INTELIGENTNO PROCESIRANJE U SQL SERVERU 2019

MILOŠ RADIVOJEVIĆ

DATA PLATFORM MVP

26. DECEMBAR 2020

MEMORY GRANT FEEDBACK

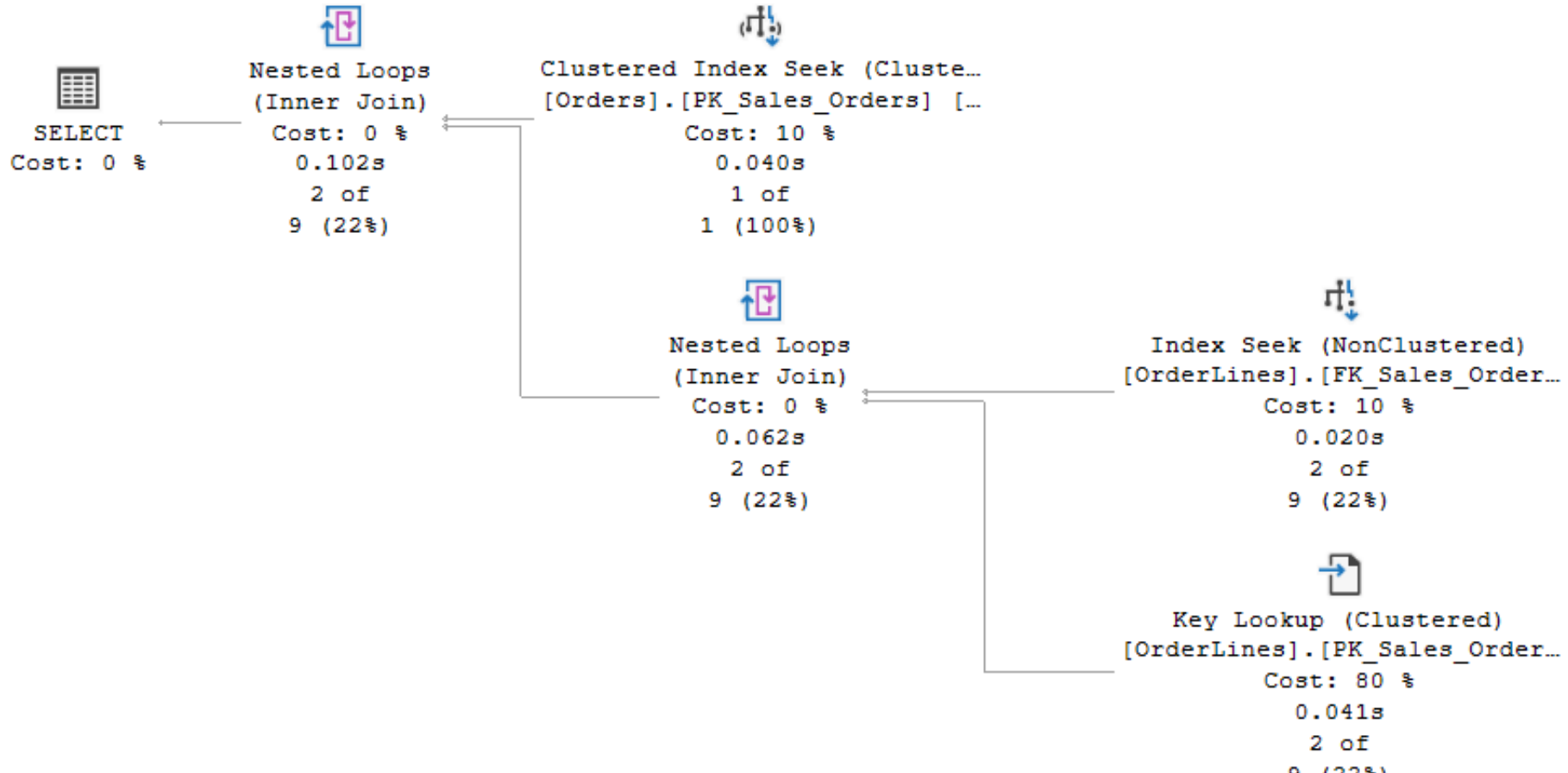
```sql
SELECT * FROM Sales.Orders o
INNER JOIN Sales.OrderLines ol ON o.OrderID = ol.OrderID
WHERE o.OrderID = 234;
```

100 %

Results | Messages | Execution plan

Query 1: Query cost (relative to the batch): 100%

SELECT * FROM Sales.Orders o INNER JOIN Sales.OrderLines ol ON o.OrderID = ol.OrderID WHERE o.OrderID = 234

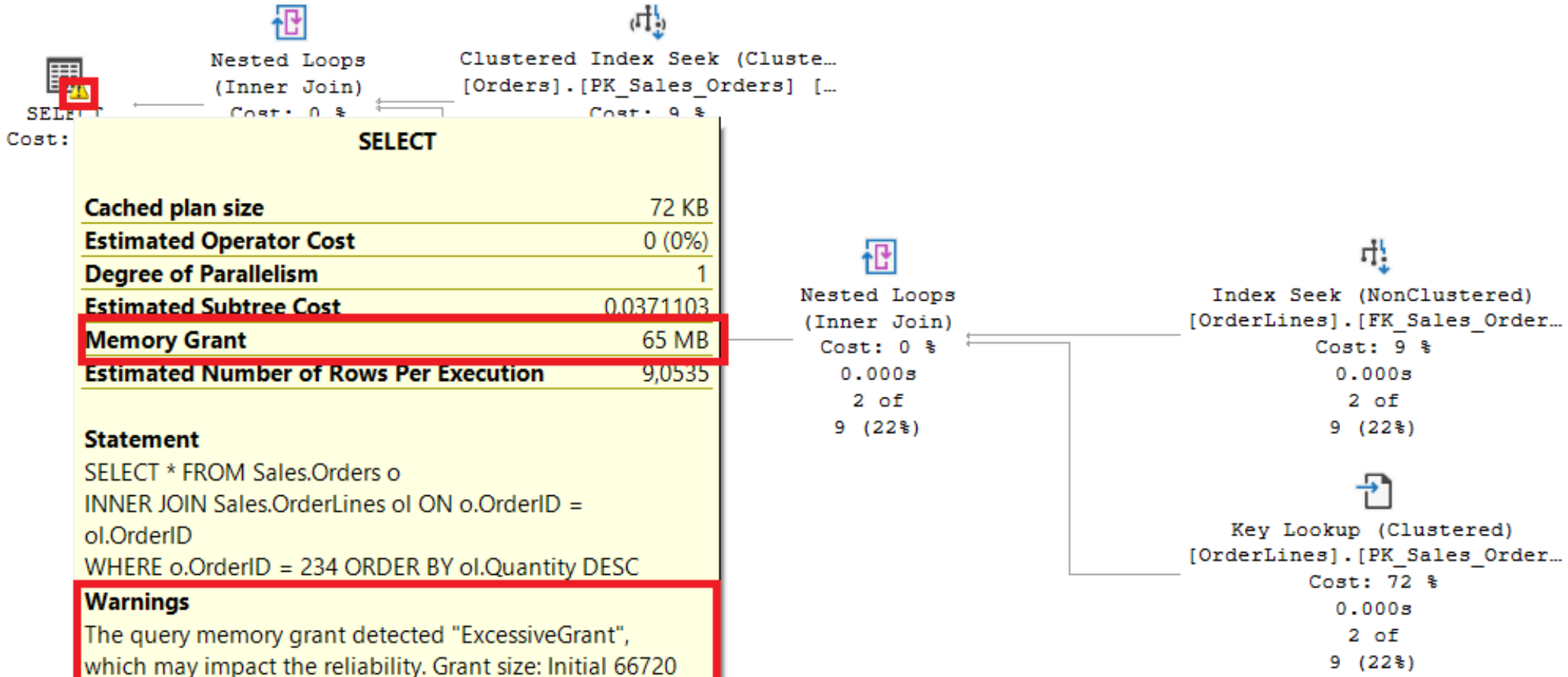


```sql
SELECT * FROM Sales.Orders o
INNER JOIN Sales.OrderLines ol ON o.OrderID = ol.OrderID
WHERE o.OrderID = 234 ORDER BY ol.Quantity DESC
```

100 %

Results | Messages | Execution plan

Query 1: Query cost (relative to the batch): 100%
SELECT * FROM Sales.Orders o INNER JOIN Sales.OrderLines ol ON o.OrderID = ol.OrderID WHERE o.OrderID = 234 ORDER BY



MEMORY GRANT ISSUES

- Significantly inaccurate estimates + sort or hash operators in the execution plan

ÜBERSCHÄTZUNG

- Unnecessarily large memory grants
- Waste of memory
- RESOURCE_SEMAPHORE waits in high-concurrency workloads

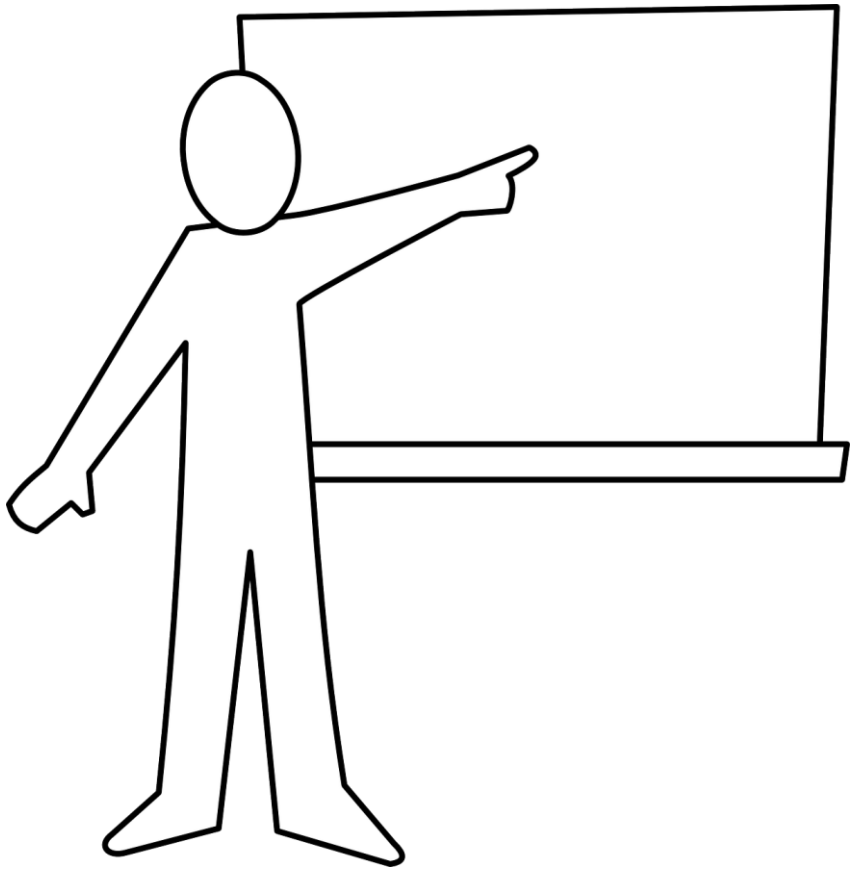
UNTERSCHÄTZUNG

- Insufficient memory grants
- Tempdb spills
- Slow execution

MEMORY GRANT FEEDBACK

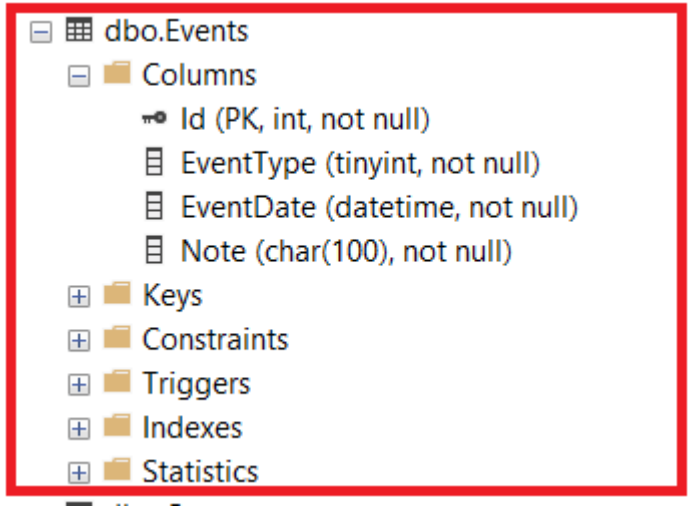
- Adjustment of the storage grant parameter in the execution plan AFTER the creation of the plan (after a few executions)
- The memory is recalculated when:
  - The query is using less than 50% allocated memory
  - Has come to tempdb spills
- Batch mode (SQL Server 2017)
  - Batch mode operators required - aka Columnstore Index
- Row mode (SQL Server 2019)
- Enterprise Edition feature

MEMORY GRANT FEEDBACK



DEMO

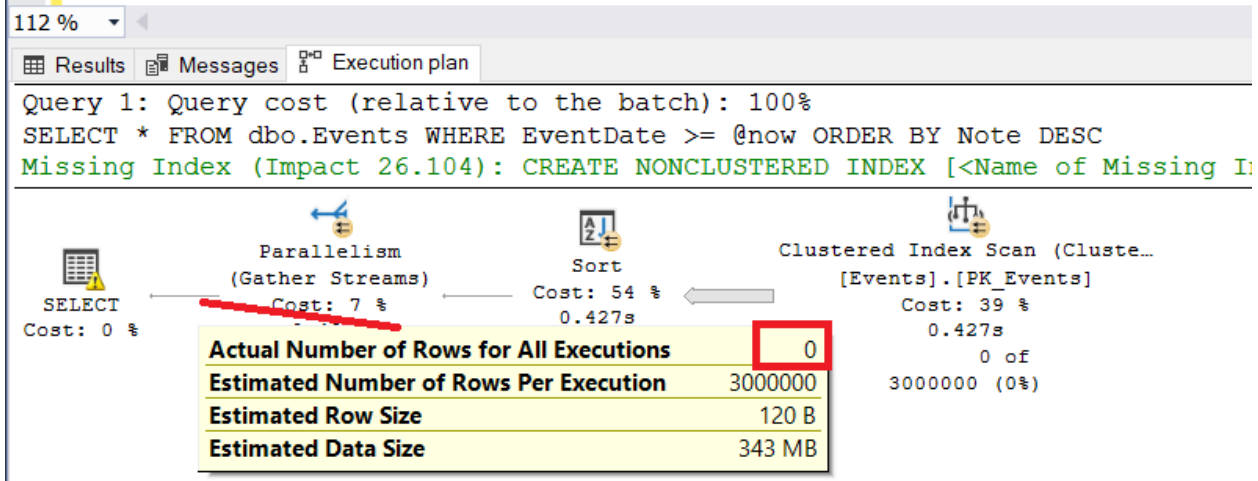
DEMO – SAMPLE TABLE



dbo.Events – **10M rows**

```
CREATE OR ALTER PROCEDURE dbo.GetEvents
@OrderDate DATETIME
AS
BEGIN
        DECLARE @now DATETIME = @OrderDate;
        SELECT * FROM dbo.Events
        WHERE EventDate >= @now
        ORDER BY Note DESC;
END
```

```
EXEC dbo.GetEvents '20180101';
GO
```



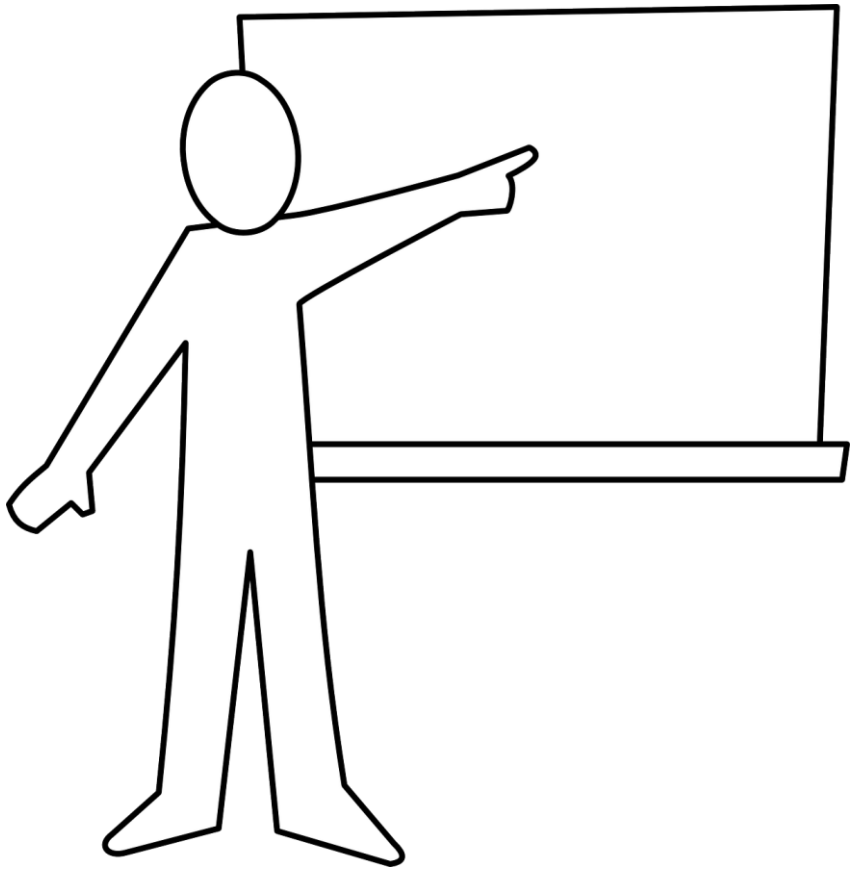
MEMORY GRANT FEEDBACK

```
<MemoryGrantInfo SerialRequiredMemory="152"
                 SerialDesiredMemory="1240"
                 RequiredMemory="1352"
                 DesiredMemory="2440"
                 RequestedMemory="2440"
                 GrantWaitTime="0"
                 GrantedMemory="2440"
                 MaxUsedMemory="1752"
                 MaxQueryMemory="1334464"
                 LastRequestedMemory="670408"
                 IsMemoryGrantFeedbackAdjusted="Yes: Adjusting"
/>
```

ISMEMORYGRANTFEEDBACKADJUSTED

Value	Description
No: First Execution	In the first execution of the query, Memory grant feedback does not adjust memory. We have shown it above as well.
No: Accurate Grant	If there is no spill to disk and the statement uses at least 50% of the granted memory, then memory grant feedback is not triggered.
No: Feedback disabled	If there is a huge variation in the memory grant in subsequent runs, the system disables the memory grant feedback for the query.
Yes: Adjusting	It shows that Memory grant feedback is in place and it may continue for the next runs as well.
Yes: Stable	If the system identifies that granted memory is stable and the memory allocated is the same as of previous execution, you can see this status.

MEMORY GRANT FEEDBACK DISABLED



DEMO

MEMORY GRANT FEEDBACK DISABLED

- If the memory allocation memory values fluctuate, the feature can be automatically disabled werden

```
EXEC dbo.GetEvents '20180101';
EXEC dbo.GetEvents '20160101';
GO 30
```

Displaying 1 Events

	name	timestamp
▶	memory_grant_feedback_loop_disabled	2019-12-06 15:38:03.3566613

Event:memory_grant_feedback_loop_disabled (2019-12-06 15:38:03.3566613)

Details

Field	Value
sql_text	EXEC dbo.GetEvents '20180101'; EXEC dbo.GetEvents '20160101';
total_execution_count	32
total_update_count	31

MGF SETTINGS IN SQL SERVER 2017 (BATCHMODUS)

- Default: `OFF`

- Enable:

```
ALTER DATABASE SCOPED CONFIGURATION SET DISABLE_BATCH_MODE_MEMORY_GRANT_FEEDBACK = OFF;
```

- Disable:
  - auf Datenbank-Ebene:

```
ALTER DATABASE SCOPED CONFIGURATION SET DISABLE_BATCH_MODE_MEMORY_GRANT_FEEDBACK = ON;
```

  - auf Abfrage-Ebene:

```
SELECT * FROM dbo.Events
WHERE EventDate >= @OrderDate ORDER BY Note DESC
OPTION (USE HINT ('DISABLE_BATCH_MODE_MEMORY_GRANT_FEEDBACK'));
```

MGF EINSTELLUNGEN IN SQL SERVER 2019 (BATCHMODUS)

- Default: ON

- Aktivieren:

```
ALTER DATABASE SCOPED CONFIGURATION SET BATCH_MODE_MEMORY_GRANT_FEEDBACK = ON;
```

- Deaktivieren:
  - auf Datenbank-Ebene:

```
ALTER DATABASE SCOPED CONFIGURATION SET BATCH_MODE_MEMORY_GRANT_FEEDBACK = OFF;
```

  - auf Abfrage-Ebene:

```
SELECT * FROM dbo.Events
WHERE EventDate >= @OrderDate ORDER BY Note DESC
OPTION (USE HINT ('DISABLE_BATCH_MODE_MEMORY_GRANT_FEEDBACK'));
```

MGF EINSTELLUNGEN IN SQL SERVER 2019 (ZEILENMODUS)

- Default: ON

- Aktivieren:

```
ALTER DATABASE SCOPED CONFIGURATION SET ROW_MODE_MEMORY_GRANT_FEEDBACK = ON;
```

- Deaktivieren:
  - auf Datenbank-Ebene:

```
ALTER DATABASE SCOPED CONFIGURATION SET ROW_MODE_MEMORY_GRANT_FEEDBACK = OFF;
```

  - auf Abfrage-Ebene:

```
SELECT * FROM dbo.Events
WHERE EventDate >= @OrderDate ORDER BY Note DESC
OPTION (USE HINT ('DISABLE_ROW_MODE_MEMORY_GRANT_FEEDBACK'));
```

CONCLUSION

- A great feature
- when memory is constantly underestimated or overestimated

- In SQL Server 2019, it works in both modes
- affects ALL queries with an operator that requires memory!

- Almost no measurable overhead (*)
  - **(*) in all my tests**

- The feature is automatically deactivated if it looks wrong