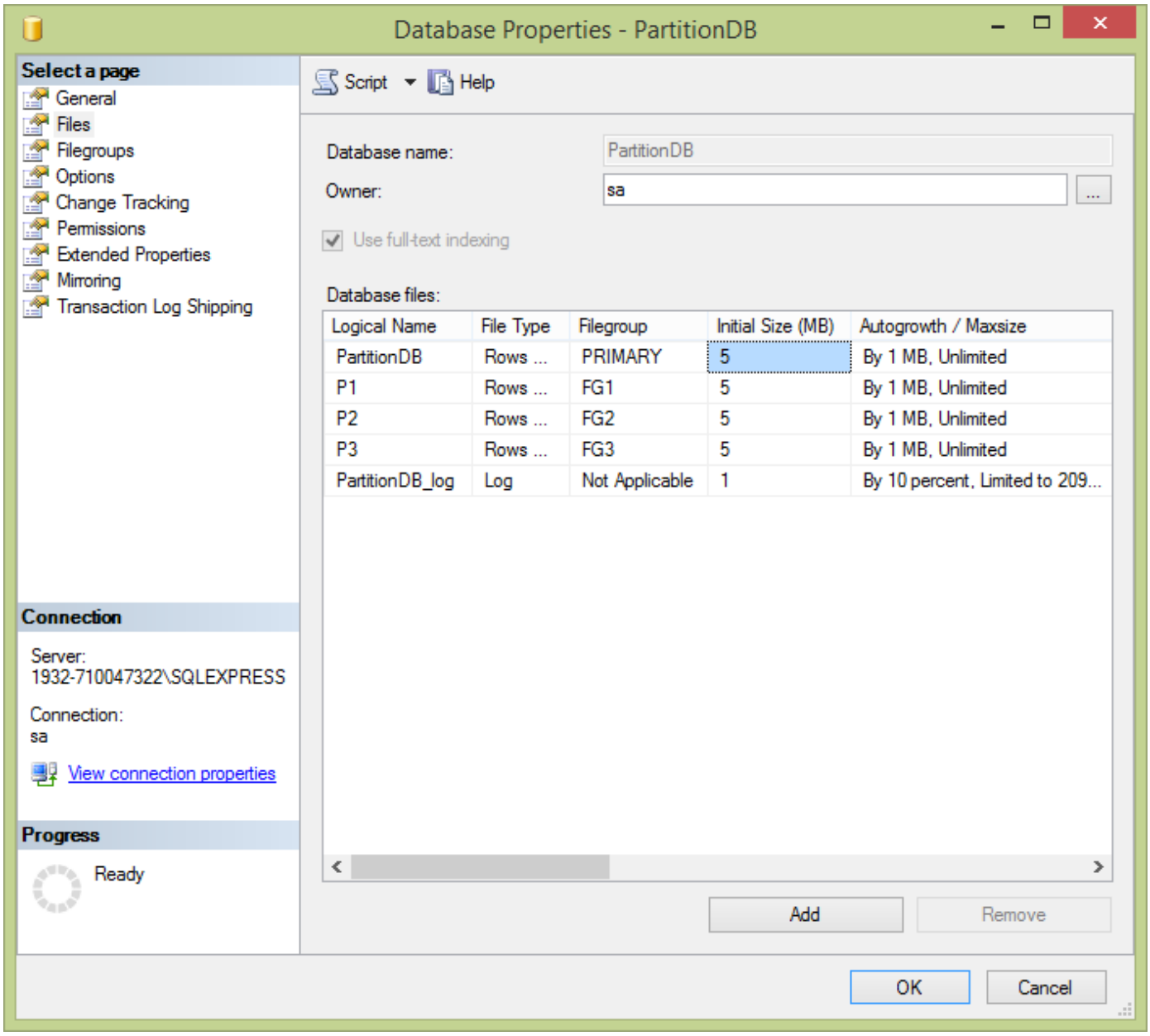
7-3- Switch Partition

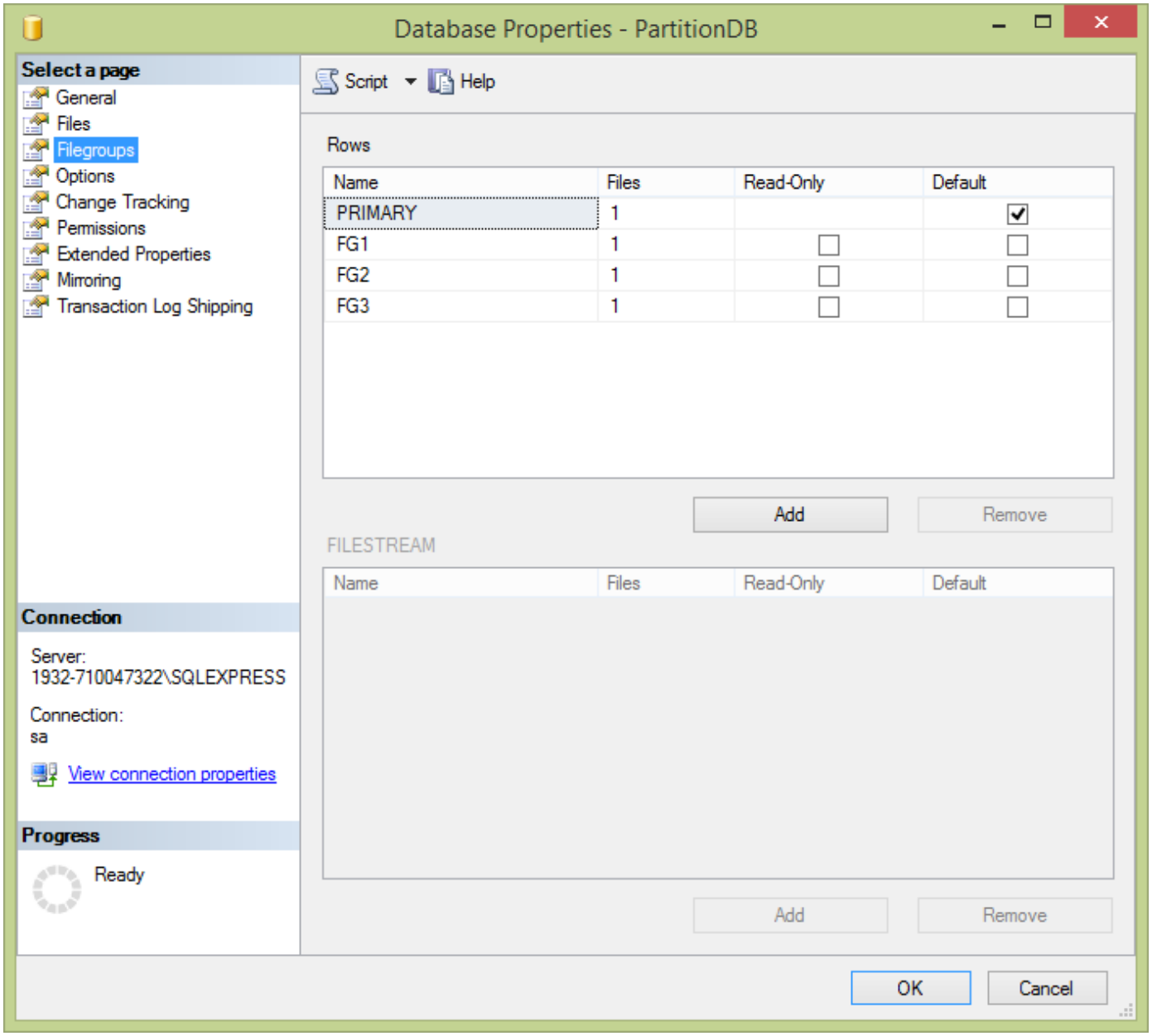
چنانچه جدول و شاخص‌های آن به صورت Aligned هستند، می‌توانید از Switch in و Switch out استفاده نمائید. عملیات بدین ترتیب انجام می‌شود که بلافاصله محتوای یک پارتیشن یا جدول (Source) در یک پارتیشن خالی جدولی دیگر و یا یک جدول خالی (Target) قرار می‌گیرد. عملیات تنها روی Meta Data انجام می‌گیرد و هیچ داده ای منتقل نمی‌شود.
محدودیت‌های بکارگیری به شرح زیر است:
- جدول یا پارتیشن Target باید حتماً خالی باشد.
- جداول Source و Target حتماً باید در یک Filegroup یکسان قرار داشته باشند.
- جدول Source باید حاوی Aligned Indexهای مورد نیاز Target و همچنین مطابقت در Filegroup را دارا باشد.
- چنانچه Target به عنوان یک پارتیشن است، اگر Source جدول است بایست دارای یک Check Constraint باشد در غیر این صورت چنانچه یک پارتیشن است باید محدوده آن در محدوده Target قرار گیرد.

8- بررسی یک سناریوی نمونه

در ابتدا یک بانک اطلاعاتی را به طریق زیر ایجاد می‌کنیم:
این بانک مطابق تصویر، شامل 3 عدد فایل گروپ (FG1، FG2 و FG3) و 3 عدد دیتا فایل (P1، P2 و P3) می‌باشد. Filegroup پیش فرض Primary است، که چنانچه در تعریف جداول به نام Partition Schema و Partition Key مرتبط اشاره نشود، به طور پیش فرض در Filegroup موسوم به Primary قرار می‌گیرد. چنانچه چک باکس Default انتخاب شود، همانطور که قابل حدس زدن است، آن Filegroup در صورت مشخص نکردن نام Filegroup در تعریف جدول، به عنوان مکان ذخیره سازی انتخاب می‌شود. چک باکس Read Only نیز همانطور که از نامش پیداست، چنانچه روی یک Filegroup تنظیم گردد، عملیات مربوط به Write روی

داده‌های آن قابل انجام نیست و برای Filegroup هایی که جنبه نگهداری آرشیو را دارند، قابل استفاده است.
چنانچه Filegroup ای را از حالت Read Only دوباره خارج کنیم، می‌توان عملیات Write را دوباره برای آن انجام داد.





پس از ایجاد بانک اطلاعاتی، گام بعدی ایجاد یک Partition Function و پس از آن یک Partition Schema است. همانطور که مشاهده می‌کنید در Partition Function از سه مقدار استفاده شده، بنابراین در Partition Schema باید از چهار Filegroup استفاده شود، که در مثال ما از Filegroup پیش فرض که Primary است، استفاده شده است.

```
USE [PartitionDB]
GO
CREATE PARTITION FUNCTION pfOrderDateRange(DATETIME)
AS
RANGE LEFT FOR VALUES ('2010/12/31','2011/12/31','2012/12/31')
GO
CREATE PARTITION SCHEME psOrderDateRange
AS
PARTITION pfOrderDateRange TO (FG1,FG2,FG3,[PRIMARY])
GO
```

پس از طی گام‌های قبل، به ایجاد یک جدول به صورت Aligned Index مبادرت ورزیده می‌شود.

```
CREATE TABLE Orders
```

```
(
OrderID INT IDENTITY(1,1) NOT NULL,
OrderDate DATETIME NOT NULL,
OrderFreight MONEY NULL,
ProductID INT NULL,
CONSTRAINT PK_Orders PRIMARY KEY CLUSTERED (OrderID ASC, OrderDate ASC)
ON psOrderDateRange (OrderDate)
) ON psOrderDateRange (OrderDate)
GO
```

در ادامه برای بررسی درج اطلاعات در پارتیشن با توجه به محدوده آنها اقدام به افزودن رکوردهایی در جدول ساخته شده می‌نمائیم.

```
SET NOCOUNT ON
DECLARE @OrderDate DATETIME
DECLARE @X INT
SET @OrderDate = '2010/01/01'
SET @X = 0
WHILE @X < 300
BEGIN
INSERT dbo.Orders ( OrderDate, OrderFreight, ProductID)
VALUES( @OrderDate + @X, @X + 10, @X)
 SET @X = @X + 1
END
GO
SET NOCOUNT ON
DECLARE @OrderDate DATETIME
DECLARE @X INT
SET @OrderDate = '2011/01/01'
SET @X = 0
WHILE @X < 300
BEGIN
INSERT dbo.Orders ( OrderDate, OrderFreight, ProductID)
VALUES( @OrderDate + @X, @X + 10, @X)
 SET @X = @X + 1
END
GO
SET NOCOUNT ON
DECLARE @OrderDate DATETIME
DECLARE @X INT
SET @OrderDate = '2012/01/01'
SET @X = 0
WHILE @X < 300
BEGIN
INSERT dbo.Orders ( OrderDate, OrderFreight, ProductID)
VALUES( @OrderDate + @X, @X + 10, @X)
 SET @X = @X + 1
END
GO
```

از طریق دستور Select زیر می‌توان نحوه توزیع داده‌ها را در جدول مشاهده کرد.

```
USE [PartitionDB]
GO
SELECT OBJECT_NAME(i.object_id) AS OBJECT_NAME,
p.partition_number, fg.NAME AS FILEGROUP_NAME, ROWS, au.total_pages,
CASE boundary_value_on_right
WHEN 1 THEN 'Less than'
ELSE 'Less or equal than' END AS 'Comparition',VALUE
FROM sys.partitions p JOIN sys.indexes i
ON p.object_id = i.object_id AND p.index_id = i.index_id
JOIN sys.partition_schemes ps ON ps.data_space_id = i.data_space_id
JOIN sys.partition_functions f ON f.function_id = ps.function_id
LEFT JOIN sys.partition_range_values rv
ON f.function_id = rv.function_id
AND p.partition_number = rv.boundary_id
JOIN sys.destination_data_spaces dds
ON dds.partition_scheme_id = ps.data_space_id
AND dds.destination_id = p.partition_number
JOIN sys.filegroups fg
ON dds.data_space_id = fg.data_space_id
JOIN (SELECT container_id, SUM(total_pages) AS total_pages
FROM sys.allocation_units
GROUP BY container_id) AS au
ON au.container_id = p.partition_id WHERE i.index_id < 2
```

خروجی دستور فوق به شرح زیر است:

	OBJECT_NAME	partition_number	FILEGROUP_NAME	ROWS	total_pages	Comparition	VALUE
1	Orders	1	FG1	300	4	Less or equal than	2010-12-31 00:00:00.000
2	Orders	2	FG2	300	4	Less or equal than	2011-12-31 00:00:00.000
3	Orders	3	FG3	300	4	Less or equal than	2012-12-31 00:00:00.000
4	Orders	4	PRIMARY	0	0	Less or equal than	NULL

در ادامه به ایجاد یک Filegroup جدید می‌پردازیم.

```
/* Query 2-3- Split a partition*/
-- Add FG4:
ALTER DATABASE PartitionDB ADD FILEGROUP FG4
Go
ALTER PARTITION SCHEME [psOrderDateRange] NEXT USED FG4
GO
ALTER PARTITION FUNCTION [pfOrderDateRange]() SPLIT RANGE('2013/12/31')
GO
-- Add Partition 4 (P4) to FG4:
GO
ALTER DATABASE PartitionDB ADD FILE
(
NAME = P4,
FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL10_50.SQLEXPRESS\MSSQL\DATA\P4.NDF'
, SIZE = 1024KB , MAXSIZE = UNLIMITED, FILEGROWTH = 10%)
 TO FILEGROUP [FG4]
--
GO
```

و در ادامه به درج اطلاعاتی برای بررسی نحوه توزیع داده‌ها در Filegroup هایمان می‌پردازیم.

```
SET NOCOUNT ON
DECLARE @OrderDate DATETIME
DECLARE @X INT
SET @OrderDate = '2013/01/01'
SET @X = 0
WHILE @X < 300
BEGIN
INSERT dbo.Orders ( OrderDate, OrderFreight, ProductID)
VALUES( @OrderDate + @X, @X + 10, @X)
 SET @X = @X + 1
END
GO
SET NOCOUNT ON
DECLARE @OrderDate DATETIME
DECLARE @X INT
SET @OrderDate = '2012/01/01'
SET @X = 0
WHILE @X < 300
BEGIN
INSERT dbo.Orders ( OrderDate, OrderFreight, ProductID)
VALUES( @OrderDate + @X, @X + 10, @X)
 SET @X = @X + 1
END
GO
```

خروجی کار تا این مرحله به شکل زیر است:

Results Messages

	OBJECT_NAME	partition_number	FILEGROUP_NAME	ROWS	total_pages	Comparition	VALUE
1	Orders	1	FG1	300	4	Less or equal than	2010-12-31 00:00:00.000
2	Orders	2	FG2	300	4	Less or equal than	2011-12-31 00:00:00.000
3	Orders	3	FG3	600	5	Less or equal than	2012-12-31 00:00:00.000
4	Orders	5	PRIMARY	0	0	Less or equal than	NULL
5	Orders	4	FG4	300	4	Less or equal than	2013-12-31 00:00:00.000

جهت ادغام پارتیشن‌ها به طریق زیر عمل می‌شود:

```
/* Query 2-4- Merge Partitions */
ALTER PARTITION FUNCTION [pfOrderDateRange]() MERGE RANGE('2010/12/31')
Go
```

پس از اجرای دستور فوق خروجی به شکل زیر خواهد بود:

	OBJECT_NAME	partition_number	FILEGROUP_NAME	ROWS	total_pages	Comparition	VALUE
1	Orders	1	FG2	600	7	Less or equal than	2011-12-31 00:00:00.000
2	Orders	2	FG3	600	5	Less or equal than	2012-12-31 00:00:00.000
3	Orders	4	PRIMARY	0	0	Less or equal than	NULL
4	Orders	3	FG4	300	4	Less or equal than	2013-12-31 00:00:00.000

به منظور آرشیو نمودن اطلاعات به طریق زیر از Switch استفاده می‌کنیم. ابتدا یک جدول موقتی برای ذخیره رکوردهایی که قصد آرشیو آنها را داریم، ایجاد می‌کنیم. همانگونه که در تعریف جدول مشاهده می‌کنید، نام Filegroup ای که برای ساخت این جدول استفاده می‌شود، با Filegroup ای که قصد آرشیو اطلاعات آنرا داریم، یکسان است.
در ادامه می‌توان مثلاً با ایجاد یک Temporary Table به انتقال این اطلاعات بدون توجه به Filegroup آنها پرداخت.

```
/* Query 2-5- Switch Partitions */
USE [PartitionDB]
GO
CREATE TABLE [dbo].[Orders_Temp](
[OrderID] [int] IDENTITY(1,1) NOT NULL,
[OrderDate] [datetime] NOT NULL,
[OrderFreight] [money] NULL,
[ProductID] [int] NULL,
 CONSTRAINT [PK_OrdersTemp] PRIMARY KEY CLUSTERED ([OrderID] ASC,[OrderDate] ASC)ON FG2
) ON FG2
GO
USE [tempdb]
GO
CREATE TABLE [dbo].[Orders_Hist](
[OrderID] [int] NOT NULL,
[OrderDate] [datetime] NOT NULL,
[OrderFreight] [money] NULL,
[ProductID] [int] NULL,
 CONSTRAINT [PK_OrdersTemp] PRIMARY KEY CLUSTERED ([OrderID] ASC,[OrderDate] ASC)
)
GO
USE [PartitionDB]
GO
ALTER TABLE [dbo].[Orders] SWITCH PARTITION 1 TO [dbo].[Orders_Temp]
GO
INSERT INTO [tempdb].[dbo].[Orders_Hist]
SELECT * FROM  [dbo].[Orders_Temp]
GO
DROP TABLE [dbo].[Orders_Temp]
GO
```

```
SELECT * FROM [tempdb].[dbo].[Orders_Hist]
```

پس از اجرای کامل این دستورات، توزیع داده در بانک اطلاعاتی مثال مورد بررسی به شکل زیر است.

	OBJECT_NAME	partition_number	FILEGROUP_NAME	ROWS	total_pages	Comparition	VALUE
1	Orders	2	FG3	600	5	Less or equal than	2012-12-31 00:00:00.000
2	Orders	4	PRIMARY	0	0	Less or equal than	NULL
3	Orders	3	FG4	300	4	Less or equal than	2013-12-31 00:00:00.000
4	Orders	1	FG2	0	0	Less or equal than	2011-12-31 00:00:00.000

نظرات خوانندگان

نویسنده: محمد رضا موسائی
تاریخ: ۱۳۹۳/۰۷/۲۵ ۱۸:۲۳

بسیار امکان زیبایی است و بهتر اینکه اطلاعات قابل دسته بندی در فایل‌های ndf بوده و از همه مهمتر اینکه می‌توان از هر پارتیشن به صورت جداگانه backup تهیه کرد.

نویسنده: مسعود خان
تاریخ: ۱۳۹۳/۰۸/۱۳ ۱۸:۵۲

سلام جناب رجبی مطلب بسیار خوبی بود مخصوصا که من واقعا بهش نیاز داشتم اما سوالاتی برام پیش اومده من دیتابیس بزرگی دارم که سه جدولش از همه بزرگته و یکیش 190 میلیون و دو تا جدول 43 میلیون رکوردی دارم و بقیه جداول زیر 5 میلیون هست چون در شروع کار تجربه خوبی نداشتم و از حجم اطلاعات مطمئن نبودم روی دیتابیسم pk نذاشتم حالا که می‌خوام pk و ایندکس گذاریش کنم برای هر جدول یک filegroup و در هر فایل گروپ یک فایل برای pk و یک فایل برای indexها گذاشتم و تقریبا هر کدوم از اون جدوال بزرگ را به 30 تا جدول تقسیم کردم تا سرعت پرس و جوهای زیادم بسیار کمتر بشه که به نظرم خوب نمی‌اومد (کار پرس و جو و ایجاد کوئری را دشوارتر می‌کنه) تا با راه حل شما اشنا شدم اما سوالم اینجاست اگر داده هام رو بر اساس تاریخ در datafileها پارتیشن بندی کنم زمانی که بخوام مثلا اطلاعات تا قبل از سال 85 را ارشیو کنم تاثیری در زمان پرس و جوی من دارد و کلیدها و ایندکس‌ها وضعیتشون به چه شکل خواهد بود اگر از راه حل خودم استفاده نکنم چون همینجوریش حجم دیتابیسم 35 گیگه و با گذاشتن کلید و ایندکس چیزی بین 15 تا 20 گیگ هم اضافه میشه که خیلی بد هست.
الان که یه خورده رو کدها کار کردم یه سوال برام پیش اومد مثالی میزنم من جدولی دارم که تاریخ را با نام شرکت میگیره و یک کد گزارش میده حالا من این کد گزارش را با مثلا 10 هزار رکورد در جدول دوم ذخیره می‌کنم و و باز با همون کد گزارش 5000 رکورد را در جدول سوم ذخیره می‌کنم و به همین ترتیب برای تمام روزها و برای کل شرکت‌ها این کد گزارش تولید و با حجم انبوهی از رکوردها در جداول ذخیره میشن
حالا سوال اینه که چطور بر اساس تاریخ که در جدول فقط اول هست جداول دیگر را پارتیشن بندی کنم ؟

نویسنده: محمد رجبی
تاریخ: ۱۳۹۳/۰۸/۱۴ ۱۳:۴۰

با سلام و احترام
پیشنهاد می‌کنم مطلب " Download "Partitioned Table and Index Strategies Using SQL Server 2008" white paper " را مطالعه فرمائید. به منظور پیاده سازی این قابلیت در بانک اطلاعاتی تان یک راه حل می‌تواند اینگونه باشد که یک DB جدید ایجاد نمائید که در آن تمامی زیرساخت‌ها ایجاد شود (ایندکس، Partition Function و ...) در ادامه به انتقال داده‌های از بانک عملیاتی به DB جدید بپردازید. برای مشاهده جزئیات به این مطلب مراجعه نمائید.

نویسنده: مسعود خان
تاریخ: ۱۳۹۳/۰۸/۱۴ ۲۰:۴۷

جناب رجبی ممنون بابت پاسختون من فایل را دارم می‌خونم که بخش هایش رو خودتون توضیح دادید اما با منطقی که جداولم دارن به مشکل خوردم. من کمی گشتم و با فشرده سازی به صورت row و page اشنا شدم که بسیار جالب بودو من ان را برای بزرگترین جدولم تست کردم و نتایج قابل توجهی داشت
برای row حجم جدول از 17.6 به 9.8 کاهش پیدا کرد
برای page حجم جدول از 17.6 به 4.8 کاهش پیدا کرد که خیلی خوبه
اما سوالی پیش میاد برام این فشرده سازی چقدر در پرس و جوها و کوئری‌های پیچیده تاثیر داره اصلا این تاثیر مثبت هست یا منفی ؟
من هر روز از این جداول پرس و جو می‌کنم بنابراین می‌شود گفت خواندن از این جداول در حد متوسطی است نوشتن ندارد و ایا فشرده سازی بر روی پریمری کدها و ایندکس‌ها هم تاثیر دارد و اگر دارد مثبت است یا خیر؟و تاثیرش بر روی پرس و جوها چقدر است؟

و سوال اخر به نظرتون با این پارتیشن بندی من pk و ایندکس‌ها را در فایل گروپی جدا ذخیره کنم یا در فایل primary به خصوص که بخوام اطلاعات قبل از سال 85 را ارشیو کنم و هر ساله این ارشیو یک سال بیشتر می‌شود
با تشکر

نویسنده: محمد رجبی
تاریخ: ۱۳۹۳/۰۸/۱۵ ۲۱:۴۴

با سلام، ابتدا از سوال آخر شروع میکنم، چنانچه تمایل دارید از امکان پارتیشنینگ در آرشیو بانک اطلاعاتی تان استفاده کنید همانگونه که در متن اشاره شده، باید جداول به شکل Aligned Index تعریف شوند.(پس نیاز به Filegroup می‌باشد)
در مورد فشرده سازی به هیچ وجه هدف افزایش سرعت در پرس و جوها نیست بلکه کاهش حجم، هدف غائی است. همینطور هدف از ایجاد ایندکس در این است که هر رکورد در مکان صحیح خود به هنگام درج قرار گیرد و بدین ترتیب چنانچه مجموعه مرتب شده باشد، مرتبه اجرائی از O(n) به O(Log n) کاهش می‌یابد و ... به همین خاطر است که در سیستم‌های OLTP توصیه شده از Index هوشمندانه استفاده شود (بدلیل اینکه عملیات درج، بروزرسانی و حذف به کندی صورت نگیرد) و در سیستم‌های DSS برعکس به دلیل ماهیت غیر عملیاتی بودن آنها استفاده فراوان از Index توصیه شده است.
بطور کلی ساختار ایندکس پایه در SQL Server عبارتند از: ایندکس‌های Clustered ، ایندکس‌های Nonclustered در Heap و ایندکس‌های Nonclustered در یک ایندکس‌های Clustered شده؛ روش ذخیره فیزیکی داده بین این ایندکس‌ها متفاوت است و همچنین روشی که SQL Server برای پیمایش ایندکس در B-Tree استفاده می‌کند بسته به این سه نوع متفاوت خواهد بود.

نویسنده: مسعود خان
تاریخ: ۱۳۹۳/۰۸/۱۶ ۱۰:۷

ممنون از پاسختون من دیشب بالاخره تونستم راهی برای پیاده سازی این روشی روی دیتابیس پیدا و اجرا کنم و نتیجه بسیار رضایت بخش بود تقریبا برای پرس و جو هایی که شامل aligned index میشد تقریبا 70 بار سریعتر بود
اما حجم فایل‌های .ndf روی هم رفته تقریبا 4 گیگ میشه در صورتی که خود جدول 1.48 گیگ هست

فعلا دارم روی فشرده سازی و تعریف ایندکس روی بعضی فیلدها کار می‌کنم
اما نمی‌دونم برای ستون int مثلا برای تعداد کدام ایندکس را پیاده سازی کنم بهتر است و تعریف این ایندکس‌ها مثل تعریف pkها که در فایل‌ها پخش میشن ب چه صورت هست فعلا دارم روش کار می‌کنم اما ممنون میشم در این زمینه هم راهنماییم کنید
با تشکر

نویسنده: مسعود خان
تاریخ: ۱۳۹۳/۰۸/۱۶ ۱۲:۵۵

جناب رجبی یه سوال در رابطه با pk دارم اونم این هست که پریمری کد را روی دو فیلد گذاشتید که کلاستر هست
حالا سوال این هست که گذاشتن دو فیلد به عنوان یک پریمری کد چقدر در حجم و کارایی تاثیر داره ؟
اگه سه تا بشه چطور ؟

```
CONSTRAINT PK_Orders PRIMARY KEY CLUSTERED (OrderID ASC, OrderDate ASC)
```

و همینطور میشه مثالی بزنید که بشه ایندکس را هم به صورت پریمری به جدول اضافه کرد که ایندکس‌ها در فایل‌ها تقسیم بشن دقیقا مثل pk بالا؟

نویسنده: محمد رجبی
تاریخ: ۱۳۹۳/۰۸/۱۷ ۸:۴۷

در مثال مذکور از **Partition Key** در زمان تعریف یک Primary Key Constraint روی جدول به منظور داشتن ساختار Aligned Index ، استفاده نموده ایم.
در هنگام ایجاد یک Primary Key Constraint بطور خودکار یک Unique Clustered Index نیز روی ستون (های) شرکت یافته در تعریف Primary Key ایجاد می‌شود و بدین ترتیب Table براساس این فیلد (ها) به شکل Sort شده نگهداری می‌شود، ضمن اینکه

هر Table می‌تواند، شامل 1 عدد Clustered Index و 249 عدد Nonclustered Index باشد.

نویسنده: پروانه
تاریخ: ۱۳۹۳/۰۸/۱۸ ۱۶:۲۷

با سلام ،

پایگاه داده‌های سیستم‌های VOIP که اطلاعات تماس در لحظه در آن ذخیره می‌شوند ، پیوسته در حال رشد است . نیاز است که یک سری اطلاعات آن آرشیو شود.

به غیر از پارتیشن بندی ، چه روش‌های ساده‌تری وجود دارد برای اینکار . به نظر می‌آید پارتیشن بندی در زمان هایی که می‌خواهیم یک پارتیشن به پارتیشن دیگر ادغام شود کند است. سوال دیگر هم این است که در صورت استفاده از پارتیشن بندی ، آیا می‌توان ماه به ماه اطلاعات در پارتیشن دیگر ادغام شود یا این کار سالانه باید انجام گیرد ؟ما لازم داریم اطلاعات فقط یک سال در یک پارتیشن قرار گیرد ،

نویسنده: محمد رجبی
تاریخ: ۱۳۹۳/۰۸/۱۹ ۱۲:۴۵

با سلام -همانطور که می‌دانیم؛ هدف فرآیند آرشیو حذف گروهی از داده‌ها از بانک اطلاعاتی می‌باشد. این داده‌ها دیگر مورد نیاز در سیستم عملیاتی نمی‌باشند ولی می‌خواهیم آنها را حفظ نمائیم. تکنیک‌های متفاوتی برای این منظور وجود داشته است از قبیل:
- **Application partitioning:** به طور کلی جهت دستیابی به اطلاعات (از محیط عملیاتی و یا محیط‌های گزارشی گیری با ماهیت آرشیو) از درون خود برنامه کاربردی با ماشین مورد نظر ارتباط برقرار می‌گردید.
- **Partition Views (DPVs):** جداول یکسانی ایجاد می‌گردید، که هر یک حاوی قسمتی از اطلاعات بودند و بدین ترتیب برای دستیابی به تمامی داده‌های این جداول با ایجاد یک View (پیوند میان این جداول) کلیه اطلاعات قابل دسترسی بود.
در تکنیک پارتیشنینگ کلیه محتوای جدول بصورت یکجا قابل دستیابی است، همچنین جهت آرشیو با استفاده از عملگر Switch این امکان را داریم که در کسری از ثانیه (با تنها انتقال Meta data) به آرشیو اطلاعات بپردازیم.
تا پیش از انتشار نسخه 2012 توان پشتیبانی از 1.000 پارتیشن وجود داشت این امکان در نسخه 2014 به 15.000 پارتیشن رسیده است.
همانگونه که ذکر گردید جهت آرشیو از عملگر Switch استفاده می‌شود (و نه عملگر Merge) همچنین در خصوص نحوه انجام پارتیشن بندی از آنجا که تابع یک Data type می‌گیرد الزامی به نوع Date نیست و مطابق شکل می‌تواند نوع آن Int نیز باشد، ولی عملیات آرشیو در ذات خود به اطلاعات تاریخچه ای اشاره می‌کند. جهت اطلاعات بیشتر به لینک Partitioned Tables and Indexes مراجعه شود.

نویسنده: مسعود خان
تاریخ: ۱۳۹۳/۰۸/۲۱ ۱۹:۳۶

جناب رجبی یک سوال سیستم ارشیو کردن در پارتیشن بندی به چه شکل هست و اینکه اگر بخشی از پارتیشن‌ها مثلا قبل از سال 2012 را به ارشیو ببریم در سرعت پرس و جوها تاثیر دارد و اینکه بعد از ارشیو کردن میشه از ارشیو در اورد او پارتیشن‌ها را یا نه؟

نویسنده: محمد رجبی
تاریخ: ۱۳۹۳/۰۸/۲۵ ۱۲:۳۰

از آنجایی که هر Data File شامل یک مسیر فیزیکی است (در یک Filegroup قرار می‌گیرد)، بنابراین این امکان وجود دارد که محتوای یک جدول حجیم را به چندین قسمت در سطح کنترلر تقسیم نمود، اگر سوالتان در خصوص اینکه «پارتیشن‌ها را بعد از اینکه آرشیو شدند، می‌توان از وضعیت آرشیو خارج نمود» و منظور Read Only نمودن پارتیشن‌ها است همانگونه که در متن ذکر شده " چک باکس Read Only ... چنانچه روی یک Filegroup تنظیم گردد، عملیات مربوط به Write روی داده‌های آن قابل انجام

نیست و برای Filegroup هایی که جنبه نگهداری آرشیو را دارند، قابل استفاده است. و چنانچه Filegroup ای را از حالت Read Only دوباره خارج کنیم، می‌توان عملیات Write را دوباره برای آن انجام داد. " در این حالت به منظور بالا بردن Performance می‌توانید از RAID 0 یا RAID 1 جهت بهره مندی از Fast read استفاده کنید.

نویسنده: محمد رجبی
تاریخ: ۱۳۹۳/۰۸/۲۵ ۱۲:۳۹

می توانید برای جداولی که حاوی داده می‌باشند نیز از امکان Partitioning استفاده نمائید. برای جدول Orders سناریوی مورد بررسی، بدون استفاده از PartitionSchema به شکل زیر ایجاد می‌شود:

```
CREATE TABLE Orders
(
OrderID INT IDENTITY(1,1) NOT NULL,
OrderDate DATETIME NOT NULL,
OrderFreight MONEY NULL,
ProductID INT NULL,
CONSTRAINT PK_Orders PRIMARY KEY CLUSTERED (OrderID ASC, OrderDate ASC)
ON [PRIMARY]
) ON [PRIMARY]
GO
```

می‌توانید به شکل زیر محتوای جدول که در فایل گروپ PRIMARY قرار گرفته را در سایر پارتیشن‌ها قرار داد.

```
ALTER TABLE Orders DROP CONSTRAINT PK_Orders
 WITH (MOVE TO psOrderDateRange (OrderDate))
 Go
```

نویسنده: مهدی آقازاده
تاریخ: ۱۳۹۳/۱۲/۰۳ ۱۱:۵

سلام
آیا فقط تنها در نسخه Sql Server E nterprise Edition می‌توان از پارتیشن استفاده کرد ؟

نویسنده: وحید فرهمندیان
تاریخ: ۱۳۹۳/۱۲/۰۴ ۱۷:۵۷

جهت مشاهده امکانات موجود در نسخ مختلف SQL Server به اینجا مراجعه کنید.