#### 1.2. How to read an execution plan.

There are two types of execution plans: estimated and actual. In the 99 per cent time estimated is the same as actual will be.

Plans should be read from right to left and top to bottom.

The size of arrows is indicating the amount of data flowing to the next operator.

SET STATISTICS IO, TIME ON --> is gonna give us more information about io operations, it will breaks out to logical reads from memory and logical reads from disk. TIME ON will give us the actual execution time and the cpu time.

#### 1.4. Key lookup: when it is good and when it is bad

A key lookup is a lookup against the clustered index. The lookup will execute once for each row, which is a very expensive operation.

INCLUDE columns are not key columns in the index, so they are not ordered. This means it isn't really useful for predicates, sorting etc as I mentioned above. However, it may be useful if you have a residual lookup in a few rows from the key column(s)

#### 1.5. Nested loop vs hash join

SQL Server employs three types of joins to bring together data: **Merge**, **hash** and **nested loop** joins:

* Hash joins are useful for large, unsorted, and non-indexed data
* Merge joins are used for large but sorted and indexed data
* Nested loop joins are good for small sets of data

You can hint your join type in query, sometime optimizer is making wrong decision

SQL Server 2017 introduces deferred join selection within plans that execute in batch execution mode. This deferred join selection will switch from nested loops to hash and vice versa, based on statistics generated as the query executes.

#### 1.6. Query Store

Introduced in SQL Server 2016. Personal favorite feature for author of this course. It is not enabled by default.

#### 1.8. Missing index warnings

SELECT s.last\_user\_seek ,

d.object\_id ,

d.equality\_columns ,

d.inequality\_columns ,

d.included\_columns ,

d.statement ,

s.avg\_user\_impact

FROM sys.dm\_db\_missing\_index\_group\_stats AS s

INNER JOIN sys.dm\_db\_missing\_index\_groups AS g ON ( s.group\_handle = g.index\_group\_handle )

INNER JOIN sys.dm\_db\_missing\_index\_details AS d ON ( g.index\_handle = d.index\_handle );

#### 3.5. What datatype conversions do to your query performance

when using where on nvarchar you shoud use N letter before value

#### 4.1. Temp tables

tempdb main task is to store and retain temporary tables. Temp tables are tight to session, they're go away when session is closed. If you're using READ\_COMMITTED\_SNAPSHOT isolation level, which is an isolation level that allows users to read data while other users are writing data, tempdb is going to hold the version history of that data. If you're using an always-on availability group, and you're on a readable secondary, SQL Server is going to create stats and store them in tempdb when you query off that readable secondary.

* You should create multiple tempdb files to redue latch contention on the underlying data files (this is done by default in SQL Server 2016 and above)

#### 4.2. Table variables

They shouldn't be used for passing large amount of data. They can be passed into procedure as parameter.

#### 4.3 In-memory temp tables

* This structure can add overhead to buffer pool
* There are configuration changes to the server that need to be made to enable in-memory OLTP.
* This mechanism avoids logging data, so performance is extremely fast
* Can replace both global and local temporary tables

#### 5.1. Choosing a PK or clustered index key

Specifically we look for two types of operations. The first is when specific data is returned from a page we call that an index seek. And when SQL Server scans all of the pages in order to locate the required index we call that an index scan. Seeks are almost always more efficient.

uniqueifier is added to every clustered index with non-unique values (4 bytes).