Importance of Indices

Owner : Venkata Praveen Abburi

vxa91760@ucmo.edu

Problem Statement:

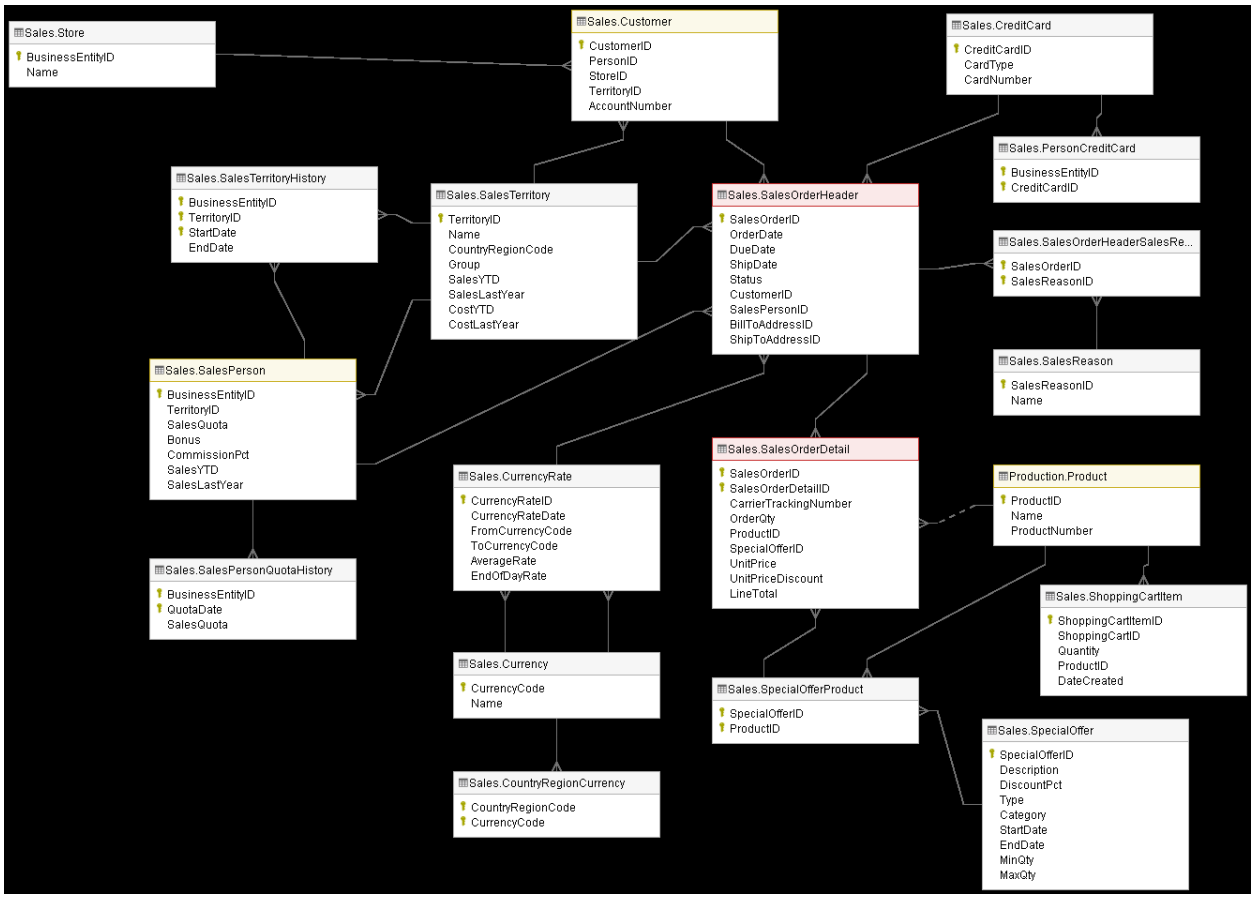
The execution of complex queries in database system with large data will be overhead in terms of performance issues. The main performance issues will be the execution speed and time taken to execute a complex query.

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 will use 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. The Data set is as following structure :



Here we are running sample queries in SQL server management studio 2017 on Adventure works with clustered, non clustered indices &without indices.

Suppose if we run a sample select query in clustered index table the result is shown below:

```
select * from sales.SalesOrderDetail where SalesOrderDetailID=121317;
```

Here I am selecting all fields from SalesOrderDetail table with the condition on clustered index table(SalesOrderDetailID);

While executing the above query enable the actual execution plan and STATISTICS IO and analyse the performance parameters in the query.

Running with Index & no index:

Actual execution plan:

Clustered index table

Key Lookup (Clustered)

Uses a supplied clustering key to lookup on a table that has a clustered index.

Physical Operation	Key Lookup
Logical Operation	Key Lookup
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Number of Rows Read	1
Actual Number of Rows	1
Actual Number of Batches	0
Estimated Operator Cost	0.0032831 (1%)
Estimated I/O Cost	0.003125
Estimated CPU Cost	0.0001581
Estimated Subtree Cost	0.0032831
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows	1
Estimated Row Size	82 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	5

Object
[AdventureWorks_index].[Sales].[SalesOrderDetail].
[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID]

Output List
[AdventureWorks_index].[Sales].
[SalesOrderDetail].CarrierTrackingNumber,
[AdventureWorks_index].[Sales].[SalesOrderDetail].OrderQty,
[AdventureWorks_index].[Sales].[SalesOrderDetail].SpecialOfferID,
[AdventureWorks_index].[Sales].[SalesOrderDetail].UnitPrice,

No index table

Table Scan

Scan rows from a table.

Physical Operation	Table Scan
Logical Operation	Table Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Number of Rows Read	121317
Actual Number of Rows	1
Actual Number of Batches	0
Estimated I/O Cost	0.920241
Estimated Operator Cost	1.05377 (97%)
Estimated CPU Cost	0.133527
Estimated Subtree Cost	1.05377
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows	1.05742
Estimated Number of Rows to be Read	121317
Estimated Row Size	95 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	False
Node ID	2

Predicate
[AdventureWorks_noindex].[Sales].[SalesOrderDetail].
[SalesOrderDetailID]=[@1]

Object
[AdventureWorks_noindex].[Sales].[SalesOrderDetail]

Output List
[AdventureWorks_noindex].[Sales].
[SalesOrderDetail].SalesOrderID,
[AdventureWorks_noindex].[Sales].

After execution of above query and open the actual execution plan here we can observe that that

1)total no.of rows can be read is 1 in clustered index because here key look up is happning at index level and only one row is reading at table level where as in no index table the table scan is done and the all tuples reading from the table.

2)Clustered index is taking less cost in terms of CPU ,Operator and IO compared with the no index table.

Now open the Messages here we can see that logical reads because we enabled the statistics IO. The following illustrate the statistics IO of the above query.

For Clustered Index:

```
(1 row affected)
Table 'SalesOrderDetail'. Scan count 1, logical reads 279, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
```

For no index table :

```
(1 row affected)
Table 'SalesOrderDetail'. Scan count 1, logical reads 1239, physical reads 0, read-ahead reads 861, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0
```

A logical read occurs every time the database engine requests a page from the buffer cache. If the page is not currently in the buffer cache, a physical read is then performed to read the page into the buffer cache. If the page is currently in the cache, no physical read is generated;

If logical reads are more tends to high memory Usage.

From the above we can conclude that the logical reads at clustered index level is less so while running query its taking less memory. Where as in no index table taking high memory utilisationg because the no.of logical reads are more.

2)Non clustered index:

Here if we run a bit complex query on non clustered index column let us analyse the performance. The Query as follows

```
SET STATISTICS IO ON
GO
 SELECT Customer.CustomerID,SalesOrderDetail.ProductID,COUNT(*) as cnt FROM
Sales.Customer  INNER JOIN sales.SalesOrderHeader
 ON Customer.CustomerID=SalesOrderHeader.CustomerID INNER JOIN sales.SalesOrderDetail ON
 SalesOrderHeader.SalesOrderID=SalesOrderDetail.SalesOrderID  GROUP BY
customer.CustomerID,ProductID having COUNT(*)>2;
```

The above query displays each customer purchased product count is more than 2 items.

Tables inclded here : 1) Customer

2) SalesOrderHeader

3) SalesOrderDetail

While executing the above query enable the actual execution plan and STATISTICS IO and analyse the performance parameters in the query.

Running with Index & no index:

Actual execution plan:

Non clustered index With no index

Index Scan (NonClustered)

Scan a nonclustered index, entirely or only a range.

Physical Operation	Index Scan
Logical Operation	Index Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Number of Rows Read	121317
Actual Number of Rows	121317
Actual Number of Batches	0
Estimated I/O Cost	0.205347
Estimated Operator Cost	0.338953 (7%)
Estimated CPU Cost	0.133606
Estimated Subtree Cost	0.338953
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows	121317
Estimated Number of Rows to be Read	121317
Estimated Row Size	15 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	False
Node ID	7

Object
[AdventureWorks_index].[Sales].[SalesOrderDetail].[IX_SalesOrderDetail_ProductID]

Output List
[AdventureWorks_index].[Sales].[SalesOrderDetail].SalesOrderID, [AdventureWorks_index].[Sales].[SalesOrderDetail].ProductID

Table Scan

Scan rows from a table.

Physical Operation	Table Scan
Logical Operation	Table Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Number of Rows Read	121317
Actual Number of Rows	121317
Actual Number of Batches	0
Estimated Operator Cost	1.05377 (16%)
Estimated I/O Cost	0.920162
Estimated CPU Cost	0.133606
Estimated Subtree Cost	1.05377
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows to be Read	121317
Estimated Number of Rows	121317
Estimated Row Size	15 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	False
Node ID	7

Object
[AdventureWorks_noindex].[Sales].[SalesOrderDetail]

Output List
[AdventureWorks_noindex].[Sales].[SalesOrderDetail].SalesOrderID, [AdventureWorks_noindex].[Sales].[SalesOrderDetail].ProductID

After execution of above query and open the actual execution plan here we can observe that that

1) The total no.of reads in the table is same in both because here the index scan & table scan is done while executing the query.
2) But if we observe the I/O ,operator & CPU cost in non clustered index is less compared to the no index table. From this we achieved the performance of database.

Now open the Messages here we can see that logical reads because we enabled the statistics IO. The following illustrate the statistics IO of the above query.

For Non-Clustered Index:

```
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'SalesOrderDetail'. Scan count 1, logical reads 276, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'SalesOrderHeader'. Scan count 1, logical reads 57, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Customer'. Scan count 1, logical reads 123, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
```

For No index table:

```
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'SalesOrderDetail'. Scan count 1, logical reads 1239, physical reads 0, read-ahead reads 1239, lob logical reads 0, lob physical reads 0, lob read-ahead reads
Table 'SalesOrderHeader'. Scan count 1, logical reads 685, physical reads 0, read-ahead reads 685, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Customer'. Scan count 1, logical reads 121, physical reads 0, read-ahead reads 121, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
```

From the above statistics we can conclude that logical read count is less in non clustered index where as no index table has more logical reads.Due to the high logical reads in no index table it can utilize more cpu compred to non clustered index.