**What is 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 so that ‘groups of rows’ are mapped into individual partitions

Table Partitioning is an Enterprise Edition feature only

**What are primary files and filgroups?**

A filegroup can be considered a logical storage unit to house database objects that maps to a file system file or multiple files

**SQL Server Partitioned Table Creation**

* Create file groups in individual disk drive (if possible)
* Create a [partition function](http://msdn.microsoft.com/en-us/library/ms187802.aspx)
* Create a [partition scheme](http://msdn.microsoft.com/en-us/library/ms179854.aspx)
* Partition the table based on the datetime column

**Information about partition**

SELECT ps.name,pf.name,boundary\_id,value

FROM sys.partition\_schemes ps

INNER JOIN sys.partition\_functions pf ON pf.function\_id=ps.function\_id

INNER JOIN sys.partition\_range\_values prf ON pf.function\_id=prf.function\_id

**Benefits of partitioning**

* Data partitioning improves the performance of SQL queries, data management, index management and storage management
* Allows you to spread data onto different physical disks, leveraging the concurrent performance of those disks to optimize query performance
* Separate the data into partition based on date and then only index the most current date data
* transfer or access subsets of data quickly and efficiently
* You can perform maintenance operations on one or more partitions more quickly
* operations are more efficient because they target only these data subsets, instead of the whole table
* 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
* Improve performance by enabling lock escalation at the partition level to reduce lock contention on the table
* extracting, moving, removing information from tables with hundreds of millions of rows without creating blockages
* Partitioning is not visible to end users, a partitioned table behaves like one logical table when queried
* Less frequently accessed data can be placed on slower disks and more frequently accessed data can be placed on faster disks.
* Historical, unchanging data can be set to read-only and then be excluded from regular backups.
* If data needs to be restored it is possible to restore the partitions with the most critical data first.

**Partition function**

Partition function defines ranges of data to be partitioned; the last partition function catches collects all data outside the boundaries of the predefined ranges

**Partition scheme**

Partition scheme uses the partition function to map data ranges to appropriate file groups

**Scripts**

USE SQL2

GO

SELECT \* FROM [dbo].[People2]--1000

SELECT \* FROM [dbo].[People3] -- 10,000,000 --2 MIN 10 SECS

SELECT \* FROM [dbo].[People] -- 100,000,000 --LONGER THAN 10,000,000

--DROP DATABASE

--use master

--go

--DROP DATABASE AdventureWorks2012

--GO

----RESTORE DATABASE

USE [master]

RESTORE DATABASE [AdventureWorks2012]

FROM DISK = N'C:\backup122215\Database backups\ADVENTUREWORKS2012.BAK'

WITH FILE = 1,

MOVE N'AdventureWorks2012\_Data'

TO N'C:\Program Files\Microsoft SQL Server\MSSQL12.MSSQLSERVER\MSSQL\DATA\AdventureWorks2012\_Data.mdf',

MOVE N'AdventureWorks2012\_Log'

TO N'C:\Program Files\Microsoft SQL Server\MSSQL12.MSSQLSERVER\MSSQL\DATA\AdventureWorks2012\_log.ldf',

NOUNLOAD,

STATS = 5

GO

---- row count

SELECT COUNT(\*) AS Total\_Rows

FROM AdventureWorks2012.Production.TransactionHistoryArchive

----89253

----Find distinct datetime per year so as to partition

SELECT DISTINCT YEAR(TransactionDate) AS Year, COUNT(\*) AS Total\_Rows

FROM AdventureWorks2012.Production.TransactionHistoryArchive

GROUP BY YEAR(TransactionDate)

ORDER BY 1

--1: Before creating the file groups, create 3 seperate folders

--on drive C: to represent 'physical drives' (drive d, drive e, drive f)

----Create 4 file groups for each year - 2005, 2006, 2007 and all data not in first three partitions

--1 ADD FILEGROUPS FOR EACH YEAR PARTITION

USE [master]

GO

ALTER DATABASE [AdventureWorks2012]

ADD FILEGROUP [DRIVE D]

GO

ALTER DATABASE [AdventureWorks2012]

ADD FILEGROUP [DRIVE E]

GO

ALTER DATABASE [AdventureWorks2012]

ADD FILEGROUP [DRIVE F]

GO

ALTER DATABASE [AdventureWorks2012]

ADD FILEGROUP [DRIVE G]

GO

----With in those four folders (drives d, drive e, drive f, drive g), create 4 seperate data files (.ndf)

----for each year in the table

---- Create a partition function for each year range (note no table has been assigned to any partition as of yet)

----thus, you can associate any table to the partition at this point

USE AdventureWorks2012

GO

SELECT o.name objectname,i.name indexname, partition\_id, partition\_number, [rows]

FROM sys.partitions p

INNER JOIN sys.objects o ON o.object\_id=p.object\_id

INNER JOIN sys.indexes i ON i.object\_id=p.object\_id and p.index\_id=i.index\_id

WHERE o.name LIKE '%TransactionHistoryArchive%'

USE [AdventureWorks2012]

GO

--2: CREATE 4 DATA (NDF)FILES IN FILEGROUP FOR EACH PARTITION

USE [master]

GO

ALTER DATABASE [AdventureWorks2012]

ADD FILE ( NAME = N'TRANS2005',

FILENAME = N'C:\drive d\TRANS2005.ndf' ,

SIZE = 4096KB ,

FILEGROWTH = 1024KB )

TO FILEGROUP [DRIVE D]

GO

ALTER DATABASE [AdventureWorks2012]

ADD FILE ( NAME = N'TRANS2006',

FILENAME = N'C:\drive e\TRANS2006.ndf' ,

SIZE = 4096KB ,

FILEGROWTH = 1024KB )

TO FILEGROUP [DRIVE E]

GO

ALTER DATABASE [AdventureWorks2012]

ADD FILE ( NAME = N'TRANS2007',

FILENAME = N'C:\drive f\TRANS2007.ndf' ,

SIZE = 4096KB ,

FILEGROWTH = 1024KB )

TO FILEGROUP [DRIVE F]

GO

ALTER DATABASE [AdventureWorks2012]

ADD FILE ( NAME = N'TRANS2015',

FILENAME = N'C:\drive g\TRANS2015.ndf' ,

SIZE = 4096KB ,

FILEGROWTH = 1024KB )

TO FILEGROUP [DRIVE G]

GO

--AT THIS POINT WE HAVE CREATED THE 'CONTAINERS' AND FILES FOR THE TABLE TO BE PARTITIONED TOO!!

-- CREATE PARTITION ON TABLE [TransactionHistoryArchive]

USE [AdventureWorks2012]

GO

BEGIN TRANSACTION

CREATE PARTITION FUNCTION [FUNCTION\_TRANSHISTORY](datetime)

AS RANGE LEFT

FOR VALUES (N'2005-12-31T23:59:59.997', N'2006-12-31T23:59:59.997', N'2007-12-31T23:59:59.997')

CREATE PARTITION SCHEME [SCHEMA\_TRANSHISTORY]

AS PARTITION [FUNCTION\_TRANSHISTORY]

TO ([DRIVE D], [DRIVE E], [DRIVE F], [PRIMARY])

ALTER TABLE [Production].[TransactionHistoryArchive] DROP CONSTRAINT [PK\_TransactionHistoryArchive\_TransactionID]

ALTER TABLE [Production].[TransactionHistoryArchive] ADD CONSTRAINT [PK\_TransactionHistoryArchive\_TransactionID] PRIMARY KEY NONCLUSTERED

([TransactionID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, IGNORE\_DUP\_KEY = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON)

CREATE CLUSTERED INDEX [ClusteredIndex\_on\_SCHEMA\_TRANSHISTORY\_635922165997540276] ON [Production].[TransactionHistoryArchive]

([TransactionDate]

)WITH (SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF) ON [SCHEMA\_TRANSHISTORY]([TransactionDate])

DROP INDEX [ClusteredIndex\_on\_SCHEMA\_TRANSHISTORY\_635922165997540276] ON [Production].[TransactionHistoryArchive]

COMMIT TRANSACTION

--VERIFY

USE AdventureWorks2012

GO

SELECT o.name objectname,i.name indexname, partition\_id, partition\_number, [rows]

FROM sys.partitions p

INNER JOIN sys.objects o ON o.object\_id=p.object\_id

INNER JOIN sys.indexes i ON i.object\_id=p.object\_id and p.index\_id=i.index\_id

WHERE o.name LIKE '%TransactionHistoryArchive%'

-- INSERT DATA TO VERIFY DIRECTION OF DATA TO PARTITION

USE [AdventureWorks2012]

GO

INSERT INTO [Production].[TransactionHistoryArchive]

([TransactionID]

,[ProductID]

,[ReferenceOrderID]

,[ReferenceOrderLineID]

,[TransactionDate]

,[TransactionType]

,[Quantity]

,[ActualCost]

,[ModifiedDate])

VALUES

(89254, 1,1,1,'2016-02-28 00:00:00.000','P',999,50.2600,'2016-02-28 00:00:00.000')

GO

--VIEW ACTULA DATA IN THE PARTITION

SELECT \* FROM [Production].[TransactionHistoryArchive]

WHERE $PARTITION.[FUNCTION\_TRANSHISTORY](TransactionDate) = 4 ;--<< PARTITION 4 CONTAIN ONLY DATA THAT IS NOT IN RANGE OF PARTITION 1,2,3

[DRIVE D]

[DRIVE E]

[DRIVE F]

[DRIVE G]

TRANS2005

TRANS2006

TRANS2007

TRANS2015

2005-12-31T23:59:59.997

2006-12-31T23:59:59.997

2007-12-31T23:59:59.997