Implementing In-Memory SQL Database Objects

Lab 2 – Creating Columnstore Indexes

Overview

You plan to improve the performance of the AdventureWorksDW data warehouse by using columnstore indexes. You need to improve the performance of queries that use the FactProductInventory tables without causing any database downtime, or dropping any existing indexes. Disk usage for this table is not an issue.

You must retain the existing indexes on the FactProductInventory table, and ensure you do not impact current applications by any alterations you make.

Before starting this lab, you should view **Module 2 – Implementing Columnstore Indexes** in the course *Implementing In-Memory SQL Database Objects*. Then, if you have not already done so, follow the instructions in the **Getting Started** document for this course to set up the lab environment.

If you find some of the challenges difficult, don't worry – you can find suggested solutions for all of the challenges in the **Lab Solution** folder for this module.

What You'll Need

To complete the labs, you will need the following:

- A SQL Server instance with the AdventureWorksDW sample database. Review the Getting Started document for information about how to provision this.
- The lab files for this course

Challenge 1: Create a Columnstore Index

In this exercise, you will create a columnstore index and improved the performance of an analytical query. This will have been done in real time without impacting transactional processing.

Examine the Existing Size of the FactProductInventory Table and Query Performance

1. In SQL Server Management Studio, type and execute the following query to find the size of the table. The queries for this task are also available in the **Setup** folder, in the **Query FactProductInventory.sql** file.

```
SET STATISTICS TIME OFF

GO

USE AdventureWorksDW

GO


sp_spaceused 'dbo.FactProductInventory';

GO
```

2. Configure SQL Server Management Studio to include the actual execution plan.
3. Type and execute the following query to find the execution plan and index use:

```
SET STATISTICS TIME ON

GO


SELECT p.EnglishProductName

        ,d.WeekNumberOfYear

        ,d.CalendarYear

        ,AVG(fpi.UnitCost) AvgCost

        ,SUM(fpi.UnitsOut) TotalUnits

        ,MAX(fpi.UnitCost) HighestPrice

FROM dbo.FactProductInventory as fpi

INNER JOIN dbo.DimProduct as p ON fpi.ProductKey = p.ProductKey

INNER JOIN dbo.DimDate as d ON fpi.DateKey = d.DateKey

GROUP BY p.EnglishProductName,

        d.WeekNumberOfYear,

        d.CalendarYear

ORDER BY p.EnglishProductName,
```

```
        d.CalendarYear,
        d.WeekNumberOfYear;
GO
```

4. Review the execution plan, making a note of the indexes used, the execution time, and disk space used.

Create a Columnstore Index on the FactProductInventory Table

1. Based on the scenario for this exercise, decide whether a clustered or non-clustered columnstore index is appropriate for the **FactProductInventory** table.
2. Create the required columnstore index. Re-execute the query to verify that the new columnstore index is used, along with existing indexes.
3. What, if any, are the disk space and query performance improvements?

Examine the Existing Size of the FactInternetSales Table and Query Performance

1. In SQL Server Management Studio, type and execute the following query to find the size of the table. The queries for this task are also available in the **Setup** folder, in the **Query FactInternetSales.sql** file.

```
SET STATISTICS TIME OFF
GO
USE [AdventureWorksDW]
GO


sp_spaceused 'dbo.FactInternetSales'
GO
```

2. Configure SQL Server Management Studio to include the actual execution plan.
3. Type and execute the following query to find the execution plan and index use:

```
SET STATISTICS TIME ON
GO


SELECT SalesTerritoryRegion
        ,p.EnglishProductName
        ,d.WeekNumberOfYear
        ,d.CalendarYear
        ,SUM(fi.SalesAmount) Revenue
        ,AVG(OrderQuantity) AverageQuantity
```

```
        ,STDEV(UnitPrice) PriceStandardDeviation
        ,SUM(TaxAmt) TotalTaxPayable
    FROM dbo.FactInternetSales as fi
    INNER JOIN dbo.DimProduct as p ON fi.ProductKey = p.ProductKey
    INNER JOIN dbo.DimDate as d ON fi.OrderDate = d.FullDateAlternateKey
    INNER JOIN dbo.DimSalesTerritory as st on fi.SalesTerritoryKey = st.SalesTerritoryKey
        AND fi.OrderDate BETWEEN '1/1/2007' AND '12/31/2007'
    GROUP BY SalesTerritoryRegion, d.CalendarYear, d.WeekNumberOfYear, p.EnglishProductName
    ORDER BY SalesTerritoryRegion, SUM(fi.SalesAmount) desc;
```

4. Review the execution plan, making a note of the indexes used, the execution time, and disk space used.

Create a Columnstore Index on the FactInternetSales Table

1. Based on the scenario for this exercise, decide whether a clustered or non-clustered columnstore index is appropriate for the **FactInternetSales** table.
2. In the **Setup** folder open **Create Columnstore Index on FactInternetSales.sql**.
3. Review the contents of the file, copy the T-SQL to a new query in SQL Server Management Studio, and execute the query.
4. Re-execute the original query to verify that the new columnstore index is used, along with the existing indexes.
5. What, if any, are the disk space and query performance improvements?