SE 308 Term Project 2

Onur Akalın 170706018 & Ahmet Arif Özçelik 170706016

Introduction

We worked 2 people in this performance experiment, so we got different results from different computers. In this way, we think our results are more accurate. We developed our program using .Net Core, ADO.Net and MSSQL.

Project

We took our before query measurements in 2 different computer and We noticed that these queries run at different performance speeds on different computers and that the gap between them is huge, nearly 10 seconds. We are using different OS and system environments so I think, It is acceptable results for us.

Average Times Before Indexıng (Second)

	Onur Akalın	Ahmet Arif Özçelik
Query 1	14.768 s	21.383 s
Query 2	9.796 s	16.108 s
Query 3	9.450 s	14.563 s

Total Times Before Indexıng (Minute)

	Onur Akalın	Ahmet Arif Özçelik
Query 1	24.61 m	35.63 m
Query 2	16.32 m	26.84 m
Query 3	15.75 m	24.27 m

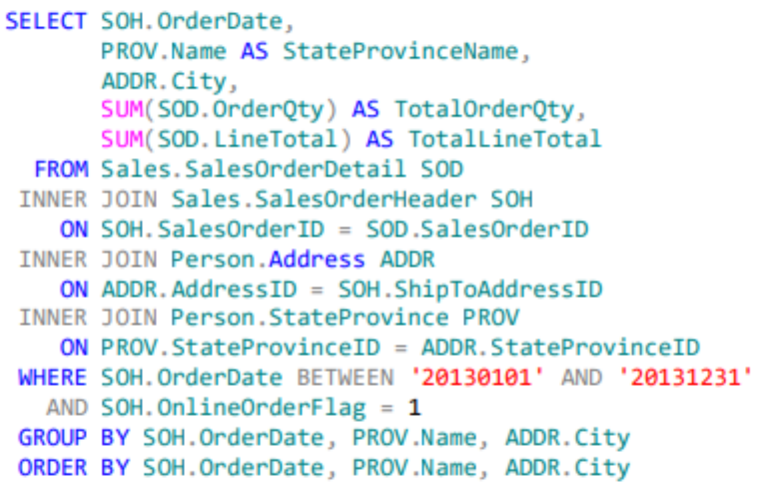
In our project, We run these queries totally 10.000 times, We defined 1 unit as 100 queries and execute totally 100 unit on each different query.

Index

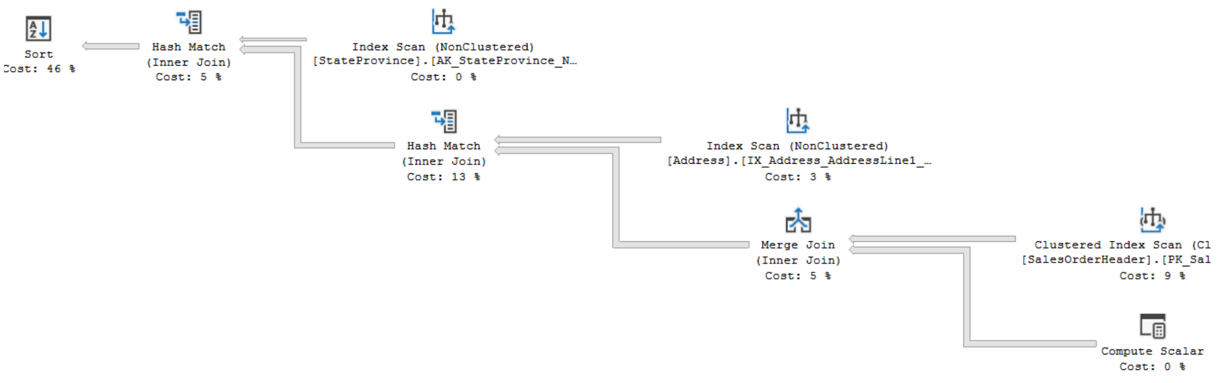
We are aimed to reduce cost of Indexes and transform them to Seeks to speed up executions but sometimes even if We reduce the costs, speed of execution would be decreased drastically so that We gave up some index solutions.

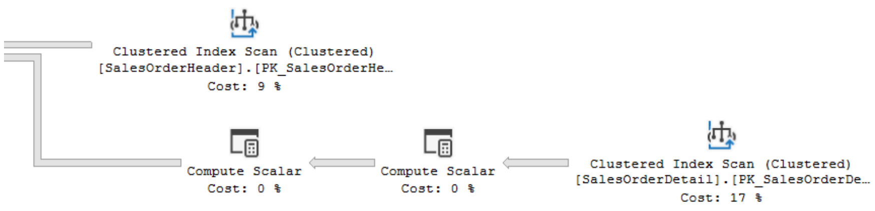
While creating indexes for each query, we deleted the indexes we created before. Since the indexes are similar to each other, we have prevented the undesired use of indexes in this way.

Query-1:

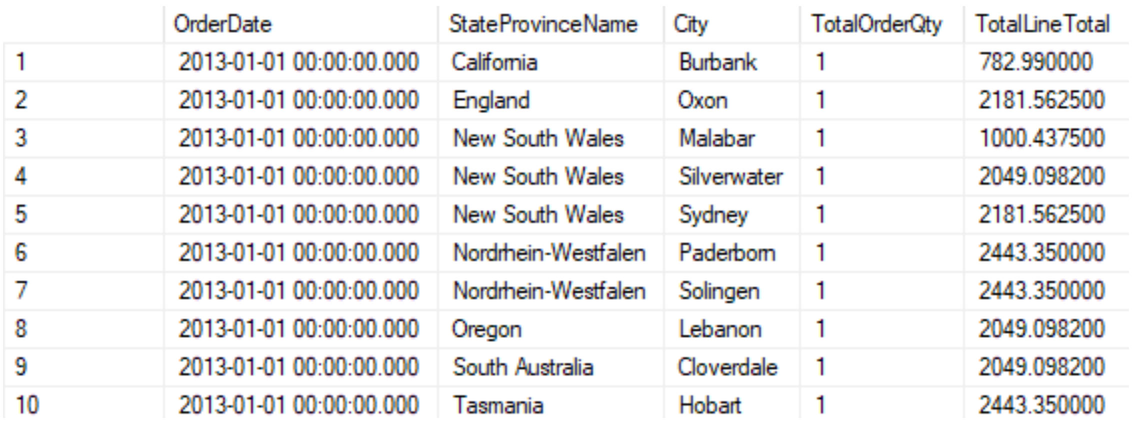


```sql
SELECT SOH.OrderDate,
       PROV.Name AS StateProvinceName,
       ADDR.City,
       SUM(SOD.OrderQty) AS TotalOrderQty,
       SUM(SOD.LineTotal) AS TotalLineTotal
  FROM Sales.SalesOrderDetail SOD
 INNER JOIN Sales.SalesOrderHeader SOH
    ON SOH.SalesOrderID = SOD.SalesOrderID
 INNER JOIN Person.Address ADDR
    ON ADDR.AddressID = SOH.ShipToAddressID
 INNER JOIN Person.StateProvince PROV
    ON PROV.StateProvinceID = ADDR.StateProvinceID
 WHERE SOH.OrderDate BETWEEN '20130101' AND '20131231'
   AND SOH.OnlineOrderFlag = 1
 GROUP BY SOH.OrderDate, PROV.Name, ADDR.City
 ORDER BY SOH.OrderDate, PROV.Name, ADDR.City
```




Clustered Index Scan (Clustered)
[SalesOrderHeader].[PK_SalesOrderHe…
Cost: 9 %
Compute Scalar
Cost: 0 %
Compute Scalar
Cost: 0 %
Clustered Index Scan (Clustered)
[SalesOrderDetail].[PK_SalesOrderDe…
Cost: 17 %

TOTAL RETURNED ROWS IS 10899



	OrderDate	StateProvinceName	City	TotalOrderQty	TotalLineTotal
1	2013-01-01 00:00:00.000	California	Burbank	1	782.990000
2	2013-01-01 00:00:00.000	England	Oxon	1	2181.562500
3	2013-01-01 00:00:00.000	New South Wales	Malabar	1	1000.437500
4	2013-01-01 00:00:00.000	New South Wales	Silverwater	1	2049.098200
5	2013-01-01 00:00:00.000	New South Wales	Sydney	1	2181.562500
6	2013-01-01 00:00:00.000	Nordrhein-Westfalen	Paderborn	1	2443.350000
7	2013-01-01 00:00:00.000	Nordrhein-Westfalen	Solingen	1	2443.350000
8	2013-01-01 00:00:00.000	Oregon	Lebanon	1	2049.098200
9	2013-01-01 00:00:00.000	South Australia	Cloverdale	1	2049.098200
10	2013-01-01 00:00:00.000	Tasmania	Hobart	1	2443.350000

1- Firstly, We want to create index for SalesOrderHeader table because It's affect nearly all parts of query.

- **SOH.OrderDate :** This column was in Select , Where , Group By and Order By parts so that it is important for reduce the costs.
- **SOH.OnlineOrderFlag :** This column was in Where and it is use for equation so that this part will be first column in index.
- **SOH.ShipToAddressID :** This column was in right sides of Nested Inner Join so that It's not important as before 2 columns and It will be in include part.
- **SOH.SalesOrderID:** We don't include this column to the index because It's primary key and It's have already have Clustered Index.

```
CREATE INDEX Q1IndexOnlineOrderFlagANDOrderDateWithShipToAddressId
  ON Sales.SalesOrderHeader (OnlineOrderFlag, OrderDate)
  INCLUDE (ShipToAddressID)
```

***OnlineOrderFlag(Equality Column) and OrderDate(Inequality Column) with including ShipToAddressID(Foreign Key for Join) speeds up our query by allowing us to access only the necessary records (join) via index.**

2- We want to created index for SalesOrderDetail table because it's affect on one Nested Inner Join and Select part

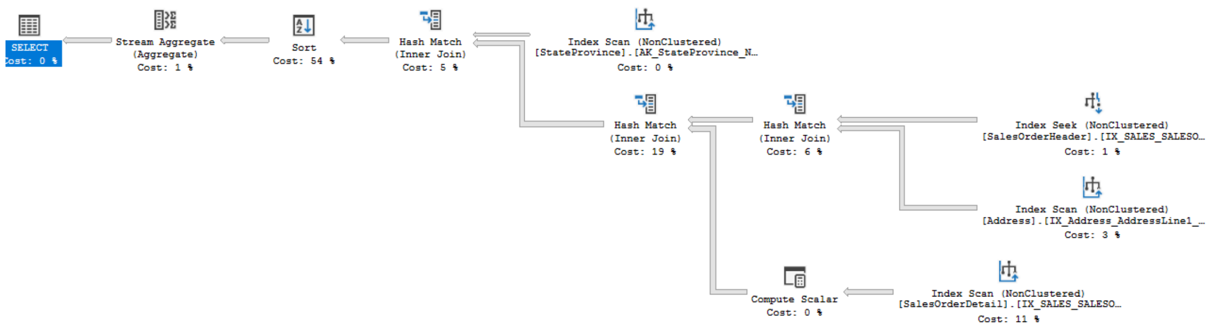
- **SOD.SalesOrderID:** SalesOrderID is primary key so that It has index but in this query, It's not enough. We want to create a new Non-Clustered index with some includes.
- **SOD.OrderQty:** OrderQTY is used in Select part so that It's not important as SalesOrderID, It will be placed in Include part.
- **SOD.LineTotal:** LineTotal is used in Select part so that It's not important as SalesOrderID, It will be placed in Include part.

```
CREATE INDEX Q1ProductIdWithOrderQtyAndLineTotal
   ON Sales.SalesOrderDetail (ProductID)
   INCLUDE (OrderQty, LineTotal)
```

***If we keep the 2 required columns while doing the index seek over the primary key, this prevents us from navigating the table again for the required fields.**

3- We don’t add any indexes for **PROV.Name** and **ADDR.City** because ADDR.City is become 0% cost after some indexes are added and PROV.Name has very low cost and has no affect on query with premade indexes for it.

After Indexing


SELECT
Cost: 0 %
Stream Aggregate
(Aggregate)
Cost: 1 %
Sort
Cost: 54 %
Hash Match
(Inner Join)
Cost: 5 %
Index Scan (NonClustered)
[StateProvince].[AK_StateProvince_N…
Cost: 0 %
Hash Match
(Inner Join)
Cost: 19 %
Hash Match
(Inner Join)
Cost: 6 %
Index Seek (NonClustered)
[SalesOrderHeader].[IX_SALES_SALESO…
Cost: 1 %
Index Scan (NonClustered)
[Address].[IX_Address_AddressLine1_…
Cost: 3 %
Compute Scalar
Cost: 0 %
Index Scan (NonClustered)
[SalesOrderDetail].[IX_SALES_SALESO…
Cost: 11 %

Average times after Indexes

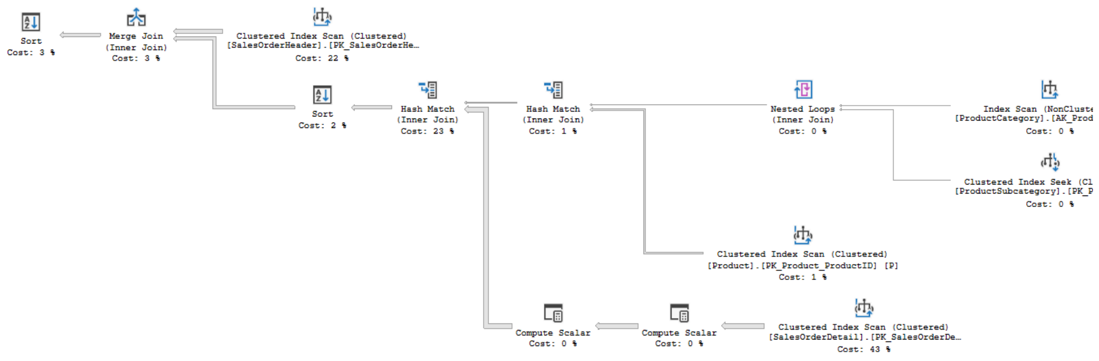
	Onur Akalın	Ahmet Arif Özçelik
Query 1	13.49 s	19.98 s
Remain	+1.278 s	+1.403 s

Total times after Indexes

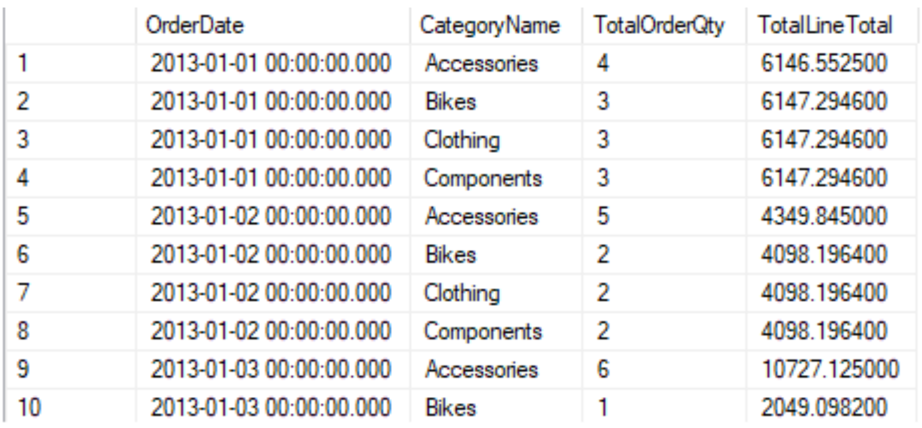
	Onur Akalın	Ahmet Arif Özçelik
Query 1	22.48 m	33.30 m
Remain	+2.13 m	+2.33 m

Query-2 (Failure):

```sql
SELECT SOH.OrderDate,
       CAT.Name as CategoryName,
       SUM(SOD.OrderQty) AS TotalOrderQty,
       SUM(SOD.LineTotal) AS TotalLineTotal
  FROM Sales.SalesOrderDetail SOD
 INNER JOIN Sales.SalesOrderHeader SOH
    ON SOH.SalesOrderID = SOD.SalesOrderID
 INNER JOIN Production.Product P
    ON P.ProductID = SOD.ProductID
 INNER JOIN Production.ProductSubcategory SUBCAT
    ON SUBCAT.ProductCategoryID = P.ProductSubcategoryID
 INNER JOIN Production.ProductCategory CAT
    ON CAT.ProductCategoryID = SUBCAT.ProductSubcategoryID
 WHERE SOH.OrderDate BETWEEN '20130101' AND '20131231'
   AND SOH.OnlineOrderFlag = 1
   AND (P.MakeFlag = 1 OR P.FinishedGoodsFlag = 1)
   AND P.Color IN ('Black', 'Yellow')
 GROUP BY SOH.OrderDate, CAT.Name
 ORDER BY SOH.OrderDate, CAT.Name
```



TOTAL RETURNED ROWS IS 1360



	OrderDate	CategoryName	TotalOrderQty	TotalLineTotal
1	2013-01-01 00:00:00.000	Accessories	4	6146.552500
2	2013-01-01 00:00:00.000	Bikes	3	6147.294600
3	2013-01-01 00:00:00.000	Clothing	3	6147.294600
4	2013-01-01 00:00:00.000	Components	3	6147.294600
5	2013-01-02 00:00:00.000	Accessories	5	4349.845000
6	2013-01-02 00:00:00.000	Bikes	2	4098.196400
7	2013-01-02 00:00:00.000	Clothing	2	4098.196400
8	2013-01-02 00:00:00.000	Components	2	4098.196400
9	2013-01-03 00:00:00.000	Accessories	6	10727.125000
10	2013-01-03 00:00:00.000	Bikes	1	2049.098200

1- Firstly, We want to create index for SalesOrderHeader table because It's affect nearly all parts of query.

- **SOH.OrderDate :** This column was in Select , Where , Group By and Order By parts so that it is important for reduce the costs and increase speed.
- **SOH.OnlineOrderFlag :** This column was in Where and it is use for equation so that this part will be first column in index, it's important for index to work fast and properly.
- **SOH.ShipToAddressID :** This column was in right sides of Nested Inner Join so that It's not important as before 2 columns and It will be in include part.
- **SOH.SalesOrderID:** We don't include this column to the index because It's primary key and It's have already have Clustered Index.

```
CREATE INDEX Q2IndexOnlineOrderFlagANDOrderDateWithShipToAddressId
   ON Sales.SalesOrderHeader (OnlineOrderFlag, OrderDate)
   INCLUDE (ShipToAddressID)
```

***OnlineOrderFlag(Equality Column) and OrderDate(Inequality Column) with including ShipToAddressID(Foreign Key for Join) speeds up our query by allowing us to access only the necessary records (join) via index.**

2- We want to create index for SalesOrderDetail table because it's affect on one Nested Inner Join and Select part

- **SOD.SalesOrderID:** SalesOrderID is primary key so that It has index but in this query, It's not enough. We want to create a new Non-Clustered index with some includes to make faster.
- **SOD.OrderQty:** OrderQTY is used in Select part so that It's not important as SalesOrderID, It will be placed in Include part.
- **SOD.LineTotal:** LineTotal is used in Select part so that It's not important as SalesOrderID, It will be placed in Include part.

```
CREATE INDEX Q2ProductIdWithOrderQtyAndLineTotal
   ON Sales.SalesOrderDetail (ProductID)
   INCLUDE (OrderQty, LineTotal)
```

***If we keep the 2 required columns while doing the index seek over the primary key, this prevents us from navigating the table again for the required fields.**

3- We want to create index for Product table because it's affect on one Nested Inner Join and Select part.

- **P.MakeFlag:** MakeFlag column was in Where and it is use for equation so that this part will be first column in index.
- **P.FinishedGoodsFlag:** FinishedGoodsFlag column was in Where and it is use for equation so that this part will be second column in index.
- **P.Color :** Color column was in Where and it is use for equation so that this part will be third column in index.

- **P. ProductSubcategoryID:** ProductSubcategoryID column was in right sides of Nested Inner Join so that It's not important as before 3 columns and It will be in include part.

```
CREATE INDEX Q2MakeFlagAndFinishedGoodsFlagAndColorWithProductSubcategoryID
    ON [Production].[Product] (MakeFlag DESC , FinishedGoodsFlag DESC, Color DESC  )
    INCLUDE (ProductSubcategoryID)
```

***This index will be reduce the sorting load of the process and execute the data in the table more faster with sorted data.**

4- We don't create index for **CAT.NAME** because It's have 0% cost and has no affect on query with premade indexes for it.

Average times after Indexes

	Onur Akalın	**Ahmet Arif Özçelik**
Query 2	11.036 s	18.253 s
Remain	-1.240 s	-2.145 s

Total times after Indexes

	Onur Akalın	**Ahmet Arif Özçelik**
Query 2	18.39 m	30.42 m
Remain	-2.07 m	-3.58 m

(When we add Index for 2 main columns (SalesOrderDetail (43% Cost) and SalesOrderHeader (22% Cost)), It reduce the speed drastically even their costs are decreased. We drop or change indexes to reduce the costs and improve speed but It didn't work.)

*We improved speed only with 3th index on this document.

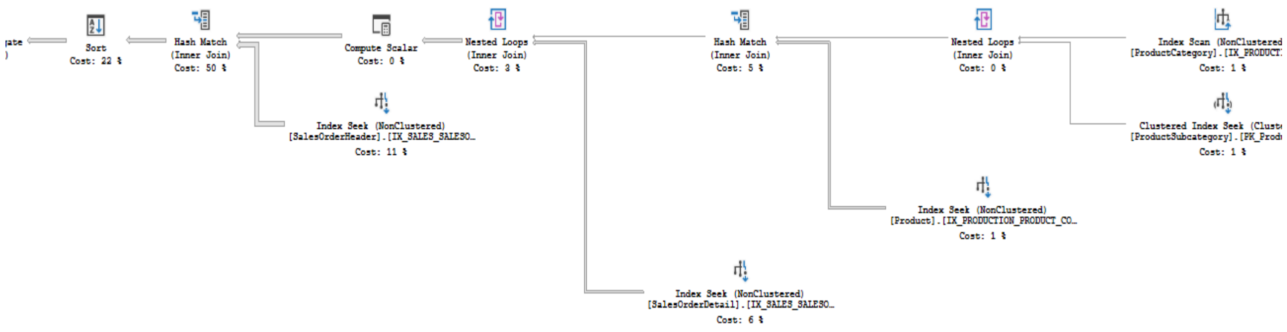
Average times after Indexes (only with 3th index)

	Onur Akalın	**Ahmet Arif Özçelik**
Query 2	8.128 s	14.126 s
Remain	+1.668 s	+1.982 s

Total times after Indexes (only with 3th index)

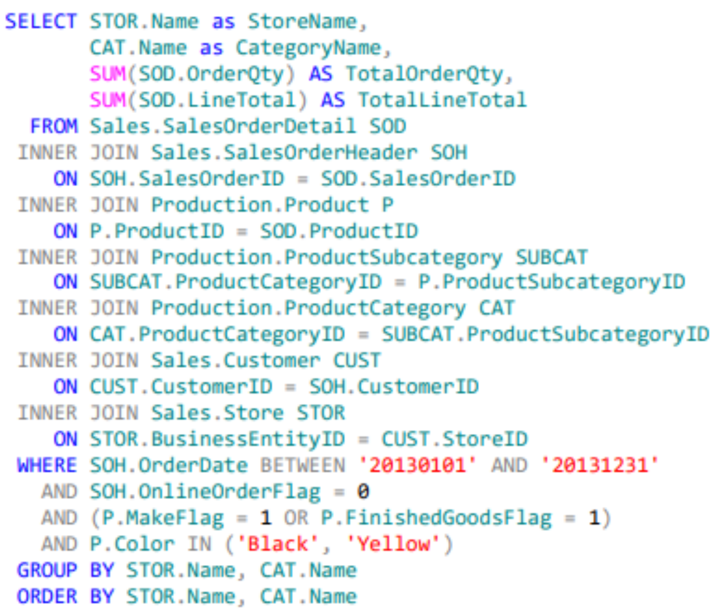
	Onur Akalın	Ahmet Arif Özçelik
Query 2	13.54 m	23.54 m
Remain	+3.18 m	+3.30 m

After Indexing

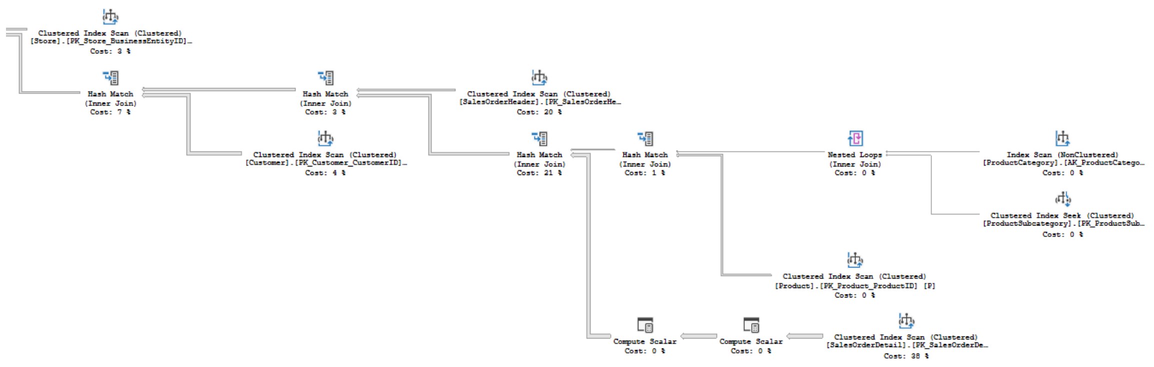


Sort
Cost: 22 %
Hash Match
(Inner Join)
Cost: 50 %
Compute Scalar
Cost: 0 %
Index Seek (NonClustered)
[SalesOrderHeader].[IX_SALES_SALESO…
Cost: 11 %
Nested Loops
(Inner Join)
Cost: 3 %
Hash Match
(Inner Join)
Cost: 5 %
Index Seek (NonClustered)
[SalesOrderDetail].[IX_SALES_SALESO…
Cost: 6 %
Nested Loops
(Inner Join)
Cost: 0 %
Index Seek (NonClustered)
[Product].[IX_PRODUCTION_PRODUCT_CO…
Cost: 1 %
Index Scan (NonClustered
[ProductCategory].[IX_PRODUCT
Cost: 1 %
Clustered Index Seek (Clust
[ProductSubcategory].[PK_Prod
Cost: 1 %

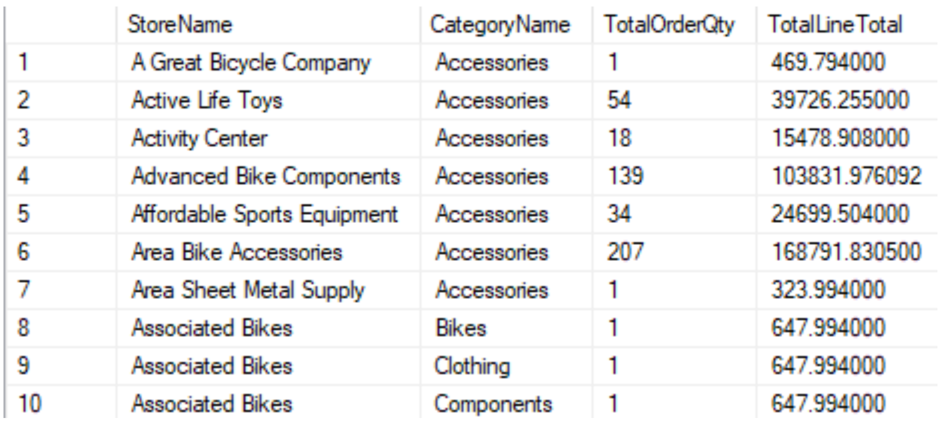
Query-3:



```sql
SELECT STOR.Name as StoreName,
       CAT.Name as CategoryName,
       SUM(SOD.OrderQty) AS TotalOrderQty,
       SUM(SOD.LineTotal) AS TotalLineTotal
  FROM Sales.SalesOrderDetail SOD
 INNER JOIN Sales.SalesOrderHeader SOH
    ON SOH.SalesOrderID = SOD.SalesOrderID
 INNER JOIN Production.Product P
    ON P.ProductID = SOD.ProductID
 INNER JOIN Production.ProductSubcategory SUBCAT
    ON SUBCAT.ProductCategoryID = P.ProductSubcategoryID
 INNER JOIN Production.ProductCategory CAT
    ON CAT.ProductCategoryID = SUBCAT.ProductSubcategoryID
 INNER JOIN Sales.Customer CUST
    ON CUST.CustomerID = SOH.CustomerID
 INNER JOIN Sales.Store STOR
    ON STOR.BusinessEntityID = CUST.StoreID
 WHERE SOH.OrderDate BETWEEN '20130101' AND '20131231'
   AND SOH.OnlineOrderFlag = 0
   AND (P.MakeFlag = 1 OR P.FinishedGoodsFlag = 1)
   AND P.Color IN ('Black', 'Yellow')
 GROUP BY STOR.Name, CAT.Name
 ORDER BY STOR.Name, CAT.Name
```


Clustered Index Scan (Clustered)
[Store].[PK_Store_BusinessEntityID]…
Cost: 3 %
Hash Match
(Inner Join)
Cost: 7 %
Hash Match
(Inner Join)
Cost: 3 %
Clustered Index Scan (Clustered)
[SalesOrderHeader].[PK_SalesOrderHe…
Cost: 20 %
Clustered Index Scan (Clustered)
[Customer].[PK_Customer_CustomerID]…
Cost: 4 %
Hash Match
(Inner Join)
Cost: 21 %
Hash Match
(Inner Join)
Cost: 1 %
Nested Loops
(Inner Join)
Cost: 0 %
Index Scan (NonClustered)
[ProductCategory].[AK_ProductCatego…
Cost: 0 %
Clustered Index Seek (Clustered)
[ProductSubcategory].[PK_ProductSub…
Cost: 0 %
Clustered Index Scan (Clustered)
[Product].[PK_Product_ProductID] [P]
Cost: 0 %
Compute Scalar
Cost: 0 %
Compute Scalar
Cost: 0 %
Clustered Index Scan (Clustered)
[SalesOrderDetail].[PK_SalesOrderDe…
Cost: 38 %

TOTAL RETURNED ROWS IS 656



	StoreName	CategoryName	TotalOrderQty	TotalLineTotal
1	A Great Bicycle Company	Accessories	1	469.794000
2	Active Life Toys	Accessories	54	39726.255000
3	Activity Center	Accessories	18	15478.908000
4	Advanced Bike Components	Accessories	139	103831.976092
5	Affordable Sports Equipment	Accessories	34	24699.504000
6	Area Bike Accessories	Accessories	207	168791.830500
7	Area Sheet Metal Supply	Accessories	1	323.994000
8	Associated Bikes	Bikes	1	647.994000
9	Associated Bikes	Clothing	1	647.994000
10	Associated Bikes	Components	1	647.994000

1- Firstly, We want to create index for SalesOrderHeader table because It's affect on where and inner join parts.

- **SOH.OrderDate :** This column was in Where and it is use for equation so that this part will be first column in index.
- **SOH.OnlineOrderFlag :** This column was in Where and it is use for equation so that this part will be second column in index.
- **SOH.CustomerID :** This column was in right sides of Nested Inner Join so that It's not important as before 2 columns and It will be in include part.
- **SOH.SalesOrderID:** We don't include this column to the index because It's primary key and It's have already have Clustered Index.

```
CREATE INDEX Q3IndexOnlineOrderFlagANDOrderDateWithShipToAddressId
   ON Sales.SalesOrderHeader (OnlineOrderFlag, OrderDate)
   INCLUDE (CustomerID)
```

***OnlineOrderFlag(Equality Column) and OrderDate(Inequality Column) with including ShipToAddressID(Foreign Key for Join) speeds up our query by allowing us to access only the necessary records (join) via index.**

2- We want to create index for SalesOrderDetail table because it's affect on one Nested Inner Join and Select part

- **SOD.SalesOrderID:** SalesOrderID is primary key so that It has index but in this query, It's not enough. We want to create a new Non-Clustered index with some includes to make faster.
- **SOD.OrderQty:** OrderQTY is used in Select part so that It's not important as SalesOrderID, It will be placed in Include part.
- **SOD.LineTotal:** LineTotal is used in Select part so that It's not important as SalesOrderID, It will be placed in Include part.

```
CREATE INDEX Q3ProductIdWithOrderQtyAndLineTotal
   ON Sales.SalesOrderDetail (ProductID)
   INCLUDE (OrderQty, LineTotal)
```

***If we keep the 2 required columns while doing the index seek over the primary key, this prevents us from navigating the table again for the required fields.**

3- We want to create index for Store table because it's affect on Select part.

- **STOR.BussinesEntityID:** is primary key so that It has index but in this query, It's not enough. We want to create a new Non-Clustered index with some includes to make faster.
- **STOR.Name:** Name is used in Select part so that It's not important as SalesOrderID, It will be placed in Include part.

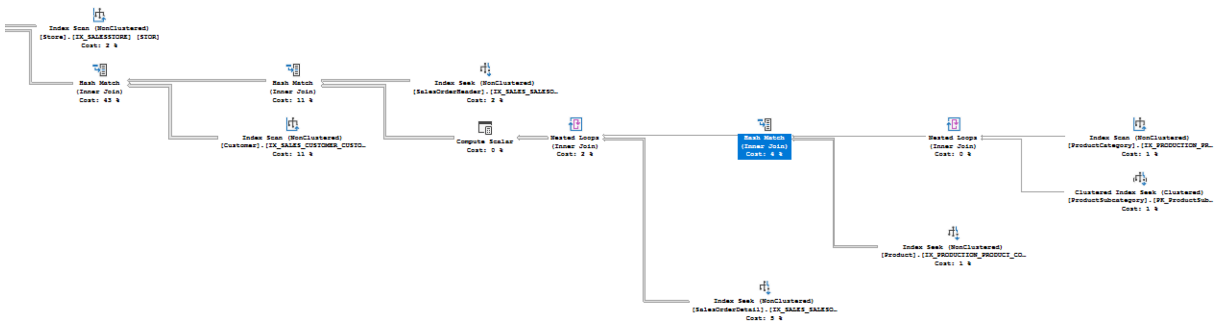
```
CREATE INDEX Q3IndexStoreName
   ON Sales.Store (BusinessEntityID)
   INCLUDE (Name)
```
***This index will be reduce the sorting load of the process and execute the data in the table more faster with sorted data.**

Average times after Indexes

	Onur Akalın	Ahmet Arif Özçelik
Query 3	7.024 s	11.473 s
Remain	+2.226 s	+3.090 s

Total times after Indexes

	Onur Akalın	Ahmet Arif Özçelik
Query 3	11.70 m	19.12 m
Remain	+4.05 m	+5.15 m


Index Scan (NonClustered)
[Store].[IX_SALESSTORE] [STOR]
Cost: 2 %
Hash Match
(Inner Join)
Cost: 43 %
Hash Match
(Inner Join)
Cost: 11 %
Index Seek (NonClustered)
[SalesOrderHeader].[IX_SALES_SALESO…
Cost: 2 %
Index Scan (NonClustered)
[Customer].[IX_SALES_CUSTOMER_CUSTO…
Cost: 11 %
Compute Scalar
Cost: 0 %
Nested Loops
(Inner Join)
Cost: 2 %
Hash Match
(Inner Join)
Cost: 4 %
Nested Loops
(Inner Join)
Cost: 0 %
Index Scan (NonClustered)
[ProductCategory].[IX_PRODUCTION_PR…
Cost: 1 %
Clustered Index Seek (Clustered)
[ProductSubcategory].[PK_ProductSub…
Cost: 1 %
Index Seek (NonClustered)
[Product].[IX_PRODUCTION_PRODUCT_CO…
Cost: 1 %
Index Seek (NonClustered)
[SalesOrderDetail].[IX_SALES_SALESO…
Cost: 5 %

Source Codes

Program.cs

```
using System;

namespace DatabaseProject2
{
    class Program
    {
        static void Main(string[] args)
        {
            // 100 Run report for each query
            string filePath =
"D:\\Proje\\DatabaseProject2\\DatabaseProject2\\Document\\Output.txt";
            Operations query1 = new Operations(Queries.Query1);
            query1.RunQuery100Times();
            query1.WriteToFile(filePath, "1");

            Operations query2 = new Operations(Queries.Query2);
            query2.RunQuery100Times();
            query2.WriteToFile(filePath, "2");

            Operations query3 = new Operations(Queries.Query3);
            query3.RunQuery100Times();
            query3.WriteToFile(filePath, "3");
        }
    }
}
```

Queries.cs

```
namespace DatabaseProject2
{
    public static class Queries
    {
        public static readonly string Query1 =
            "DBCC FREEPROCCACHE; " +
            "DBCC DROPCLEANBUFFERS; " +
            "SELECT SOH.OrderDate, " +
                "PROV.Name AS StateProvinceName, " +
                "ADDR.City, " +
                "SUM(SOD.OrderQty) AS TotalOrderQty, " +
                "SUM(SOD.LineTotal) AS TotalLineTotal " +
            "FROM Sales.SalesOrderDetail SOD " +
                "INNER JOIN Sales.SalesOrderHeader SOH " +
                    "ON SOH.SalesOrderID = SOD.SalesOrderID " +
                "INNER JOIN Person.Address ADDR " +
                    "ON ADDR.AddressID = SOH.ShipToAddressID " +
                "INNER JOIN Person.StateProvince PROV " +
                    "ON PROV.StateProvinceID = ADDR.StateProvinceID " +
            "WHERE SOH.OrderDate BETWEEN '20130101' AND '20131231' " +
                "AND SOH.OnlineOrderFlag = 1 " +
            "GROUP BY SOH.OrderDate, PROV.Name, ADDR.City " +
            "ORDER BY SOH.OrderDate, PROV.Name, ADDR.City";
        public static readonly string Query2 =
            "DBCC FREEPROCCACHE; " +
            "DBCC DROPCLEANBUFFERS; " +
            "SELECT SOH.OrderDate, " +
                    "CAT.Name as CategoryName, " +
                    "SUM(SOD.OrderQty) AS TotalOrderQty, " +
                    "SUM(SOD.LineTotal) AS TotalLineTotal " +
                "FROM Sales.SalesOrderDetail SOD " +
                    "INNER JOIN Sales.SalesOrderHeader SOH " +
                        "ON SOH.SalesOrderID = SOD.SalesOrderID " +
                    "INNER JOIN Production.Product P " +
                        "ON P.ProductID = SOD.ProductID " +
                    "INNER JOIN Production.ProductSubcategory SUBCAT " +
                        "ON SUBCAT.ProductCategoryID = P.ProductSubcategoryID " +
                    "INNER JOIN Production.ProductCategory CAT " +
                        "ON CAT.ProductCategoryID = SUBCAT.ProductSubcategoryID " +
                "WHERE SOH.OrderDate BETWEEN '20130101' AND '20131231' " +
                    "AND SOH.OnlineOrderFlag = 1 " +
                    "AND (P.MakeFlag = 1 OR P.FinishedGoodsFlag = 1) " +
                    "AND P.Color IN ('Black', 'Yellow') " +
                "GROUP BY SOH.OrderDate, CAT.Name " +
                "ORDER BY SOH.OrderDate, CAT.Name";

        public static readonly string Query3 =
            "DBCC FREEPROCCACHE; " +
            "DBCC DROPCLEANBUFFERS; " +
            "SELECT STOR.Name as StoreName, " +
                    "CAT.Name as CategoryName, " +
                    "SUM(SOD.OrderQty) AS TotalOrderQty, " +
                    "SUM(SOD.LineTotal) AS TotalLineTotal " +
                "FROM Sales.SalesOrderDetail SOD " +
                    "INNER JOIN Sales.SalesOrderHeader SOH " +
                        "ON SOH.SalesOrderID = SOD.SalesOrderID " +
                    "INNER JOIN Production.Product P " +
                        "ON P.ProductID = SOD.ProductID " +
                    "INNER JOIN Production.ProductSubcategory SUBCAT " +
                        "ON SUBCAT.ProductCategoryID = P.ProductSubcategoryID " +
                    "INNER JOIN Production.ProductCategory CAT " +
                        "ON CAT.ProductCategoryID = SUBCAT.ProductSubcategoryID " +
                    "INNER JOIN Sales.Customer CUST " +
                        "ON CUST.CustomerID = SOH.CustomerID " +
                    "INNER JOIN Sales.Store STOR " +
                        "ON STOR.BusinessEntityID = CUST.StoreID " +
                "WHERE SOH.OrderDate BETWEEN '20130101' AND '20131231' " +
                    "AND SOH.OnlineOrderFlag = 0 " +
                    "AND (P.MakeFlag = 1 OR P.FinishedGoodsFlag = 1) " +
                    "AND P.Color IN ('Black', 'Yellow') " +
                "GROUP BY STOR.Name, CAT.Name " +
                "ORDER BY STOR.Name, CAT.Name";
    }
}
```

Operations.cs

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.IO;

namespace DatabaseProject2
{
    public class Operations
    {
        private readonly string _connectionString =
            @"Data Source=(LocalDB)\MSSQLLocalDB;Initial Catalog=AdventureWorks2012;Integrated
Security=True;Connection Timeout = 6000";

        private readonly SqlCommand _command;
        private TimeSpan averageTime;

        public Operations(String query)
        {
            _command = new SqlCommand(query);
        }
        private TimeSpan RunQuery()
        {
            DateTime beginTime = DateTime.Now;
            SqlConnection connection = new SqlConnection(_connectionString);
            try
            {
                connection.Open();
                _command.Connection = connection;

                for (int i = 0; i < 100; i++)
                    _command.ExecuteNonQuery();
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
                throw;
            }
            finally
            {
                if (connection.State == ConnectionState.Open)
                    connection.Close();
            }

            DateTime endTime = DateTime.Now;
            TimeSpan elapsed = endTime - beginTime;
            Console.WriteLine(beginTime + " " + endTime);
            Console.Write("Birim ölçüm sonucu(toplam süre) : " + elapsed );
            return elapsed;
        }

        public void RunQuery100Times()
        {
            TimeSpan totalTime = TimeSpan.Zero;
            for (int i = 0; i < 100; i++)
            {
                totalTime += RunQuery();
                Console.WriteLine(" (100/" + (i + 1) + ")" + "\n");
            }

            averageTime = totalTime / 100;
            Console.WriteLine("100 ölçüm sonucu(ortalama süre) : " + averageTime + "\n");
        }

        public void WriteToFile(string fileName, string numberOfQuery)
        {
            string[] lines =
            {
                "**************************************************************",
                " ",
                "After Indexes",
                "Query"+numberOfQuery+" : Average Time is " + averageTime ,
                " ",
                "**************************************************************\n"
            };

            File.AppendAllLines(fileName, lines);
            averageTime = TimeSpan.Zero;
        }
    }
}
```

Environments

Onur Akalın

MacOS – SQL2019 as a Docker Container – Rider IDE Used

Processor: Intel® CoreTM i5-8257U 1.40 Ghz, up to 3.90 GHz Ram: 8 Gb

Ahmet Arif Özçelik

Windows – SQL 2019 – Visual Studio IDE Used

Processor: Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz 2.40 GHz Ram: 16 Gb