[batch mode sql server rowstore in short - Search](https://www.bing.com/search?q=batch%20mode%20sql%20server%20rowstore%20in%20short&qs=n&form=QBRE&sp=-1&ghc=1&lq=0&pq=batch%20mode%20sql%20server%20rowstore%20in%20short&sc=9-39&sk=&cvid=56249C1BBA304BD788E8EB8A335D4CDE&ghsh=0&ghacc=0&ghpl=)

Key Points:

Batch Processing: Processes multiple rows at a time (typically 900 rows per batch), reducing CPU usage and improving efficiency.

Rowstore: Works with traditional row-based storage, unlike Batch Mode on Columnstore which is designed for column-based storage.

Compatibility: Available in SQL Server 2019 and later versions.

Use Cases: Particularly beneficial for complex queries involving large datasets, such as aggregations, joins, and scans.

Benefits:

Improved Performance: Faster query execution times due to reduced CPU cycles.

Resource Efficiency: Better utilization of system resources, leading to overall improved system performance.

Example:

SELECT

ProductID,

SUM(SalesAmount) AS TotalSales

FROM

Sales

GROUP BY

ProductID

OPTION (USE HINT ('ENABLE\_BATCH\_MODE'));

This query leverages Batch Mode Rowstore to optimize the aggregation operation.

[SQL SERVER - Simple Example of Batch Mode in RowStore - SQL Authority with Pinal Dave](https://blog.sqlauthority.com/2019/11/18/sql-server-simple-example-of-batch-mode-in-rowstore/)

**Batch Mode in RowStore**

SQL Server 2019 now supports Batch Mode in RowStore, which was previously only possible via few hacks or using a column store index. However, in SQL Server 2019, now Batch Mode is available for a query without even having a column store index.

The biggest difference between the newer batch mode and traditional row mode operation is the performance of the query with aggregation. The traditional Row Execution Mode processes are performed on a row-by-row basis, whereas the batch execution mode processes data by grouping them into batches. The typical batch is of several hundreds of rows. Usually, the number rows per batches are between 64 to 912 rows but again, this number can easily change if Microsoft releases a new patch or updates to the algorithm.

Let us see a simple example of we can get the batch mode in rowstore.

**Simple Example**

First I will be building a table with lots of data. Please note that this demo needs really lots of rows in your table otherwise, it will find the traditional use of row mode efficient.

In the following example, I am using an AdventureWorks database, but you can use any database with lots of rows.

USE AdventureWorks2017

GO

-- Create New Table

CREATE TABLE [dbo].[MySalesOrderDetail](

[SalesOrderID] [int] NOT NULL,

[SalesOrderDetailID] [int] NOT NULL,

[CarrierTrackingNumber] [nvarchar](25) NULL,

[OrderQty] [smallint] NOT NULL,

[ProductID] [int] NOT NULL,

[SpecialOfferID] [int] NOT NULL,

[UnitPrice] [money] NOT NULL,

[UnitPriceDiscount] [money] NOT NULL,

[LineTotal] [numeric](38, 6) NOT NULL,

[rowguid] [uniqueidentifier] NOT NULL,

[ModifiedDate] [datetime] NOT NULL

) ON [PRIMARY]

GO

-- Create clustered index

CREATE CLUSTERED INDEX [CL\_MySalesOrderDetail] ON [dbo].[MySalesOrderDetail]

( [SalesOrderDetailID])

GO

-- Create Sample Data Table

-- WARNING: This Query may run upto 2-10 minutes based on your systems resources

INSERT INTO [dbo].[MySalesOrderDetail]

SELECT [SalesOrderID],[SalesOrderDetailID],[CarrierTrackingNumber],

[OrderQty],[ProductID],[SpecialOfferID],[UnitPrice],

[UnitPriceDiscount],[LineTotal],[rowguid],[ModifiedDate]

FROM Sales.SalesOrderDetail S1

GO 50

Now let us run the following script where we keep the compatibility level of the database to 140 which represents SQL Server 2017.

-- 2017

ALTER DATABASE [AdventureWorks2017] SET COMPATIBILITY\_LEVEL = 140

GO

-- Comparing Regular Index with ColumnStore Index

USE AdventureWorks2017

GO

SET STATISTICS IO, TIME ON

GO

-- Select Table with regular Index

SELECT ProductID, SUM(UnitPrice) SumUnitPrice, AVG(UnitPrice) AvgUnitPrice,

SUM(OrderQty) SumOrderQty, AVG(OrderQty) AvgOrderQty

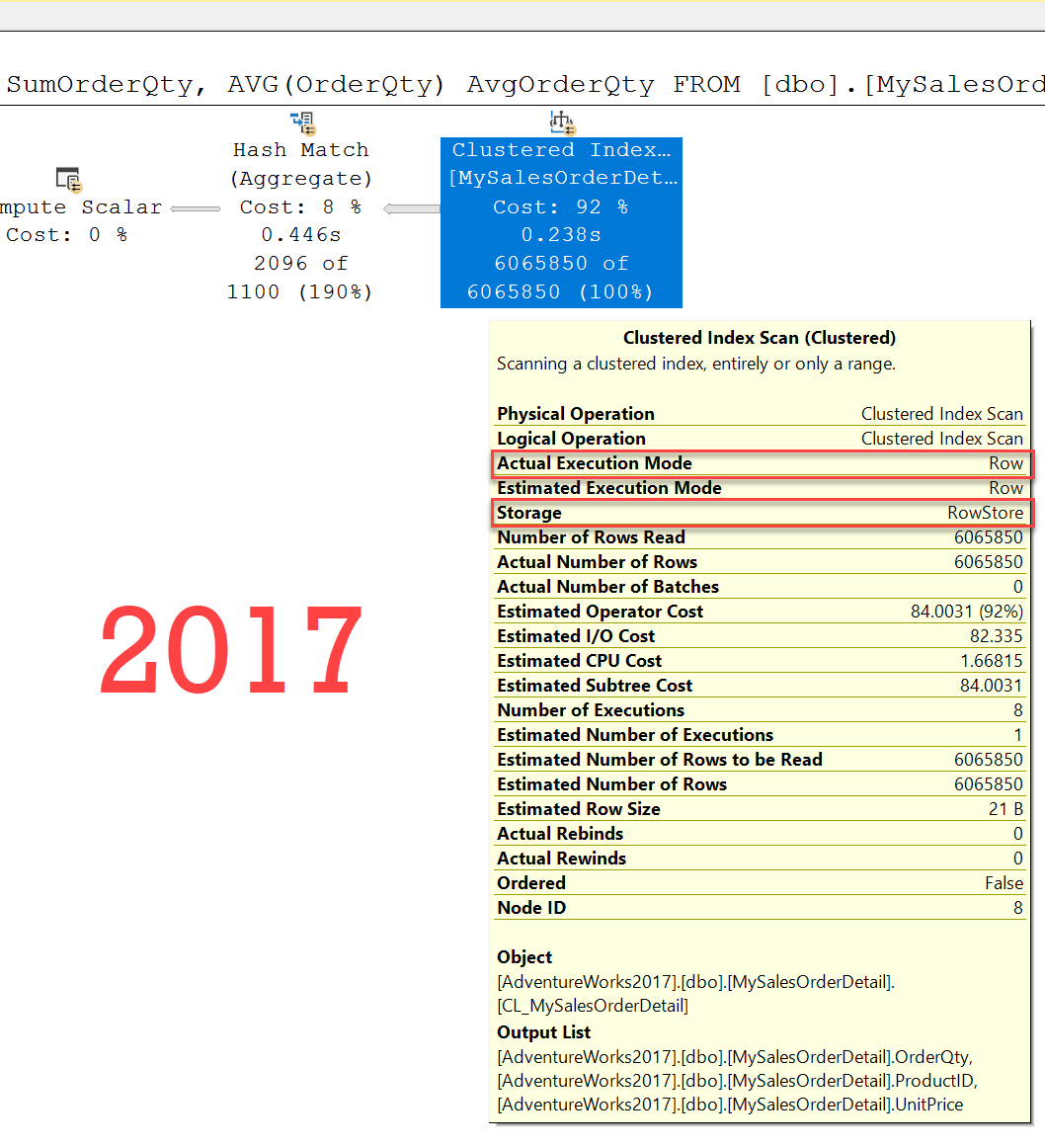
FROM [dbo].[MySalesOrderDetail]

GROUP BY ProductID

ORDER BY ProductID

GO

Check the execution plan of the query.



Now let us run the following script where we keep the compatibility level of the database to 150 which represents SQL Server 2019.

-- 2019

ALTER DATABASE [AdventureWorks2017] SET COMPATIBILITY\_LEVEL = 150

GO

-- Comparing Regular Index with ColumnStore Index

USE AdventureWorks2017

GO

SET STATISTICS IO, TIME ON

GO

-- Select Table with regular Index

SELECT ProductID, SUM(UnitPrice) SumUnitPrice, AVG(UnitPrice) AvgUnitPrice,

SUM(OrderQty) SumOrderQty, AVG(OrderQty) AvgOrderQty

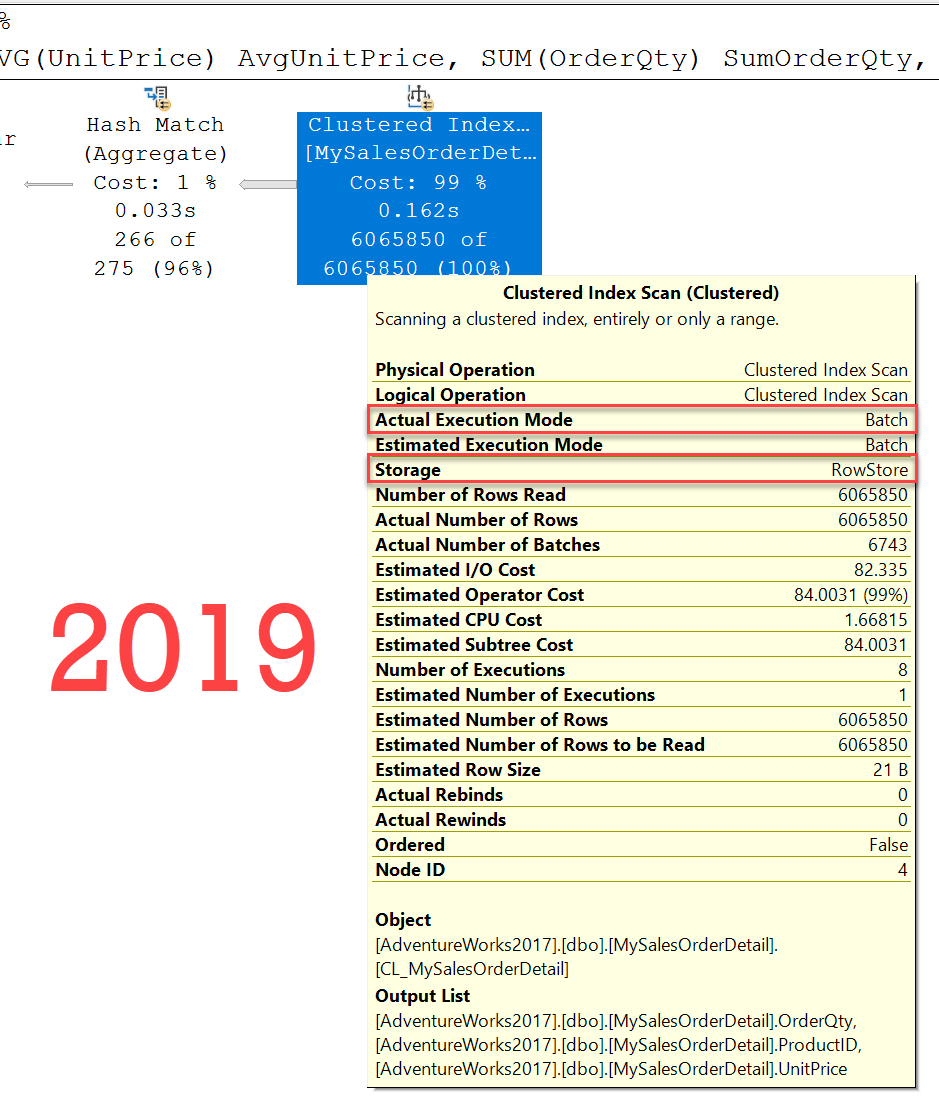
FROM [dbo].[MySalesOrderDetail]

GROUP BY ProductID

ORDER BY ProductID

GO

Check the execution plan of the query.



I have done a performance test on various queries and I figured out that whenever the batch mode is triggered for the query, the overall performance is improved by 20% to 30%. I will have a detailed post about the performance comparison in the near future.