

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 کوئری‌هایی 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 Left و Right 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 نمایید:

```
CREATE PARTITION FUNCTION myRangePF1
(int) AS RANGE LEFT FOR VALUES
(1,100,1000)
```

Partition	1	2	3	4
Values	C1 <=1	C1 > 1 AND C1<=100	C1>100 AND C1<=1000	C1>1000

```
CREATE PARTITION FUNCTION myRangePF1
(int) AS RANGE RIGHT FOR VALUES
(1,100,1000)
```

Partition	1	2	3	4
Values	C1 <1	C1 >= 1 AND C1<100	C1>=100 AND C1<1000	C1>=1000

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.

Right or Left 3-1-

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.

Partition Schema 4-

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.

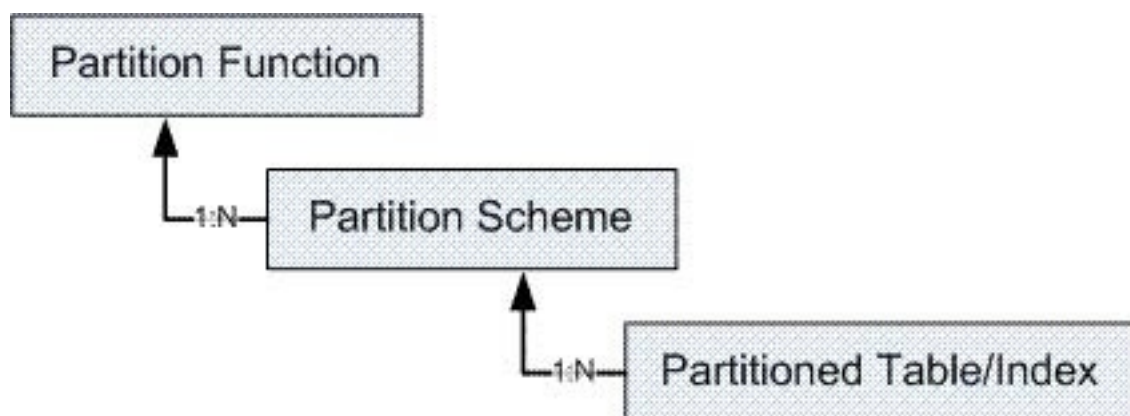
Partition Table 5-

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 ذخیره‌سازی اضافی ایجاد خواهد شد که داده‌های جدول در آن قرار خواهند گرفت.

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 پارتیشن‌بندی 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 داده‌ای 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.

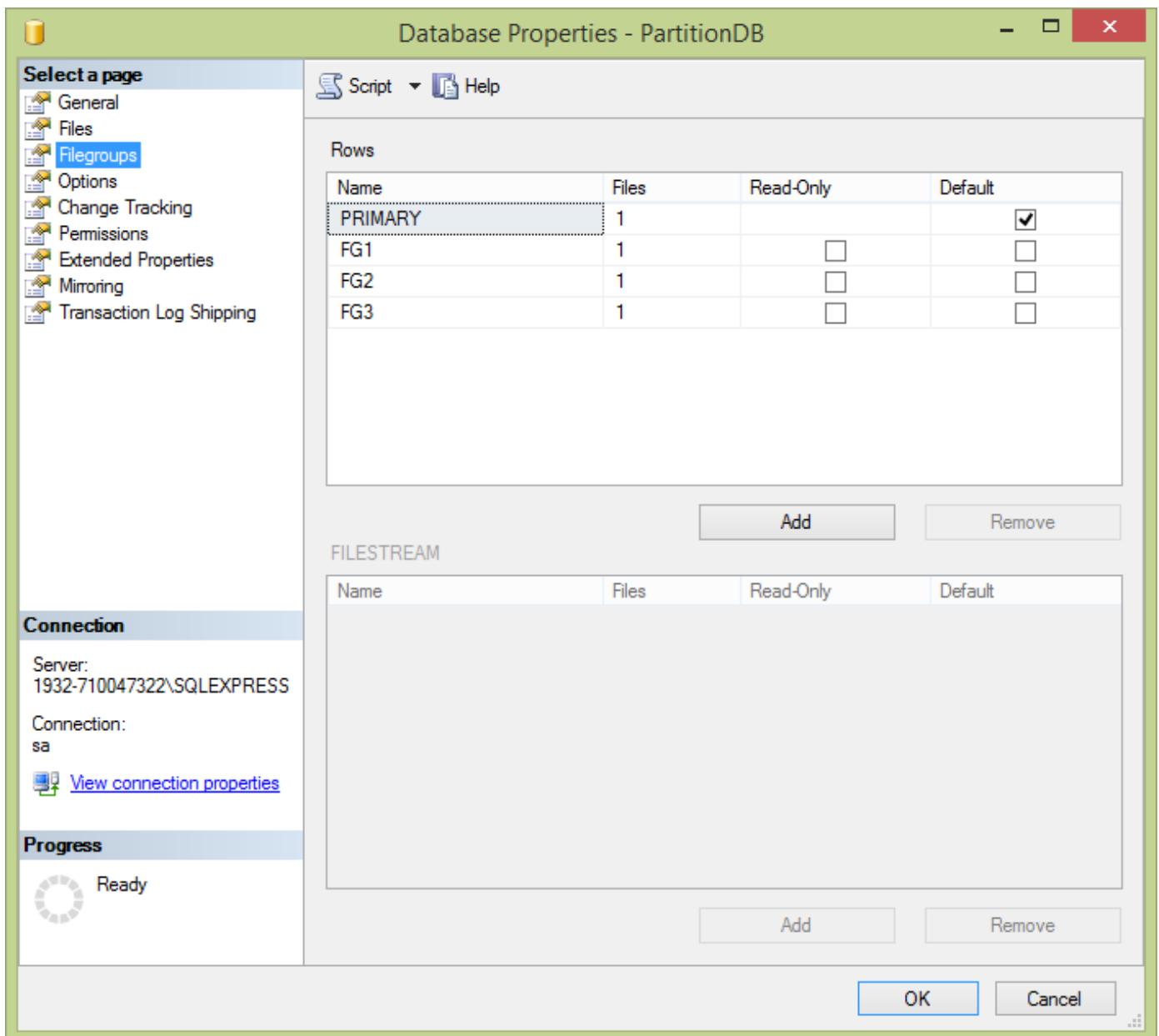
8- بررسی یک سناریوی نمونه

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 'Comparison',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	Comparison	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:

	OBJECT_NAME	partition_number	FILEGROUP_NAME	ROWS	total_pages	Comparison	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	Comparison	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 آنها را 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	Compartion	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