Sergio Govoni

T-SQL magic tricks!

Materials: https://bit.ly/3kQU10w

Sergio Govoni

- CTO @ Centro Software
- Microsoft Data Platform MVP
- UGISS Vice President

Your contacts

- Linkedin: linkedin.com/in/sgovoni
- Twitter: twitter.com/segovoni

Your sites

- GitHub: github.com/segovoni
- Blog: segovoni.medium.com
- UGISS: www.ugiss.org
- MVP: mvp.microsoft.com/it-it/PublicProfile/4029181
- Sessionize: sessionize.com/sergio-govoni

Agenda

- SARGable predicates
  - NULLs
  - Dynamic sorting
- Query mode execution
- Join order
- Table aliases

SARGable predicates

The definition of SARGable

Wikipedia (en.wikipedia.org/wiki/Sargable) defines SARGability in this way:

In relational databases, a condition (or predicate) in a query is said to be sargable if the DBMS engine can take advantage of an index to speed up the execution of the query. The term is derived from a contraction of **Search ARGument ABLE**.

A query failing to be sargable is known as a non-sargable query and typically has a negative effect on query time, so one of the steps in query optimization is to convert them to be sargable. The effect is similar to searching for a specific term in a book that has no index, beginning at page one each time, instead of jumping to a list of specific pages identified in an index.

SARGable predicates

- SARGable means that the predicate can be evaluated/executed using a Seek
- Predicates

 <expression><operator><expression>

 <column><operator><expression>

DEMO

SARGable/non-SARGable predicates

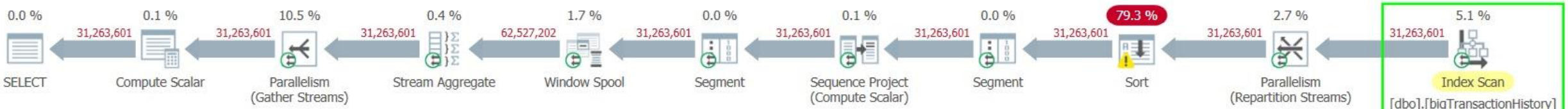
Query mode processing

Row mode execution

- Row mode execution is a query processing method used with traditional RDBMS tables, where data is stored in row format
- When a query is executed and accesses data in row store tables, the execution tree operators and child operators read each required row, across all the columns specified in the table schema
- From each row that is read, SQL Server retrieves the columns that are required for the result set, as referenced by a SELECT statement, JOIN predicate, or filter predicate

docs.microsoft.com

Row mode execution



Sort
Reorders the input.

Node ID: 7
Physical Operation: Sort
Logical Operation: Sort
Estimated Execution Mode: Row
Estimated Rows: 31,263,600
Estimated I/O Cost: 983.4300000
Estimated CPU Cost: 805.0430000
Estimated Executions: 1.0
Estimated Operator Cost: 1,788.4730000 (79.3%)
Estimated Subtree Cost: 1,964.1900000
Estimated Row Size: 19 B
Estimated Data Size: 566 MB
Memory Fractions:
In: 100.00%
Out: 100.00%

Order By:
[AdventureWorks2017].[dbo].[bigTransactionHistory].[ProductID] Ascending
[AdventureWorks2017].[dbo].[bigTransactionHistory].[TransactionID] Ascending

Output List:
[AdventureWorks2017].[dbo].[bigTransactionHistory].[ProductID]
[AdventureWorks2017].[dbo].[bigTransactionHistory].[Quantity]
[AdventureWorks2017].[dbo].[bigTransactionHistory].[TransactionID]

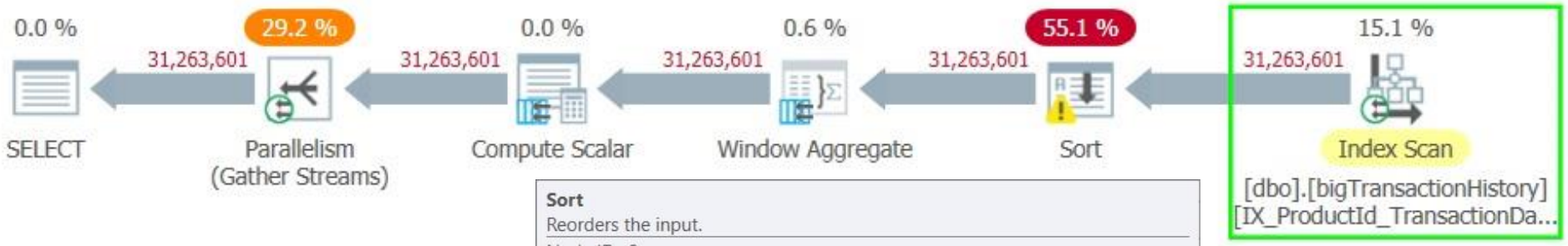
#DataWeekender

Batch mode execution

- Batch mode execution is a query processing method used to process multiple rows together, query operators process data more efficiently
- Each column within a batch is stored as a vector in a separate area of memory, so batch mode processing is vector-based
- Batch mode processing operates on compressed data when possible, and eliminates the exchange operator used by row mode execution. The result is better parallelism and faster performance

docs.microsoft.com

Batch mode execution



Sort
Reorders the input.

Node ID: 3
Physical Operation: Sort
Logical Operation: Sort
Estimated Execution Mode: Batch
Estimated Rows: 31,263,600
Estimated I/O Cost: 337.1887000
Estimated CPU Cost: 80.5043000
Estimated Executions: 1.0
Estimated Operator Cost: 417.6930000 (55.1%)
Estimated Subtree Cost: 532.1950000
Estimated Row Size: 19 B
Estimated Data Size: 566 MB
Memory Fractions:
In: 100.00%
Out: 100.00%

Order By:
[AdventureWorks2017].[dbo].[bigTransactionHistory].[ProductID] Ascending
[AdventureWorks2017].[dbo].[bigTransactionHistory].[TransactionID] Ascending

Output List:
[AdventureWorks2017].[dbo].[bigTransactionHistory].[ProductID]
[AdventureWorks2017].[dbo].[bigTransactionHistory].[Quantity]
[AdventureWorks2017].[dbo].[bigTransactionHistory].[TransactionID]

#DataWeekender

Columnstore and query mode execution

- SQL Server 2012 introduced a new feature to accelerate analytical workloads: **columnstore indexes**
  - SQL Server expanded the use cases and improved the performance of columnstore indexes in each subsequent release
- SQL Server 2016 enables the creation of **empty filtered columnstore indexes**
- Up to SQL Server 2017 batch mode processing requires a columnstore index to be enabled
- Starting with SQL Server 2019 (15.x) and in Azure SQL Database, batch mode execution no longer requires columnstore indexes, the feature is called Batch mode on rowstore!

DEMO

Query mode processing

Join order

Join order

- Query Optimizer must find the optimal sequence of joins between the tables used in the query, it defines the join order
- Finding the optimal join order is one of the most difficult problems in query optimization and it has be done within the available time
- Does the Query Optimizer analyze all possible join orders? No, it doesn't! It finds a balance between the optimization time and the quality of the resulting plan

Join order

Please, consider this query…

```
SELECT
  C.CustomerName, PS.SupplierName
FROM Sales.Customers AS C
INNER JOIN Sales.Orders AS O
  ON O.CustomerID=C.CustomerID
INNER JOIN Sales.OrderLines AS OL
  ON O.OrderID=OL.OrderID
INNER JOIN Warehouse.StockItems AS S
  ON OL.StockItemID=S.StockItemID
INNER JOIN Purchasing.Suppliers AS PS
  ON S.SupplierID=PS.SupplierID;
```

Supplier-Customer that have joint activity

Now imagine that you want to preserve customers who have no orders..

Join order

```
SELECT
  C.CustomerName, PS.SupplierName
FROM Sales.Customers AS C
LEFT OUTER JOIN Sales.Orders AS O
  ON O.CustomerID=C.CustomerID
INNER JOIN Sales.OrderLines AS OL
  ON O.OrderID=OL.OrderID
INNER JOIN Warehouse.StockItems AS S
  ON OL.StockItemID=S.StockItemID
INNER JOIN Purchasing.Suppliers AS PS
  ON S.SupplierID=PS.SupplierID;
```

Query optimizer has detected the contradiction…

Hash Keys Build	[WideWorldImporters].[Sales].[Customers].Custo
Alias	[C]
Column	CustomerID
Database	[WideWorldImporters]
Schema	[Sales]
Table	[Customers]
Hash Keys Probe	[WideWorldImporters].[Sales].[Orders].Custome
Alias	[O]
Column	CustomerID
Database	[WideWorldImporters]
Schema	[Sales]
Table	[Orders]
Logical Operation	Inner Join

#DataWeekender

DEMO

Join order

Table aliases

Table aliases

- Introducing table aliases you can change the meaning of the query and potentially the results
- The basic rule is that SQL Server tries to match within the same scope and only goes to an outer scope if needed
- Pay attention to the potentially not correlated predicates
- If you have a query that uses more than one table **always use aliases for all tables and always prefix each column with the proper alias**!

DEMO

Table aliases

Summary

- One of the steps in the query optimization process is to convert non-sargable predicates to sargable predicates
  - Pay attention to NULLs
- SQL Server 2016 enables the creation of empty filtered columnstore indexes that you can use to enable batch mode execution in the OLTP scenarios without maintenance costs on columnstore indexes
- The logical join ordering is determined by the order of ON clauses
- If you have a query that uses more than one table always use aliases for all tables

Resources

- Sargable predicates
  - https://segovoni.medium.com/sargable-predicates-and-null-values-in-sql-server-c43ec3d8b108
- Query mode execution
  - https://www.ugiss.org/2022/02/16/modalita-di-elaborazione-query-e-indici-columnstore/
  - https://bit.ly/3Hmcyuf
  - An article will be coming to my English blog soon ☺ here: https://segovoni.medium.com/
- Thinking Big (Adventure) by Adam Machanic
  - http://dataeducation.com/thinking-big-adventure/
- Session materials on Github
  - https://bit.ly/3kQU10w

#DataWeekender

Thanks for attending #DataWeekender CU5!