

SQLServerFast.com

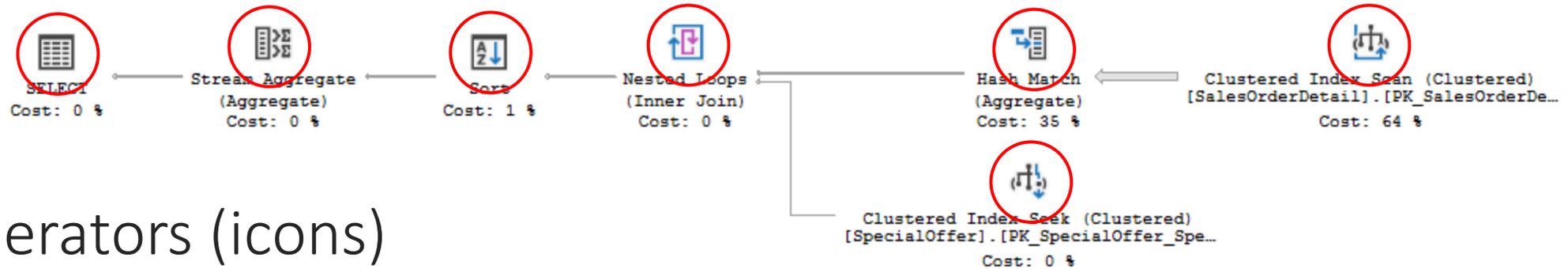
Execution Plan Video Training

Block 1: Understanding execution plans

Level: Basic

Chapter 3: How to read an execution plan

Elements of an execution plan



Operators (icons)

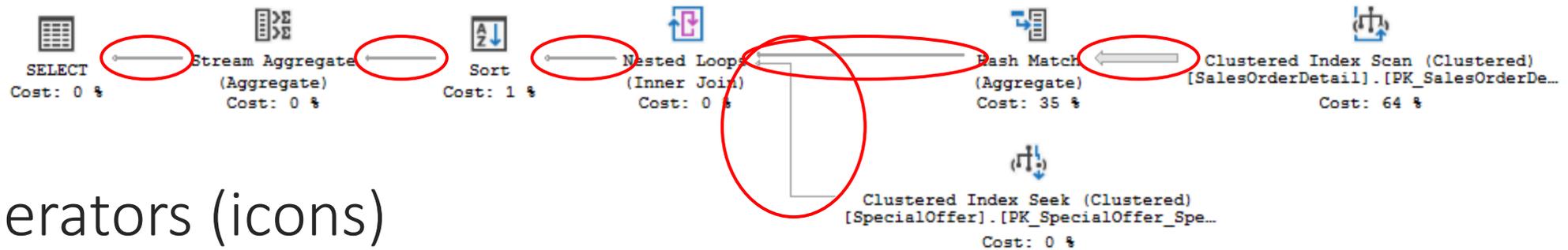
Small, specialized programs

Each does their own task

Receive data to process

Pass processed data to next operator

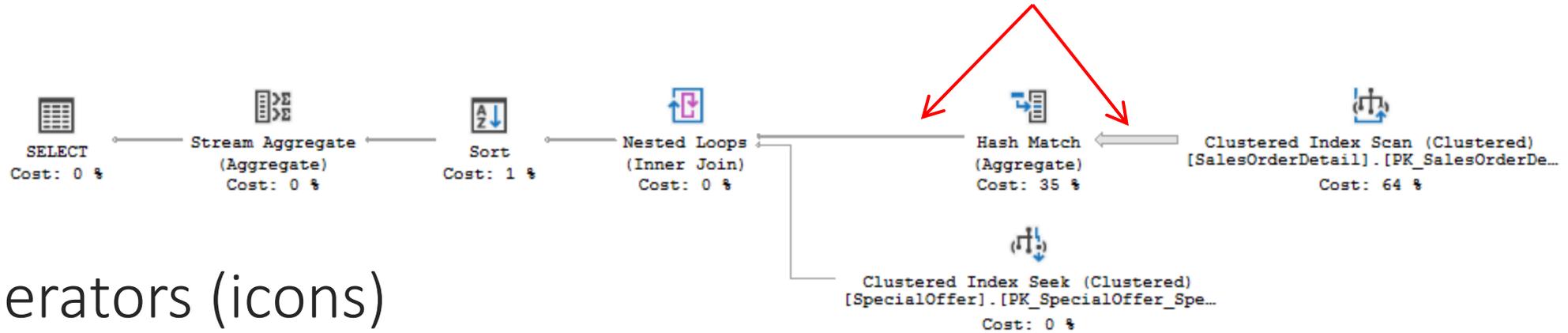
Elements of an execution plan



Operators (icons)

Data flows (arrows)

Elements of an execution plan



Operators (icons)

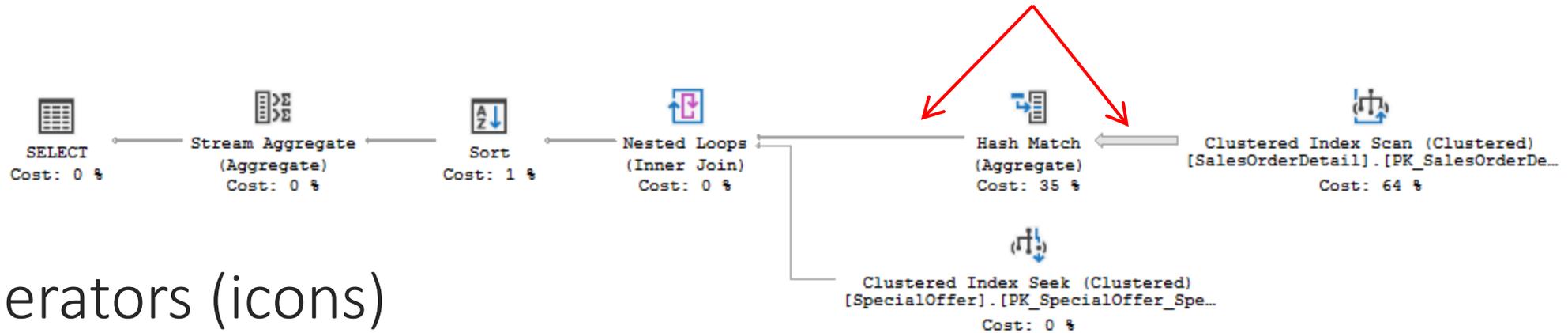
Data flows (arrows)

Width of arrow scales with number of rows

Based on actual row count when available

(Execution plan plus run-time statistics)

Elements of an execution plan



Operators (icons)

Data flows (arrows)

Width of arrow scales with number of rows

Based on actual row count when available

Based on estimated row count otherwise

Some exceptions apply

*More information about arrow width
in chapter 6: Cardinality in the execution plan*

Elements of an execution plan

Operators (icons)

Small computer programs

Run when requested

Three “methods” (request types)

Initialize()

Prepare to do work

Prepare to *restart* work

Result may be different

Underlying data changed

New parameter value passed in

Elements of an execution plan

Operators (icons)

Small computer programs

Run when requested

Three “methods” (request types)

Initialize()

GetNext()

Produces a single row

(How this is done varies by operator)

Returns that row to calling operator

Execution pauses

Resumes upon next GetNext() call

Result is either a row, or an “end of data” signal

Elements of an execution plan

Operators (icons)

Small computer programs

Run when requested

Three “methods” (request types)

Initialize()

GetNext()

Close()

Clean up, release memory, close dependent operators, etc.

Called once, at end of work

Not called for operators that were never initialized

Elements of an execution plan

Operators (icons)

Small computer programs

Run when requested

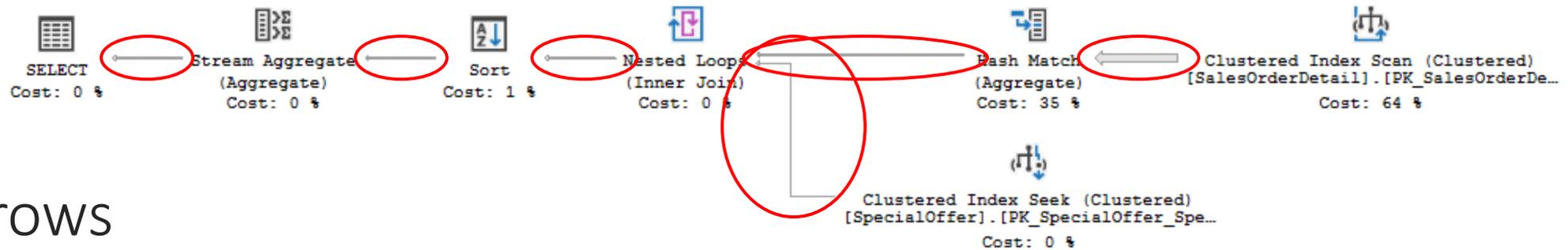
Three “methods” (request types)

Initialize() ← May be important, in some cases

GetNext() ← Most important for understanding execution plans

Close() ← No need to ever look at

Elements of an execution plan

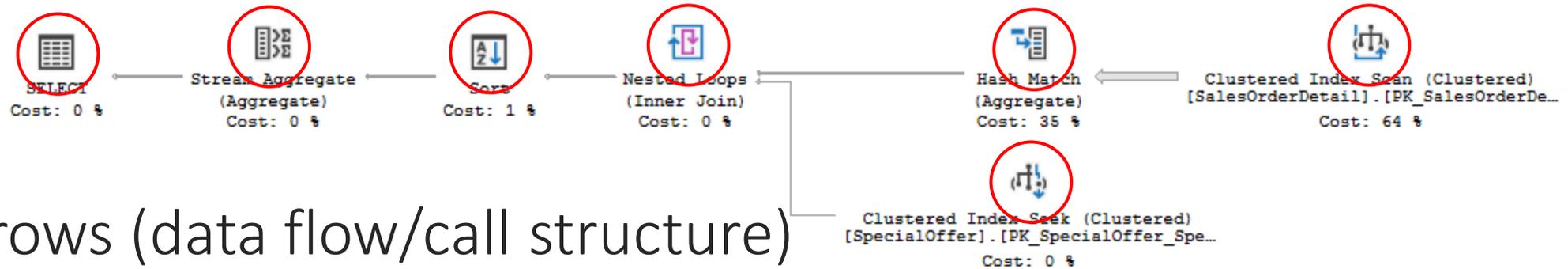


Arrows

Data flows

But also: call structure

Elements of an execution plan



Arrows (data flow/call structure)

All operators pass data to exactly one operator

Reverse is not true

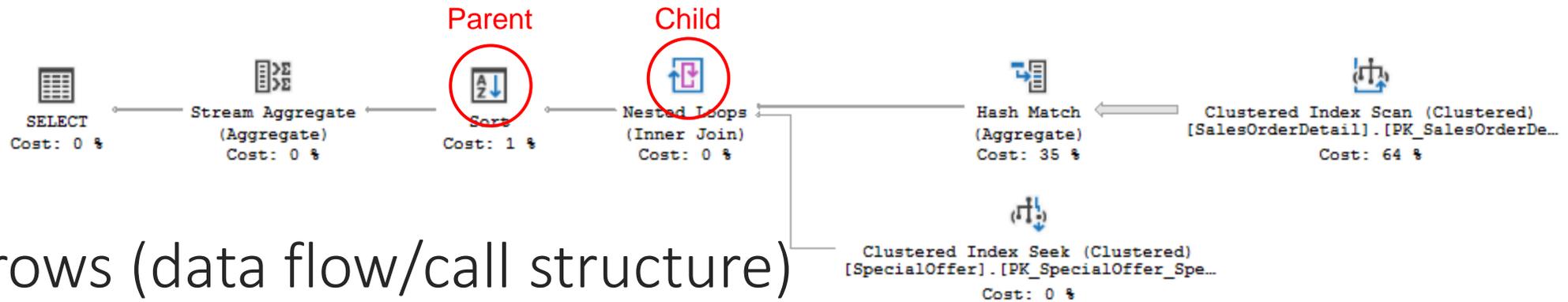
Most operators call and receive data from one operator

Some operators call no operators at all

Some operators call two or more operators

There is no upper limit

Elements of an execution plan



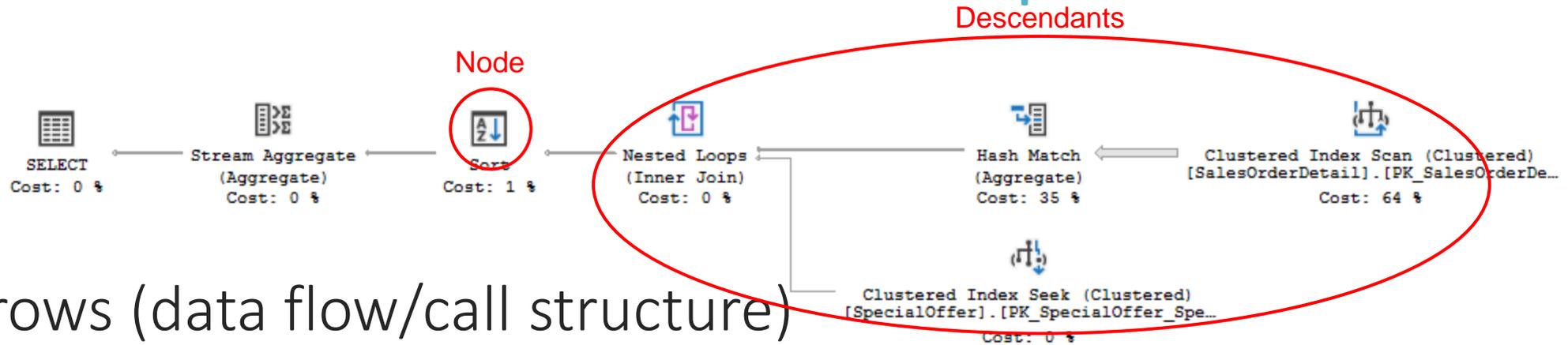
Arrows (data flow/call structure)

“Directed tree” (graph theory)

Operators may be called “nodes”

“Parent” calls “child”

Elements of an execution plan



Arrows (data flow/call structure)

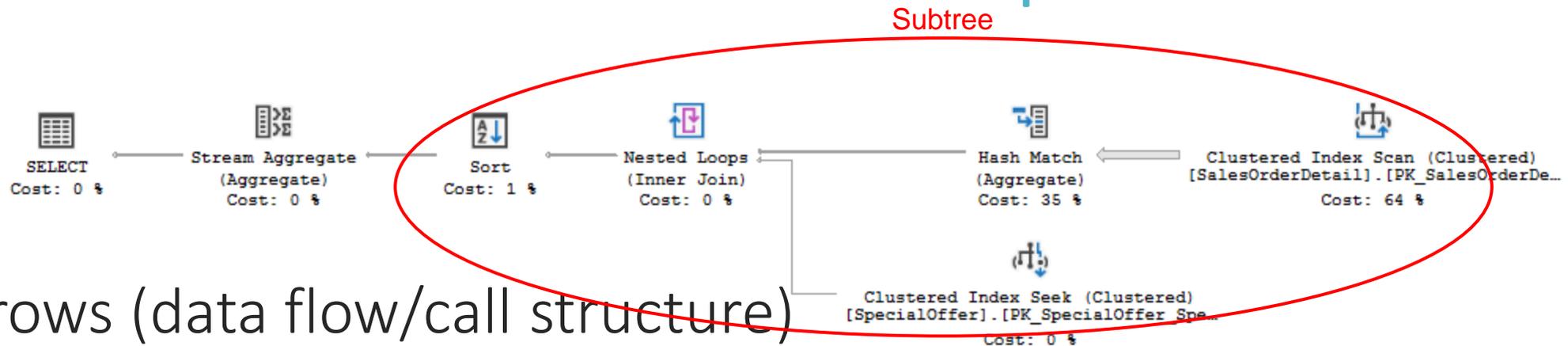
“Directed tree” (graph theory)

Operators may be called “nodes”

“Parent” calls “child”

“Descendants”: all children, and their children, and so on

Elements of an execution plan



Arrows (data flow/call structure)

“Directed tree” (graph theory)

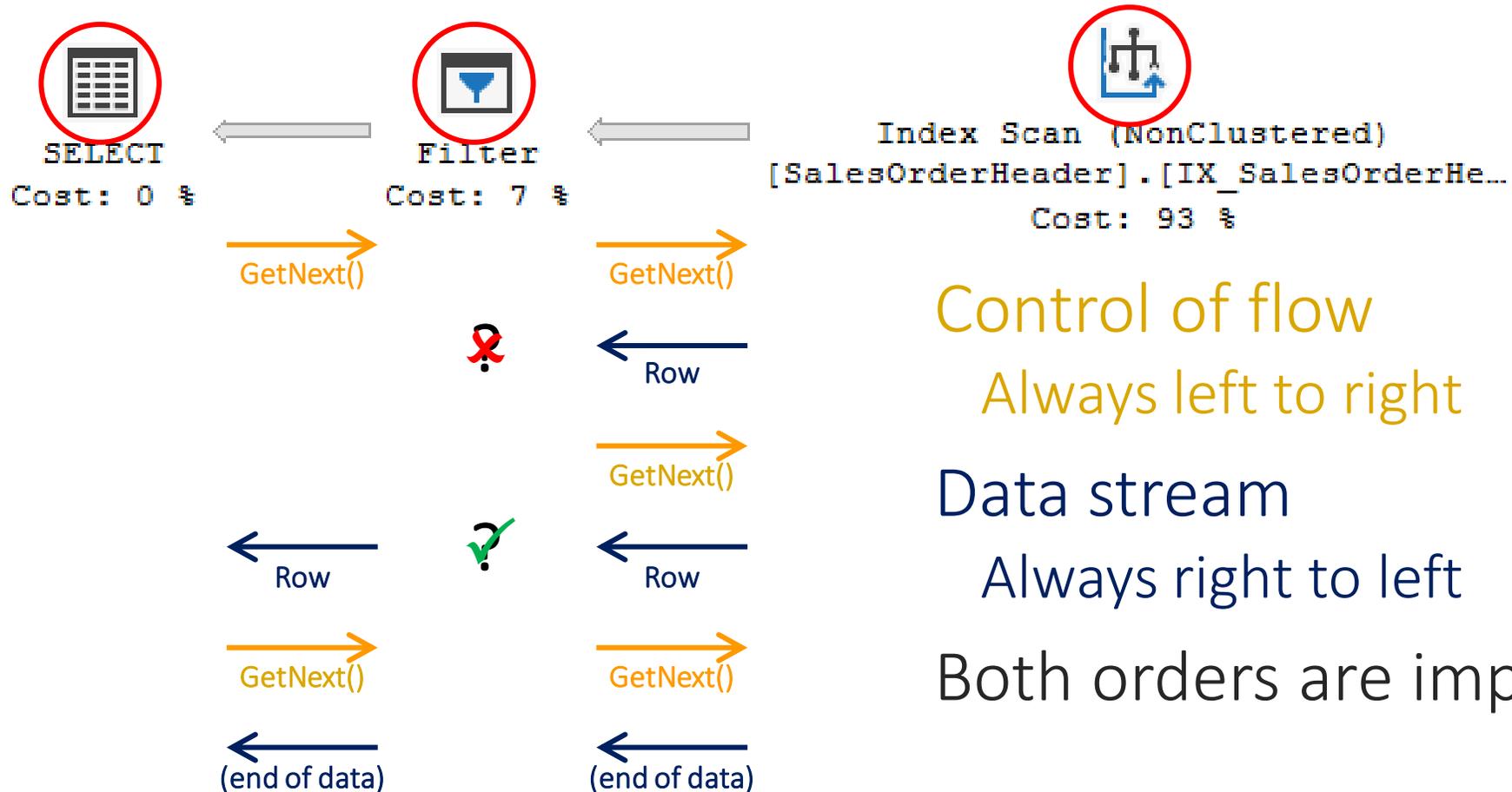
Operators may be called “nodes”

“Parent” calls “child”

“Descendants”: all children, and their children, and so on

“Subtree”: node plus all descendants

Operator interactions



Control of flow
Always left to right

Data stream
Always right to left

Both orders are important!

“Bad” operators?

“Scans are bad, you need an index!”

Scans are actually ideal for reading most rows, or just the first few rows

“Key Lookup? Create a covering index!”

What if the cost of that extra (or extra wide) index exceeds the benefit?

“Oh no, a Table Spool, let’s rewrite the query!”

Chosen too often by optimizer, but still good in some cases

“Bad” operators?

No operator is “bad”

(Almost) all operators *can be* bad ... *in a specific context*

Caused by interaction with other operators ...

... combined with the specific data in the tables

Requires understanding of operators and execution plans

Summary

Operators

- Small programs

- Called by one operator, calls zero, one, or more operators

- GetNext() call: returns a row of data

Order of control: left to right

Order of data flow: right to left

No “bad” operators

- But operators *can* be bad when used incorrectly

Next chapters

Chapter 4: Properties

What are properties, why are they important?

Where to find

Dangerous misconceptions

Chapter 5: Where to find execution plans

Chapter 6: Cardinality in the execution plan

Chapter 7: Percentages in the execution plan