Advanced Databases

Hands-On Lab

Performance Tunning

Lab version: 1.0.0

Last updated: 11/5/2020

Task 1 – Transfer a table to a new file group

1. Identify in which filegroup the table exists on

```
SELECT  tbl.name AS [Table Name],
          CASE WHEN dsidx.type='FG' THEN dsidx.name ELSE '(Partitioned)' END AS [File
Group]
FROM    sys.tables AS tbl
              JOIN      sys.indexes AS idx ON idx.object_id = tbl.object_id AND
idx.index_id <= 1
              LEFT JOIN sys.data_spaces AS dsidx ON dsidx.data_space_id =
idx.data_space_id
ORDER BY  [File Group], [Table Name]
```

2. [Optional] Create a new file group

```
USE Master
ALTER DATABASE AdventureWorks2014
ADD FILEGROUP NewFileGroup;
```

3. Create a new data file

```
ALTER DATABASE AdventureWorks2014
ADD FILE
(
    NAME = Test1dat2,
    FILENAME = 'D:\DBFiles\t1dat2.ndf',
    SIZE = 5MB,
    MAXSIZE = 100MB,
    FILEGROWTH = 5MB
)
TO FILEGROUP NewFileGroup;--Skip this line to add file to the default FG
```

4. Move the table to the new file group

```
USE AdventureWorks2014
CREATE UNIQUE CLUSTERED INDEX PK_Department_DepartmentID
ON HumanResources.Department(DepartmentID)
WITH (DROP_EXISTING=ON,ONLINE=ON) ON NewFileGroup
```

5. Check the file group of tables with script in step 1

Task 2 – Split table vertically

1. Create a table of 10000 rows

```
CREATE DATABASE TablePartitioning

CREATE TABLE Employee
(
EmployeeID int IDENTITY (1,1) NOT NULL,
FirstName varchar(100),
LastName varchar(100),
Picture image,
CONSTRAINT E_PK PRIMARY KEY CLUSTERED (EmployeeID)
)

CREATE TABLE RandomNames
(id int, name varchar(100), gender char(1))
```

```
INSERT INTO RandomNames
(id, name,gender)
select 1,'Bill','M'
union
select 2,'John','M'
union
select 3,'Steve','M'
union
select 4,'Mike','M'
union
select 5,'Phil','M'
union
select 6,'Sarah','F'
union
select 7,'Ann','F'
union
select 8,'Marie','F'
union
select 9,'Liz','F'
union
select 10,'Stephanie','F'

DECLARE @i int
SET @i = 1
WHILE @i <=10000
BEGIN
	DECLARE @fName varchar(100)
	DECLARE @lName varchar (100)

	SELECT @fName = name
	FROM RandomNames
	WHERE id =  (CAST(RAND()*100 as int) % 10) +1

	SELECT @lName = name
	FROM RandomNames
	WHERE id =  (CAST(RAND()*100 as int) % 10) +1

	INSERT INTO Employee (FirstName, LastName, Picture)
	SELECT  @fName, @lName,
	BulkColumn FROM Openrowset( Bulk 'D:\DBFiles\avatar.png', Single_Blob) as
EmployeePicture

	SET @i=@i+1
END
```

2. Create a table Emp(EmployeeID, FirstName, LastName) consisting first 3 columns of Employee

```
CREATE TABLE Emp
(
EmployeeID int IDENTITY (1,1) NOT NULL,
FirstName varchar(100),
LastName varchar(100),
CONSTRAINT Emp_PK PRIMARY KEY CLUSTERED (EmployeeID)
)
```

```
SET IDENTITY_INSERT dbo.Emp ON
INSERT INTO Emp
(
    EmployeeID,
    FirstName,
    LastName
)
SELECT  EmployeeID,    FirstName,    LastName
FROM Employee
          SET IDENTITY_INSERT dbo.Emp OFF
```

3. Create a table EmpPicture(EmployeeID, Picture)

```
CREATE TABLE EmpPicture
(
EmployeeID int IDENTITY (1,1) NOT NULL,
Picture image,
)

SET IDENTITY_INSERT dbo.EmpPicture ON
INSERT INTO EmpPicture
(
    EmployeeID, Picture
)
SELECT  EmployeeID, Picture
FROM Employee
          SET IDENTITY_INSERT dbo.EmpPicture OFF
```

4. Run queries and observe then compare and explain the statistics of them

```
SET STATISTICS IO ON
SET STATISTICS TIME ON
SELECT EmployeeID, FirstName, LastName
FROM Employee
SET STATISTICS IO OFF
SET STATISTICS TIME OFF

GO

SET STATISTICS IO ON
SET STATISTICS TIME ON
SELECT EmployeeID, FirstName, LastName
FROM Emp
SET STATISTICS IO OFF
          SET STATISTICS TIME OFF
```

Task 3 – Split table horizontally

1. Define file groups and files for partitions

```
USE TablePartitioning
GO

ALTER DATABASE TablePartitioning
ADD FILEGROUP FirstGroup
```

```
ALTER DATABASE TablePartitioning
ADD FILEGROUP SecondGroup


ALTER DATABASE TablePartitioning
    ADD FILE
    (
    NAME = [FirstPart],
    FILENAME = 'C:\DBFiles\FirstPart.ndf',
        SIZE = 3072 KB,
        MAXSIZE = UNLIMITED,
        FILEGROWTH = 1024 KB
    ) TO FILEGROUP  FirstGroup

ALTER DATABASE TablePartitioning
    ADD FILE
    (
    NAME = [SecondPart],
    FILENAME = 'C:\DBFiles\SecondPart.ndf',
        SIZE = 3072 KB,
        MAXSIZE = UNLIMITED,
        FILEGROWTH = 1024 KB
    ) TO FILEGROUP SecondGroup
```

2. Define function and scheme for partitioning

```
CREATE PARTITION FUNCTION [PFPartitioningByID] (int)
AS RANGE RIGHT FOR VALUES (5000);

CREATE PARTITION SCHEME myPartitionScheme
AS PARTITION [PFPartitioningByID] TO (FirstGroup, SecondGroup)

DROP PARTITION FUNCTION [PFPartitioningByID]
DROP PARTITION SCHEME myPartitionScheme
```

3. Define new tables and transfer data to it

```
CREATE TABLE EmpHoziPart
(
EmployeeID int IDENTITY (1,1) NOT NULL,
FirstName varchar(100),
LastName varchar(100),
Picture image,
CONSTRAINT EmpHoziPart_PK PRIMARY KEY CLUSTERED (EmployeeID)
) ON myPartitionScheme(EmployeeID)


SET IDENTITY_INSERT dbo.EmpHoziPart ON
INSERT INTO EmpHoziPart
(EmployeeID,    FirstName,    LastName, Picture)
SELECT  EmployeeID,    FirstName,    LastName, Picture
FROM Employee
SET IDENTITY_INSERT dbo.EmpHoziPart OFF
```

4. See how many rows each partition holds to verify the results

```
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) = 'EmpHoziPart'
```

Task 4 – Define Indexed view

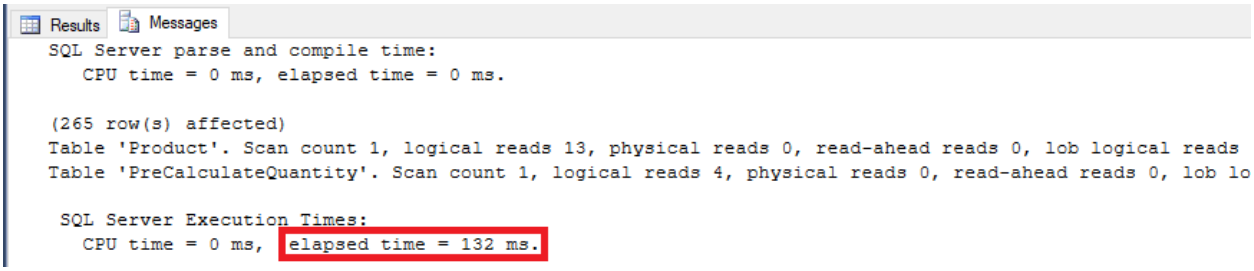
Suppose we want to make a report on purchased products like the below figure

	Name	OrderQty	ReceivedQty	RejectedQty	Count
1	Adjustable Race	154	153.00	3.00	51
2	Bearing Ball	150	150.00	3.00	50
3	Headset Ball Bearings	153	152.00	1.00	51
4	LL Crankarm	44000	43262.00	3102.00	80
5	ML Crankarm	44000	43784.00	2805.00	80
6	HL Crankarm	71500	71172.00	1178.00	130
7	Chainring Bolts	375	375.00	3.00	125
8	Chainring Nut	375	375.00	3.00	125

1. Normally, we do by following query. Please pay attention to the execution time

```
SET STATISTICS TIME ON
SET STATISTICS IO ON

SELECT P.Name, POD.OrderQty, POD.ReceivedQty, POD.RejectedQty, POD.Count
FROM Production.Product P JOIN (
                    SELECT ProductID, SUM(OrderQty) OrderQty, SUM(ReceivedQty)
ReceivedQty, SUM(RejectedQty) RejectedQty, COUNT_BIG(*) AS Count
                    FROM Purchasing.PurchaseOrderDetail
                    GROUP BY ProductID) POD ON P.ProductID = POD.ProductID
```



2. We can speed up the query by pre-calculate aggregation value with following indexed view

```
CREATE VIEW Purchasing.PreCalculateQuantity
WITH SCHEMABINDING
AS
SELECT ProductID, SUM(OrderQty) OrderQty, SUM(ReceivedQty) ReceivedQty, SUM(RejectedQty)
RejectedQty, COUNT_BIG(*) AS Count
FROM Purchasing.PurchaseOrderDetail
GROUP BY ProductID
GO
CREATE UNIQUE CLUSTERED INDEX IX_Purchasing ON Purchasing.PreCalculateQuantity(ProductID)
```

3. Re-make the report but use the indexed view instead of the base table

```
SET STATISTICS TIME ON
SET STATISTICS IO ON

SELECT P.Name, POD.OrderQty, POD.ReceivedQty, POD.ReceivedQty, POD.ReceivedQty, POD.Count
FROM Production.Product P JOIN Purchasing.PreCalculateQuantity POD ON P.ProductID =
POD.ProductID
GO
```

```
(265 row(s) affected)
Table 'Product'. Scan count 1, logical reads 13, physical reads 0, read-ahead reads 0, lob logical read
Table 'PreCalculateQuantity'. Scan count 1, logical reads 4, physical reads 0, read-ahead reads 0, lob

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 53 ms.
```

4. Capture your execution message and explain the different elapsed times of two queries

Task 5 – Select an appropriate isolation level

1. Suppose that a user is running a transaction manipulating data

```
BEGIN TRANSACTION

DECLARE @i int
SET @i = 1
WHILE @i <=50000
BEGIN
	DECLARE @fName varchar(100)
	DECLARE @lName varchar (100)

	SELECT @fName = name
	FROM RandomNames
	WHERE id =  (CAST(RAND()*100 as int) % 10) +1

	SELECT @lName = name
	FROM RandomNames
	WHERE id =  (CAST(RAND()*100 as int) % 10) +1

	INSERT INTO Employee (FirstName, LastName, Picture)
	SELECT  @fName, @lName,
	BulkColumn FROM Openrowset( Bulk 'C:\DBFiles\avatar.png', Single_Blob) as
EmployeePicture

	SET @i=@i+1
END
ROLLBACK TRANSACTION
```

2. Another user wants to run a query on changing data. By default, the ISOLATION LEVEL is READ COMMITTED so this user must wait because the resource is locked.

```
SELECT COUNT(*)
FROM Employee
WHERE FirstName = 'Steve'
```

3. The 2nd user still can run the query if he/she uses a lower ISOLATION LEVEL

```
SET TRAN ISOLATION LEVEL READ UNCOMMITTED
```

```sql
SELECT COUNT(*)
FROM Employee
WHERE FirstName = 'Steve'
```

Assignment for students

Submit a word document with the file name like NGUYEN_VAN_A.doc to complete following missions:

1. Task 1: Capture your screen to prove that you successfully transfer the table to a new file group.
2. Task 2: Write a view to reconstruct the logical original table from two partitioned tables
3. Task 3: Capture your screen of step 4