Importance of Ordered Indices

Owner: Venkata Praveen Abburi

Problem Statement:

The execution of complex queries in database system with large data will be overhead in terms of performance issues. The main performance issues will be the execution speed and time taken to execute a complex query.

Tentative Solution:

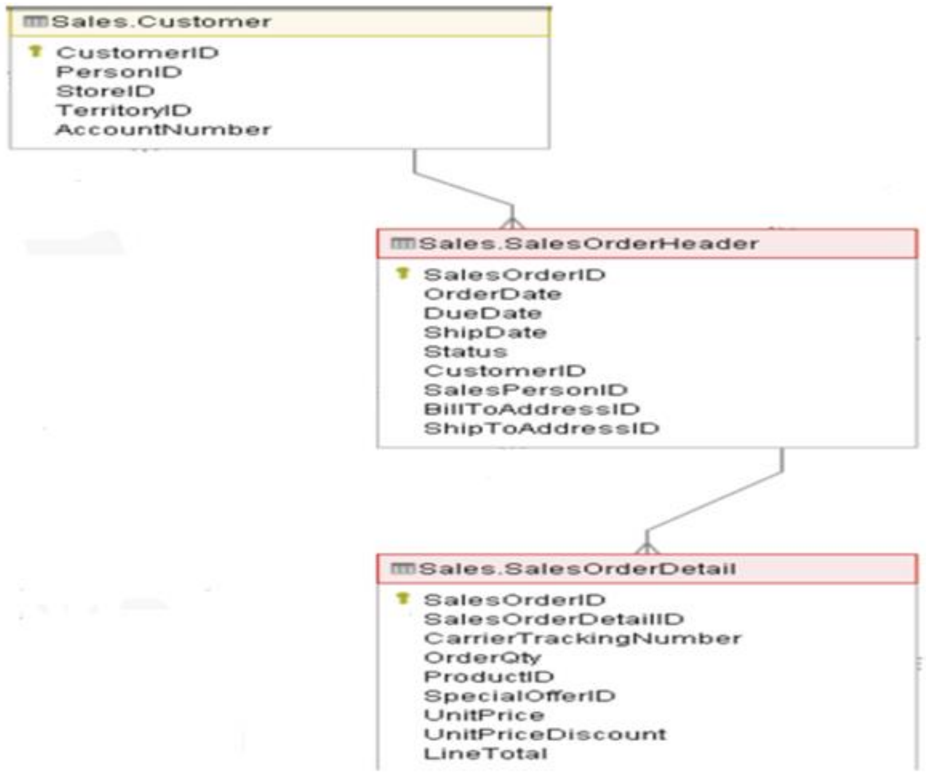
This type of performance issues in the real time databases can be overcome by using indices concept. Ordered Indices help us to improve the performance of system in terms of speed and time. In our scenario we used joins and filters to implement the complex query, usage of clustered and non-clustered indices helps to improve the performance cost.

Data Set:

Here we are using Sales data set from Adventure works(Sample data provided by Microsoft) database 2017 in SQL server.

Below are the three database tables in sales schema used in our project:

1. Customer
2. SalesOrderHeader
3. SalesOrderDetail



Experiment on problem statement:

In our project, we will illustrate how we can improve performance with indices(clustered, nonclustered)by running complex queries in SQL server management studio 2017 on Adventure works database.

1. Clustered index vs no index table:

Here we are creating Clustered index on different fields from the above tables and running two different queries to do our analysis.

Case 1:

Clustered index on SalesOrderId from SalesOrderDetail table.

```
CREATE CLUSTERED INDEX IX_CLU_SOID ON Sales.SalesOrderDetail (SalesOrderID);
```

Query:

Set STATISTICS IO ON;

```
SELECT SalesOrderID,SUM(UnitPrice) AS Total FROM Sales.SalesOrderDetail WHERE
SalesOrderID>63559 and unitprice>100 GROUP BY SalesOrderID;
```

The above query displays sum of Unit Price for Particular SalesOrderID on condition SalesOrderID greaterthan 63559 and Unit price value morethan 100.

After executing the above query in execution plan ,we can figure out how the performance is better when compared to query with no clustered index. By setting **STATISTICS IO ON** logical reads data can be figured out .

Please find the below table for performance figures:

	Clustered Index	Without Clustered Index
Operation	Index Seek	Table Scan
Estimated I/O Cost	0.29794	0.920162
Estimated CPU Cost	0.040704	0.133606
Estimated Operator Cost	0.338644	1.05377
No. of Rows Read	36850	121317
Logical Reads	373	1239

Our observations from execution plan is as below:

1. The **number of rows read** are **36850** in **clustered** index, while in **no index** table **number of rows read** are **121317**(Total tuples in the table).In clustered index the Index Seek is happening at index level, where as in no index table the scan is done on all tuples from the table.

2. Clustered index is consuming less cost in terms of CPU ,Operator and IO, compared to the no index table.

The following are the statistics IO of the above query.

If logical reads are more it tends to high memory Usage. It is clear from the above screenshot that the logical reads in clustered index are less compared to no index table.

we can conclude that the logical reads at clustered index level is less therefore consuming less memory utilization, while in noindex table the logical reads are high and hence consuming more memory utilization .

Case2:

In this case we are creating clustered index on Unitprice field from SalesOrderDetail table.

- `CREATE CLUSTERED INDEX IX_CLU_UP ON Sales.SalesOrderDetail (UnitPrice);`

Query:

```
SELECT SalesOrderID,SUM(UnitPrice) AS Total FROM Sales.SalesOrderDetail WHERE
SalesOrderID>63559 and unitprice>100 GROUP BY SalesOrderID;
```

After executing the above query in execution plan ,we can figure out how the performance is better when compared to query with no clustered index. By setting **STATISTICS IO ON** logical reads data can be figured out .

Please find the below table for performance figures:

	Clustered Index	Without Clustered Index
Operation	Index Seek	Table Scan
Estimated I/O Cost	0.4349	0.920162
Estimated CPU Cost	0.0582	0.133606
Estimated Operator Cost	0.4932	1.05377
No. of Rows Read	52985	121317
Logical Reads	625	1239

Our observations from the execution plan is as below:

1. The **number of rows read** are **52985** in **clustered** index, while in **no index** table **number of rows read** are **121317**(Total tuples in the table).
2. In clustered index the Index Seek is happening at index level, where as in no index table the scan will done on all tuples from the table.
3. Clustered index is consuming less cost in terms of CPU ,Operator and IO compared to the no index table.
4. The logical reads are also less in Clustered index, which means it is consuming less memory utilization compared with no index table.

Query 2:

Here we are doing analysis of three cases, in each case we are using the same query with clustered index on different fields Table.

Case 1:

Here we are creating Clustered index on CustomerId from customer table.

```
CREATE CLUSTERED INDEX IX_CLU_CID ON Sales.customer (CustomerID);
```

Query:

```
SELECT customer.CustomerID,YEAR(OrderDate) Year,
COUNT(YEAR(OrderDate)) Ordered FROM sales.Customer INNER JOIN sales.SalesOrderHeader
ON Customer.CustomerID=SalesOrderHeader.CustomerID WHERE customer.CustomerID between
29710 and  29720 and YEAR(OrderDate)=2012
GROUP BY YEAR(OrderDate),Customer.CustomerID

ORDER BY 3
```

The above query results the total orders placed by the customer in the year of 2012 on condition where CustomerID between 29710 and 29720

After executing the above query in execution plan ,we can figure out how the performance is better in nonclustered when compared with no clustered index. By setting **STATISTICS IO ON** logical reads data can be figured out .

Please find the below table for performance figures:

	Clustered index	No clustered index
Operation	Index Seek + Table Scan	Table Scan
Estimated I/O Cost	0.512917	0.60180
Estimated CPU Cost	0.0349371	0.0567
Estimated Operator Cost	0.54785	1.6585
No. of Rows Read	31476	51285
Logical Reads	687	806

Our observations from the execution plan is as below:

1. The number of rows read in clustered index are less than no index table.
2. Index seek is happening on Customer table because cluster index is applied and Table scan is happening on SalesOrderHeader table which doesn't have any index.
3. For No clustered index table scan is done in all tuples of the tables.
4. Even though Index seek and table scan is happening for the above query the result is much better performance in terms of CPU ,Operator and IO Cost when compared with no index tables because one Index seek is happening on Customer table.
5. The logical reads are also less in Clustered index, which means it is consuming less memory utilization than no index table.

Case 2:

In this case we are creating Clustered index on CustomerID field from SalesOrderHeader table.

- `CREATE CLUSTERED INDEX` IX_CLU_CID `ON` Sales.salesOrderHeader (CustomerID);

Query:

```
SELECT customer.CustomerID,YEAR(OrderDate) Year,
COUNT(YEAR(OrderDate)) Ordered FROM sales.Customer INNER JOIN sales.SalesOrderHeader
ON Customer.CustomerID=SalesOrderHeader.CustomerID WHERE customer.CustomerID between
29710 and  29720 and YEAR(OrderDate)=2012
GROUP BY YEAR(OrderDate),Customer.CustomerID

ORDER BY 3
```

After executing the above query in execution plan ,we can figure out how the performance is better when compared to query with no clustered index. By setting STATISTICS IO ON logical reads data can be figured out .

Please find the below table for performance figures:

	Clustered index	No clustered index
Operation	Index Seek + Table Scan	Table Scan
Estimated I/O Cost	0.0958	0.60180
Estimated CPU Cost	0.0222	0.0567
Estimated Operator Cost	0.1180	1.6585
No. of Rows Read	19904	51285
Logical Reads	127	806

By observing the execution plan :

1. The number of rows read in clustered index are less than no index table.
2. Index seek is happening on SalesOrderHeader table because cluster index is applied and Table scan is happening on Customer table which doesn't have any index.
3. For No clustered index table scan is done in all tuples of the tables.
4. Even though Index seek and table scan is happening for the above query the result is much better performance in terms of CPU ,Operator and IO Cost when compared with no index tables because one Index seek is happening on Customer table.
5. The logical reads are also less in Clustered index, which means it is consuming less memory utilization than no index table.

Case 3:

In this case we are creating two indices on two tables.

Clustered index on CustomerID field from customer table and salesOrderHeader table.

- `CREATE CLUSTERED INDEX` IX_CLU_CID `ON` Sales.customer (CustomerID);
- `CREATE CLUSTERED INDEX` IX_CLU_CID `ON` Sales.salesOrderHeader (CustomerID);

Query:

```
SELECT customer.CustomerID,YEAR(OrderDate) Year,
COUNT(YEAR(OrderDate)) Ordered FROM sales.Customer INNER JOIN sales.SalesOrderHeader
ON Customer.CustomerID=SalesOrderHeader.CustomerID WHERE customer.CustomerID between
29710 and  29720 and YEAR(OrderDate)=2012
GROUP BY YEAR(OrderDate),Customer.CustomerID

ORDER BY 3
```

After executing the above query in execution plan ,we can figure out how the performance is better when compared to query with no clustered index. By setting STATISTICS IO on logical reads data can be figured out .

Please find the below table for performance figures:

	Clustered index	No clustered index
Operation	Index Seek	Table Scan
Estimated I/O Cost	0.00625	0.60180
Estimated CPU Cost	0.000334	0.0567
Estimated Operator Cost	0.01109	1.6585
No. of Rows Read	95	51285
Logical Reads	37	806

Our observations from the actual execution plan is as below:

1. The number of rows read in clustered index are less than the number of rows read in no index table.
2. In clustered index the Index seek is happening at index level, where as in no index table the scan will done on all tuples from the table.
3. Clustered index is consuming less cost in terms of CPU ,Operator and IO compared to the no index table.
4. The logical reads are also less in Clustered index, which means it is consuming less memory utilization than no index table.

Analysis

From the above two cases , the table with Clustered indices is performing better when compared to the no index table.

2. Nonclustered index vs no index table:

Here we are creating nonclustered index on different fields from the above tables and running a query to do our analysis.

Case1:

In this case we are creating three indices on three tables.

We created nonclustered index on CustomerID field of customer table, salesOrderId field of salesOrderHeader table and salesOrderId field of SalesOrderDetail table.

- `CREATE NONCLUSTERED INDEX IX_NC_S_C_CID ON Sales.customer (CustomerId);`
- `CREATE NONCLUSTERED INDEX IX_NC_S_SOH_SOI ON Sales.salesOrderHeader (salesOrderId);`
- `CREATE NONCLUSTERED INDEX IX_NC_S_SOD_SOI ON Sales.SalesOrderDetail (salesOrderId);`

Query:

```
SELECT sc.CustomerID,so.ProductID ,COUNT(ProductID) AS Purchased FROM
Sales.Customer sc INNER JOIN sales.SalesOrderHeader soh
ON sc.CustomerID=soh.CustomerID INNER JOIN sales.SalesOrderDetail so ON
soh.SalesOrderID=so.SalesOrderID
WHERE sc.CustomerID between 29707 and 29725 and ProductID =726
GROUP BY sc.CustomerID,ProductID
```

The above query displays Total products purchased by customer's with ProductID 726 and their CustomerID between 29707 and 29725.

In the above query, one index scan (on Customer) and two table scans (on SalesOrderHeader and SalesOrderDetail).

After executing the above query in execution plan ,we can figure out how the performance is better when compared to query with no clustered index. By setting STATISTICS IO on logical reads data can be figured out .

Please find the below table for performance figures:

	Non Clustered Index	Without Non Clustered Index
Operation	1 Index Seek+ 2 Table Scan	Table Scan
Estimated I/O Cost	1.433079	1.5219676
Estimated CPU Cost	0.1685326	0.1979645
Estimated Operator Cost	1.601643	1.712303
No. of Rows Read	152805	212242
Logical Reads	1970	4707

From the above table it is clear that :

1. The number of rows read in nonclustered index are less than no index table.
2. Index seek is happening on Customer table because noncluster index is applied and Table scan is happening on SalesOrderHeader and salesOrderDetail tables which doesn't have any index on required fields.
3. For No clustered index table scan is done in all tuples of the tables.
4. Even though Index seek and table scan is happening for the above query the result is much better performance in terms of CPU ,Operator and IO Cost when compared with no index tables because one Index seek is happening on Customer table.
5. The logical reads are also less in nonclustered index, which means it is consuming less memory utilization than no index table.

Case 2:

In this case we are creating three indices on three tables.

We created non clustered index on StoreID field of customer, CustomerId field of salesOrderHeader and ProductID of SalesOrderDetailtable.

- `CREATE NONCLUSTERED INDEX IX_NC_S_C_SID ON Sales.customer (StoreID);`
- `CREATE NONCLUSTERED INDEX IX_NC_S_SOH_CID ON Sales.salesOrderHeader (CustomerId);`
- `CREATE NONCLUSTERED INDEX IX_NC_S_SOD_PID ON Sales.SalesOrderDetail (ProductID);`

Query:

```
SELECT sc.CustomerID,so.ProductID ,COUNT(ProductID) AS Purchased FROM
Sales.Customer sc INNER JOIN sales.SalesOrderHeader soh
ON sc.CustomerID=soh.CustomerID INNER JOIN sales.SalesOrderDetail so ON
```

```
soh.SalesOrderID=so.SalesOrderID
WHERE sc.CustomerID between 29707 and 29725 and ProductID =726
GROUP BY sc.CustomerID,ProductID
```

In the above query, Two index scans (on SalesOrderDetail and SalesOrderHeader table) and one table scan (on customer table).

After executing the above query in execution plan ,we can figure out how the performance is better when compared to query with no clustered index. By setting STATISTICS IO on logical reads data can be figured out .

Please find the below table for performance figures:

	Non Clustered Index	Without Non Clustered Index
Operation	2 Index Seek +1 Table Scan	Table Scan
Estimated I/O Cost	0.0982639	1.5219676
Estimated CPU Cost	0.0225979	0.1979645
Estimated Operator Cost	0.123833	1.712303
No. Of Rows Read	20256	212242
Logical Reads	598	4707

From the above table it is clear that :

1. The number of rows read in nonclustered index are less than no index table.
2. Index seek is happening on SalesOrderDetail and SalesOrderHeader tables because noncluster index is applied and Table scan is happening on customer table which doesn't have any index on required fields.
3. For No clustered index table scan is done in all tuples of the tables.
4. Even though Index seek and table scan is happening for the above query the result is much better performance in terms of CPU ,Operator and IO Cost when compared with no index tables because one Index seek is happening on Customer table.
5. The logical reads are also less in nonclustered index, which means it is consuming less memory utilization than no index table.

Case3:

In this case we are creating three indices on three tables.

We created non clustered index on CustomerId field of customer table, CustomerId field of salesOrderHeader and ProductID of SalesOrderDetail table.

- `CREATE NONCLUSTERED INDEX IX_NC_S_C_CID ON  Sales.customer (CustomerId);`
- `CREATE NONCLUSTERED INDEX IX_NC_S_SOH_CID ON  Sales.salesOrderHeader (CustomerId);`
- `CREATE NONCLUSTERED INDEX IX_NC_S_SOD_PID ON  Sales.SalesOrderDetail (ProductID);`

Query:

```sql
SELECT sc.CustomerID,so.ProductID ,COUNT(ProductID) AS Purchased FROM
Sales.Customer sc INNER JOIN sales.SalesOrderHeader soh
ON sc.CustomerID=soh.CustomerID INNER JOIN sales.SalesOrderDetail so ON
soh.SalesOrderID=so.SalesOrderID
WHERE sc.CustomerID between 29707 and 29725 and ProductID =726
GROUP BY sc.CustomerID,ProductID
```

This query displays the no. of products purchased by customers with Id in between 29707 and 29725 and ProductID =726

In the above query, index seek is happening at 3 tables which results the good performance of query.

After executing the above query in execution plan ,we can figure out how the performance is better when compared to query with no clustered index. By setting STATISTICS IO on logical reads data can be figured out .

Please find the below table for performance figures:

	NonClustered Index	Without Non Clustered Index
Operation	index Seek	Table Scan
Estimated I/O Cost	0.009375	1.5219676
Estimated CPU Cost	0.0009429	0.1979645
Estimated Operator Cost	0.0103773	1.712303
No. Of Rows Read	459	212242
Logical Reads	487	4707

From the above query it is clear that:

1. The number of rows read in nonclustered index are less than no index table.
2. Index seek is happening on all 3 tables where as in No clustered index table scan is done in all tuples of the tables.
3. Full index seeks results much better performance in terms of CPU ,Operator and IO Cost when compared with no index tables.
4. The logical reads are also less in nonclustered index, which means it is consuming less memory utilization than no index table.

Analysis:

From the above two cases , the table with nonclustered indices is performing better when compared to the no index table.

Result of Project:.

Hence it is proved from the figures that the performance cost is improved in terms of speed and execution time of query with the usage of indices.

References:

[1] **Bert Wagner**, “Clustered vs Nonclustered: “What Index Is Right For My Data?”, hackernoon.com , Dec. 18, 2017. [Online].Available: https://hackernoon.com/clustered-vs-nonclustered-what-index-is-right-for-my-data-717b329d042c[Accessed: Feb. 24, 2019].

[2] Microsoft Developer Network , “Adventure Works Installation and configuration” Available: “ https://docs.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-2017[Accessed: Feb. 24, 2019].

[3] Abraham Silberschatz, Henry F. Korth, and S. Sudarshan, *Database System Concepts Sixth Edition*.