SQLServerFast.com Execution Plan Video Training

Block 2: Reading data

Level: Advanced

Chapter 5: Assorted read optimizations

Read-ahead reads

- Read-ahead reads
  - Implemented in storage engine
  - Not directly exposed in execution plan
    - Impacts performance
      - Usually good
      - Sometimes not so good
    - Might influence some run-time statistics in the execution plan plus

Read-ahead reads

- Read-ahead reads
  - Implemented in storage engine
  - Not directly exposed in execution plan
  - Only used for physical I/O
  - Only used when scanning data
    - Includes Index Seek doing a range seek

Read-ahead reads

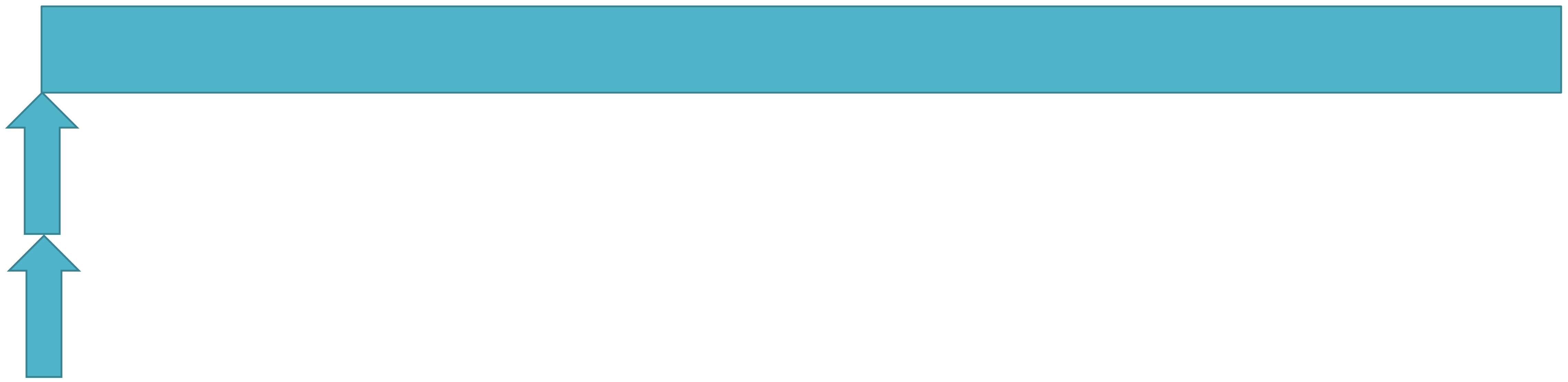
- Read-ahead reads
  - Implemented in storage engine
  - Not directly exposed in execution plan
  - Only used for physical I/O
  - Only used when scanning data
  - Requests needed page ***and the pages after***
    - Data (hopefully) already in buffer pool when needed
    - Benefits from large buffer size for I/O requests

Advanced scan

- Advanced scan
  - Also known as “merry-go-round scan” or “piggyback scan”
  - Enterprise Edition only

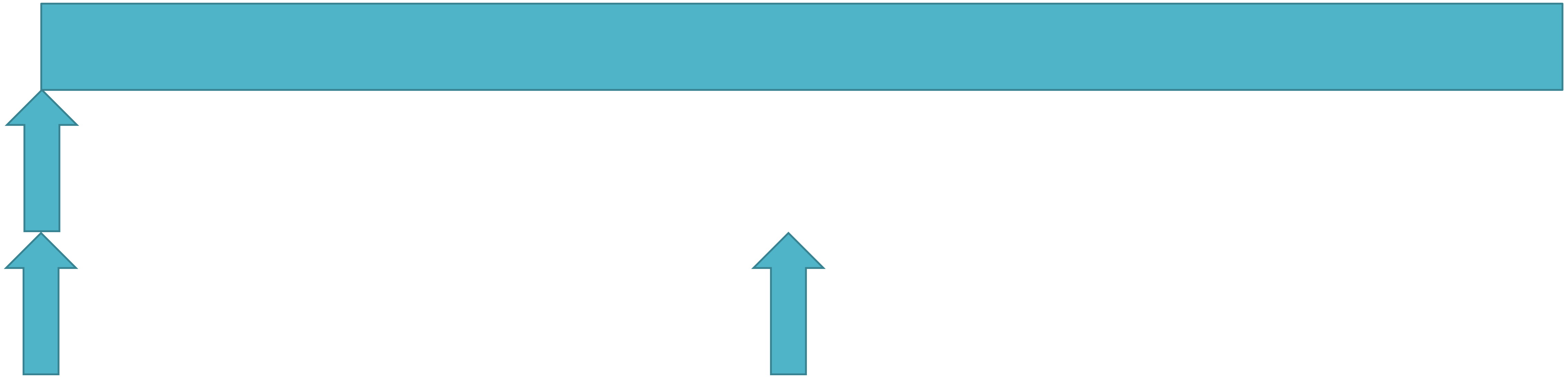
Advanced scan

- Advanced scan
  - Normal behavior



Advanced scan

- Advanced scan
  - Merry-go-round scan



Advanced scan

- Advanced scan
  - Also known as “merry-go-round scan” or “piggyback scan”
  - Enterprise Edition only
  - New scan connects to scan in progress
  - Each page read is available for both scans
  - Original scan disconnects from shared scan when done
  - Second scan wraps back to start and reads until its own starting point

Advanced scan

Advanced scan

Possible combinations

First query

Second query		Ordered = True	Leaf page scan	IAM scan
	Ordered = True	NO	NO	NO
	Leaf page scan			
	IAM scan			

Advanced scan

Advanced scan

Possible combinations

First query

Second query		Ordered = True	Leaf page scan	IAM scan
	Ordered = True	NO	NO	NO
	Leaf page scan			NO
	IAM scan			

Advanced scan

Advanced scan

Possible combinations

First query

Second query

	Ordered = True	Leaf page scan	IAM scan
Ordered = True	NO	NO	NO
Leaf page scan	YES	YES	NO
IAM scan			

Advanced scan

Advanced scan

Possible combinations

First query

Second query		Ordered = True	Leaf page scan	IAM scan
	Ordered = True	NO	NO	NO
	Leaf page scan	YES	YES	NO
	IAM scan			YES

Advanced scan

Advanced scan

Possible combinations

First query

Second query	Ordered = True	Leaf page scan	IAM scan
Ordered = True	NO	NO	NO
Leaf page scan	YES	YES	NO
IAM scan	YES *	YES *	YES

* runtime switch to leaf page scan

Advanced scan

Advanced scan

Actually implemented combinations

First query

Second query

	Ordered = True	Leaf page scan	IAM scan
Ordered = True	NO	NO	NO
Leaf page scan	NO	NO	NO
IAM scan	NO	NO	YES

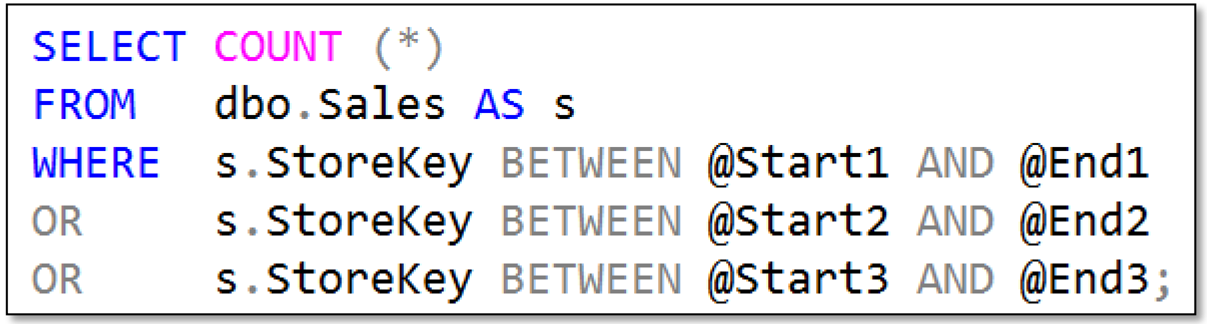
Advanced scan

- Advanced scan
  - Also known as “merry-go-round scan” or “piggyback scan”
  - Enterprise Edition only
  - Only implemented for IAM scans
    - Second scan connects to first, then wraps around to start
    - Not limited to two scans: third, fourth, etc. can also connect

Dynamic seek range

- Dynamic seek range
  - Multiple ranges with known values
    - Optimizer reorders
    - Optimizer detects and collapses overlapping ranges
      - Otherwise duplicate rows would be returned!

Dynamic seek range



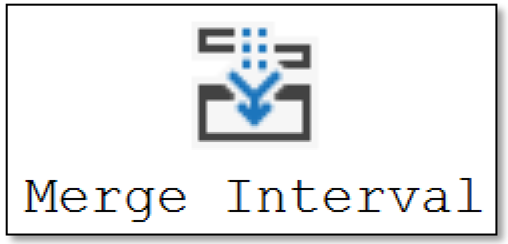
```
SELECT COUNT (*)
FROM   dbo.Sales AS s
WHERE  s.StoreKey BETWEEN @Start1 AND @End1
OR     s.StoreKey BETWEEN @Start2 AND @End2
OR     s.StoreKey BETWEEN @Start3 AND @End3;
```

- Dynamic seek range
  - Multiple ranges with known values
  - Multiple ranges with variables (unknown values)
    - Still need to detect and collapse overlapping intervals
      - Constant Scan + Concatenation to get each interval in a row
      - Sort operator to sort the intervals in order
      - Merge Interval operator detects and collapses overlapping intervals

Merge Interval

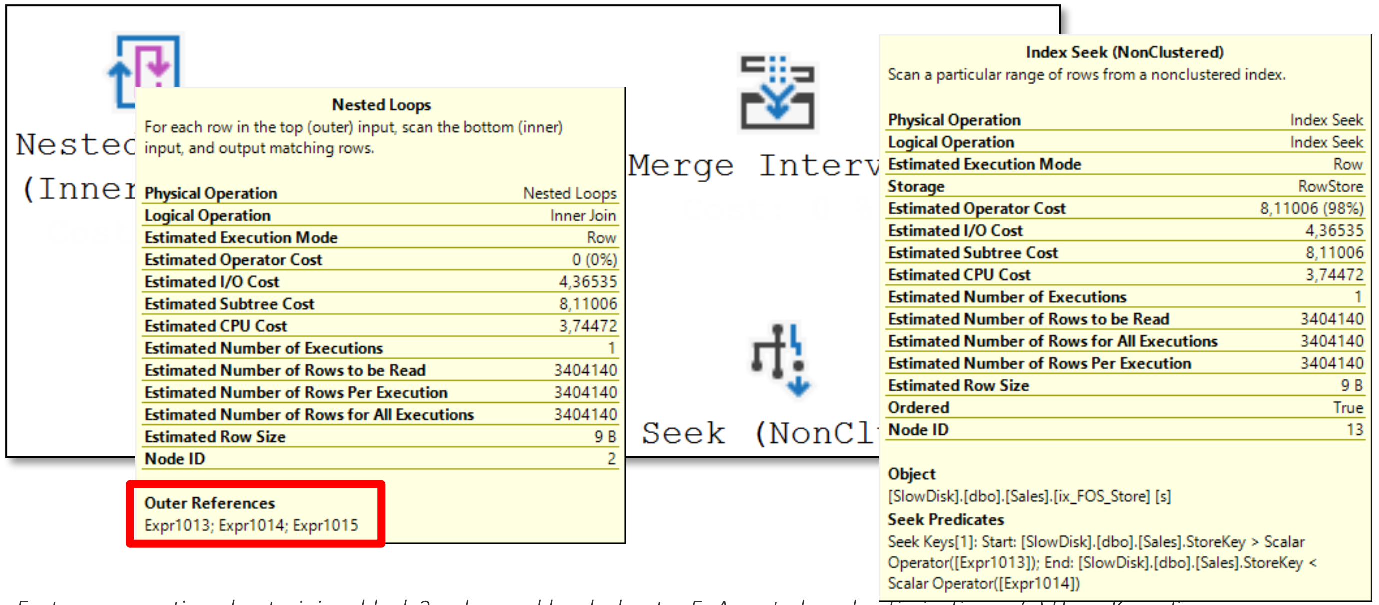
Dynamic seek range

- Merge Interval operator
  - No properties used to control its behavior
  - Fixed input and output columns
    - Column 1 and 2: Start and end of interval
    - Column 3: Bitmap to define boundary behavior
    - Columns 4 to 6 (input only): Derived from columns 1 to 3
      - Materialized to facilitate sorting
      - Not actually needed for operator (but might still be used internally)



Dynamic seek range

- Merge Interval operator
  - Third column unused in Index Seek?
    - Probably overlooked in conversion from internal plan representation to XML
    - Does get pushed into Index Seek by the Nested Loops



Nested Loops

For each row in the top (outer) input, scan the bottom (inner) input, and output matching rows.

Physical Operation	Nested Loops
Logical Operation	Inner Join
Estimated Execution Mode	Row
Estimated Operator Cost	0 (0%)
Estimated I/O Cost	4,36535
Estimated Subtree Cost	8,11006
Estimated CPU Cost	3,74472
Estimated Number of Executions	1
Estimated Number of Rows to be Read	3404140
Estimated Number of Rows Per Execution	3404140
Estimated Number of Rows for All Executions	3404140
Estimated Row Size	9 B
Node ID	2

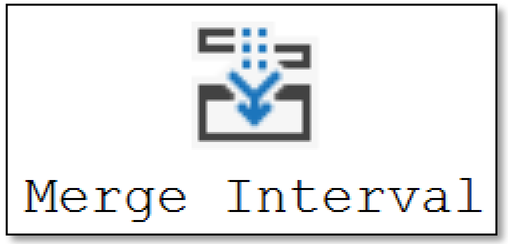
Outer References
Expr1013; Expr1014; Expr1015

Index Seek (NonClustered)

Scan a particular range of rows from a nonclustered index.

Physical Operation	Index Seek
Logical Operation	Index Seek
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	8,11006 (98%)
Estimated I/O Cost	4,36535
Estimated Subtree Cost	8,11006
Estimated CPU Cost	3,74472
Estimated Number of Executions	1
Estimated Number of Rows to be Read	3404140
Estimated Number of Rows for All Executions	3404140
Estimated Number of Rows Per Execution	3404140
Estimated Row Size	9 B
Ordered	True
Node ID	13

Object
[SlowDisk].[dbo].[Sales].[ix_FOS_Store] [s]
Seek Predicates
Seek Keys[1]: Start: [SlowDisk].[dbo].[Sales].StoreKey > Scalar Operator([Expr1013]); End: [SlowDisk].[dbo].[Sales].StoreKey < Scalar Operator([Expr1014])



Partitioning

- Partitioned tables or indexes
  - Implemented as multiple, independent objects
  - E.g. partitioned clustered index is actually multiple B-trees
    - Each individual B-tree is called a “partition”
  - Value in “partitioning column” determines which partition to use

Partitioning

- Partitioned tables or indexes
  - Seek operator
    - Partition(s) determined by PtnId*nnnn* in *Seek Predicates* property
  - Scan operator
    - Partition(s) determined by PtnId*nnnn* in *Seek Predicates* property
    - This is the only case where a scan operator can have a *Seek Predicates* property
  - RangePartionNew() function used to find partition number at runtime

Summary

Read-ahead reading

Advanced scan (aka “merry-go-round scan”)

Dynamic range seeks

Partitioning

Summary

Block 2, basic level

- Storage structures
  - For rowstore data
- Scan operators
- Seek operators
- Lookup operators
- Special scans

Block 2, advanced level

- Other storage structures
  - Columnstore
  - Memory-optimized
- Other index types
- Parallel and batch mode plans
- Other optimizations

Next chapters

Block 3: Combining data – basic level

- Logical join types
- Physical join operators
  - Nested Loops
  - Merge Join
  - Hash Match
  - Adaptive Join
- Other combining operators

Next chapters

Block 3: Combining data – basic level

Block 3: Combining data – advanced level

- More details about the physical join operators
  - Nested Loops (advanced)
  - Merge Join (advanced)
  - Hash Match (advanced)
  - Adaptive Join (advanced)