

SQLServerFast.com

Execution Plan Video Training

Block 2: Reading data

Level: Basic

Chapter 5: Special scans

Special scans

Eight different scan operators

Clustered Index Scan

Columnstore Index Scan

Constant Scan

Deleted Scan

Index Scan

Inserted Scan

Remote Scan

Table Scan

Or more???

Log Row Scan

Parameter Table Scan

Remote Index Scan

Mentioned on some websites

No details given

I've never seen them

Relics of older versions?

Special scans

Eight different scan operators

Clustered Index Scan

Columnstore Index Scan

Constant Scan

Deleted Scan

Index Scan

Inserted Scan

Remote Scan

Table Scan

Special scans

Eight different scan operators

Clustered Index Scan

Columnstore Index Scan

Constant Scan

Deleted Scan

Index Scan

Inserted Scan

Remote Scan

Table Scan

Special scans

Eight different scan operators

Clustered Index Scan

Columnstore Index Scan

Constant Scan

Deleted Scan

Index Scan

Inserted Scan

Remote Scan

Table Scan

Special scans

Constant Scan

Creates row without external source (table, index)
Data for rows stored in execution plan itself



Constant Scan

Key Lookup

Properties (popup)

Data to return
(multiple rows;
multiple columns)

Column names for
columns to return

Constant Scan	
Scan an internal table of constants.	
Physical Operation	Constant Scan
Logical Operation	Constant Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Actual Number of Rows for All Executions	2
Actual Number of Batches	0
Estimated I/O Cost	0
Estimated Operator Cost	0,0000022 (100%)
Estimated Subtree Cost	0,0000022
Estimated CPU Cost	0,0000022
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows Per Execution	2
Estimated Row Size	21 B
Actual Rebinds	0
Actual Rewinds	0
Node ID	0
Values	(Scalar Operator((1)); Scalar Operator('One'); Scalar Operator('First')); (Scalar Operator((2)); Scalar Operator('Two'); Scalar Operator('Second'))
Output List	Union1006; Union1007; Union1008



Special scans

Constant Scan

Creates row without external source (table, index)

Data for rows stored in execution plan itself

Makes the execution plan very large if used for many rows/columns

Takes up space in plan cache

May result in other plans being evicted

Possibly bad for performance!



Special scans

Constant Scan

Creates row without external source (table, index)

Data for rows stored in execution plan itself

Makes the execution plan very large if used for many rows/columns

Better to put large amounts of data in a permanent table

- Saves space in plan cache

- Reduces size of application code

- Less network traffic

- More tuning opportunities

- No code change / deploy if data needs to be changed



Special scans

Constant Scan

Creates row without external source (table, index)

Data for rows stored in execution plan itself

Sometimes used “creatively” by optimizer!

E.g. generate partition numbers to process partition table one partition at a time

E.g. produce empty row(s), when columns get added later

Empty Output List property

Bug / shortcoming: *Values* property not present in the execution plan XML



Constant Scan

Key Lookup

Properties (popup)

Use *Estimated Number of Rows* to deduct number of empty rows in *Values* property of the “real” plan

No *Output List* (empty in full list) to generate “empty” rows

No *Values* property (bug in XML representation) when generating empty rows

Constant Scan	
Scan an internal table of constants.	
Physical Operation	Constant Scan
Logical Operation	Constant Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Actual Number of Rows for All Executions	1
Actual Number of Batches	0
Estimated I/O Cost	0
Estimated Operator Cost	0,0000012 (0%)
Estimated Subtree Cost	0,0000012
Estimated CPU Cost	0,0000012
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows Per Execution	1
Estimated Row Size	98
Actual Rebinds	0
Actual Rewinds	0
Node ID	?



Special scans

Constant Scan

Creates row without external source (table, index)

Data for rows stored in execution plan itself

Sometimes used “creatively” by optimizer!

E.g. generate partition numbers to process partition table one partition at a time

E.g. produce empty row(s), when columns get added later

Empty Output List property

Bug / shortcoming: *Values* property not present in the execution plan XML

And many more

Check *Output List*



Special scans

Constant Scan

Creates row without external source (table, index)

Data for rows stored in execution plan itself

Sometimes used “creatively” by optimizer!

E.g. generate partition numbers to process partition table one partition at a time

E.g. produce empty row(s), when columns get added later

Empty Output List property

Bug / shortcoming: *Values* property not present in the execution plan XML

And many more

Check *Output List* and *Values* properties

Follow columns through rest of the execution plan



Special scans

Deleted Scan / Inserted Scan

Used in trigger code

Trigger:

Code that automatically runs after (or instead of) modifying data

Runs once after each modification statement

(So does **NOT** run once per row, as in some other DBMS's)

Data affected available through “pseudo-tables” *deleted* and *inserted*

Same schema as underlying table

For updated rows, old data goes to *deleted* and new data is in *inserted*



Deleted Scan



Inserted Scan

Special scans

Deleted Scan / Inserted Scan

Used in trigger code

Deleted Scan: Used to read data from *deleted* pseudo-table

Inserted Scan: Used to read data from *inserted* pseudo-table

Execution plan for trigger

Not visible in execution plan only

Request execution plan plus run-time statistics (or live execution plan) to see



Deleted Scan



Inserted Scan

Special Scans

Deleted Scan

Properties (popup)
(note: Inserted Scan is mostly the same)

Standard stuff, relevant for *all* operators

Table the trigger is defined on
(not the actual source of the data!)

Return only these columns

Deleted Scan	
Scanning the pseudo-table 'deleted' within a trigger.	
Physical Operation	Deleted Scan
Logical Operation	Deleted Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Actual Number of Rows for All Executions	2
Actual Number of Batches	0
Estimated Operator Cost	0,0032842 (100%)
Estimated I/O Cost	0,0032035
Estimated CPU Cost	0,0000807
Estimated Subtree Cost	0,0032842
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows Per Execution	2
Estimated Number of Rows to be Read	2
Estimated Row Size	11 B
Actual Rebinds	0
Actual Rewinds	0
Node ID	0
Object	[Playground].[dbo].[TriggerTest]
Output List	[Playground].[dbo].[TriggerTest].a

Deleted Scan

Inserted Scan

Special scans

Deleted Scan / Inserted Scan

Used in trigger code

Deleted Scan: Used to read data from *deleted* pseudo-table

Inserted Scan: Used to read data from *inserted* pseudo-table

Source of data: “temporary, memory-resident tables”

Insert: Add copy of new data to *inserted*

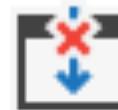
Source: <https://docs.microsoft.com/en-us/sql/relational-databases/triggers/use-the-inserted-and-deleted-tables?view=sql-server-ver15>

Delete: Copy rows to *deleted* before removing

Update: Copy original row to *deleted*, then copy new row to *inserted*

Not indexed

Supports scan only, no seek possible



Deleted Scan



Inserted Scan

Special scans

Deleted Scan / Inserted Scan

Used in trigger code

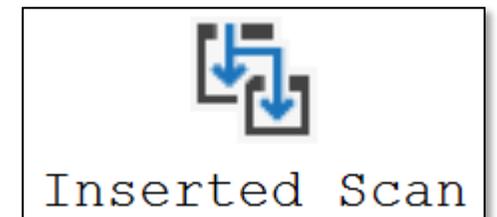
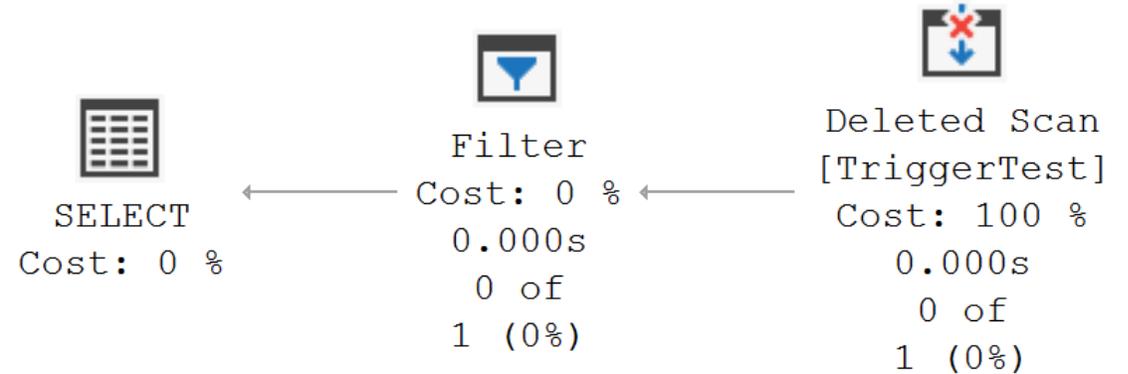
Deleted Scan: Used to read data from *deleted* pseudo-table

Inserted Scan: Used to read data from *inserted* pseudo-table

Source of data: “temporary, memory-resident tables”

No support for predicate pushdown

Filtering possibly via explicit Filter operator



Summary

Special scans

Constant Scan

- Creates rows without external source

- Data stored in execution plan

 - Beware plan size!

Deleted Scan / Inserted Scan

- Used in triggers

- Read data from *deleted* / *inserted* pseudo-table

Summary

Block 2, basic level

Storage structures

For rowstore data

Scan operators

Seek operators

Lookup operators

Special scans

Next chapters

Block 2: Reading data – advanced level

Reading from other index types

- Columnstore indexes

- Memory-optimized indexes

- Other indexes

 - XML indexes, spatial indexes, full-text indexes

Special cases for reading data

- Reading data in a parallel or batch mode plan

- Assorted read optimizations

Next chapters

Block 2: Reading data – advanced level

Block 3: Combining data – basic level

- Logical join operations

- Physical join operators

 - Nested Loops

 - Merge Join

 - Hash Match

 - Adaptive Join

- Other combining operators