Übung Execution Plan lesen und MAXDOP

Ziel: Vertraut machen mit Ausführungsplan und MAXDOP Settings; Auswirkung eines Index

Aufgabenstellung:

1. Verwenden Sie die KU1 Tabelle, die im Kurs erstellt wurde. Wenn Sie diese Übung später ausführen und die KU1 Tabelle nicht mehr vorhanden ist, erstellen Sie eine Tabelle mit Testdaten, z.B.:

```
SELECT  c.CustomerID
              , c.CompanyName
              , c.ContactName
              , c.ContactTitle
              , c.City
              , c.Country
              , o.EmployeeID
              , o.OrderDate
              , o.freight
              , o.shipcity
              , o.shipcountry
              , od.OrderID
              , od.ProductID
              , od.UnitPrice
              , od.Quantity
              , p.ProductName
              , e.LastName
              , e.FirstName
              , e.birthdate
into Test.dbo.KundenUmsatz
FROM   Northwind.dbo.Customers c
              INNER JOIN Northwind.dbo.Orders o ON c.CustomerID = o.CustomerID
              INNER JOIN Northwind.dbo.Employees e ON o.EmployeeID = e.EmployeeID
              INNER JOIN Northwind.dbo.[Order Details] od ON o.orderid = od.orderid
              INNER JOIN Northwind.dbo.Products p ON od.productid = p.productid

INSERT INTO KundenUmsatz
SELECT * FROM KundenUmsatz
GO 9


-- Kopie mit ID Spalte
SELECT * INTO KU1 FROM KundenUmsatz
ALTER TABLE KU1 ADD ID INT IDENTITY
```

2. Wenn Sie schon mit der KU1-Tabelle geübt haben und beispielsweise Indizes oder Partitionen dafür erstellt haben, erstellen Sie eine Kopie davon.

```
select *
into ku4
from ku1
```

3. Schalten Sie die Statistik ein und aktivieren Sie den Actual Execution Plan.

```
SET STATISTICS IO, TIME ON
```

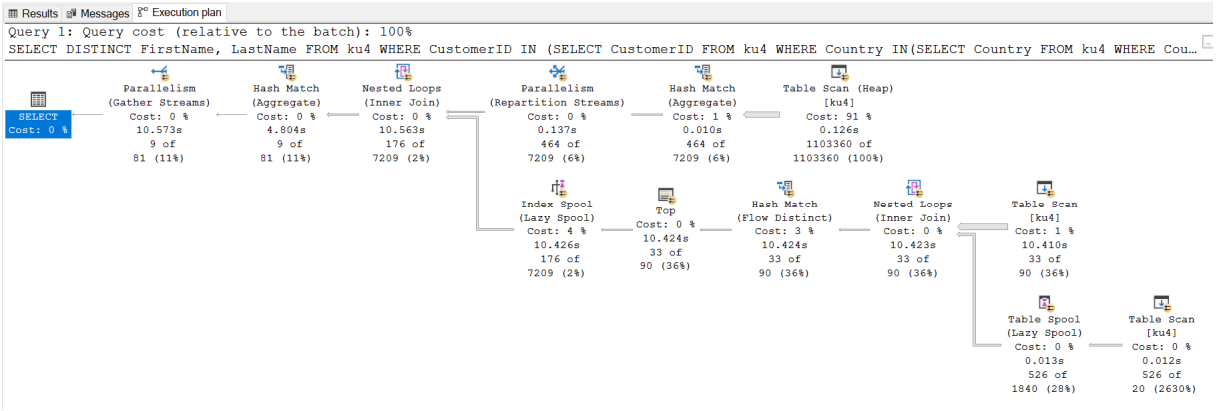
Sie können den Execution Plan über die Menüleiste aktivieren, oder indem Sie STRG+M drücken.

4. Führen Sie eine verschachtelte Abfrage durch, und betrachten Sie die Statistik.

```
SELECT DISTINCT FirstName, LastName
FROM ku4
WHERE CustomerID IN (SELECT CustomerID FROM ku4 WHERE Country
IN(SELECT Country FROM ku4 WHERE Country LIKE '[a-f]%'))
```

Die Abfrage ergibt (für die KU-Tabelle von weiter oben) `-- 3520803 logical reads` und `-- 10625 ms` Abfragezeit, davon `-- 31875 ms CPU time`. Abhängig von der Tabelle, die Sie verwendet haben, Ihrem Rechner und Ihren Settings kann die Abfragezeit stark variieren!

5. Betrachten Sie den Execution Plan für diese Abfrage.

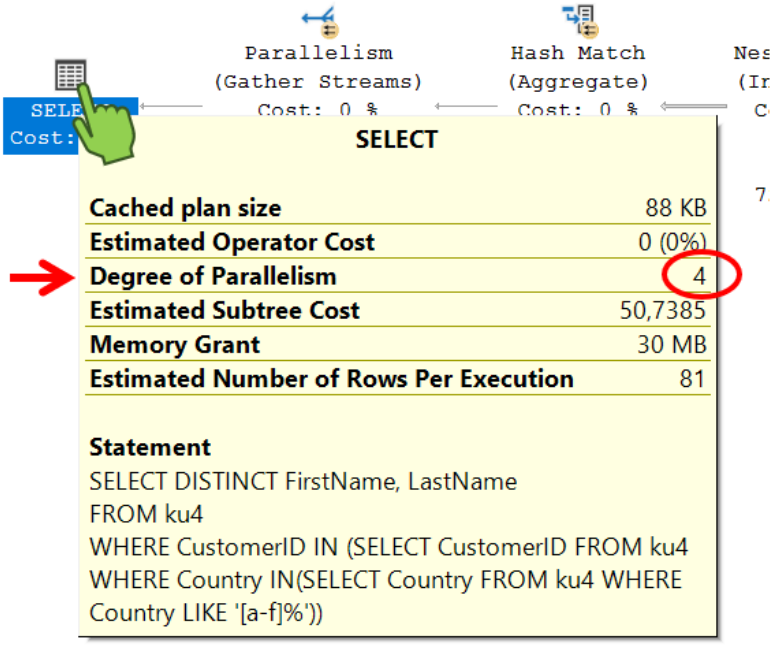


Durch die Unterabfragen (Subqueries) gibt es hier mehrere Ebenen, die wieder zusammengeführt werden müssen.

Da wir keine Indizes in dieser Tabelle haben, müssen überall Table Scans durchgeführt werden.

Die gelb hinterlegten Pfeilchen bedeuten, dass für diese Abfrage Parallelismus verwendet wurde, dass also mehr als ein CPU an der Ausführung dieser Abfrage beteiligt war.

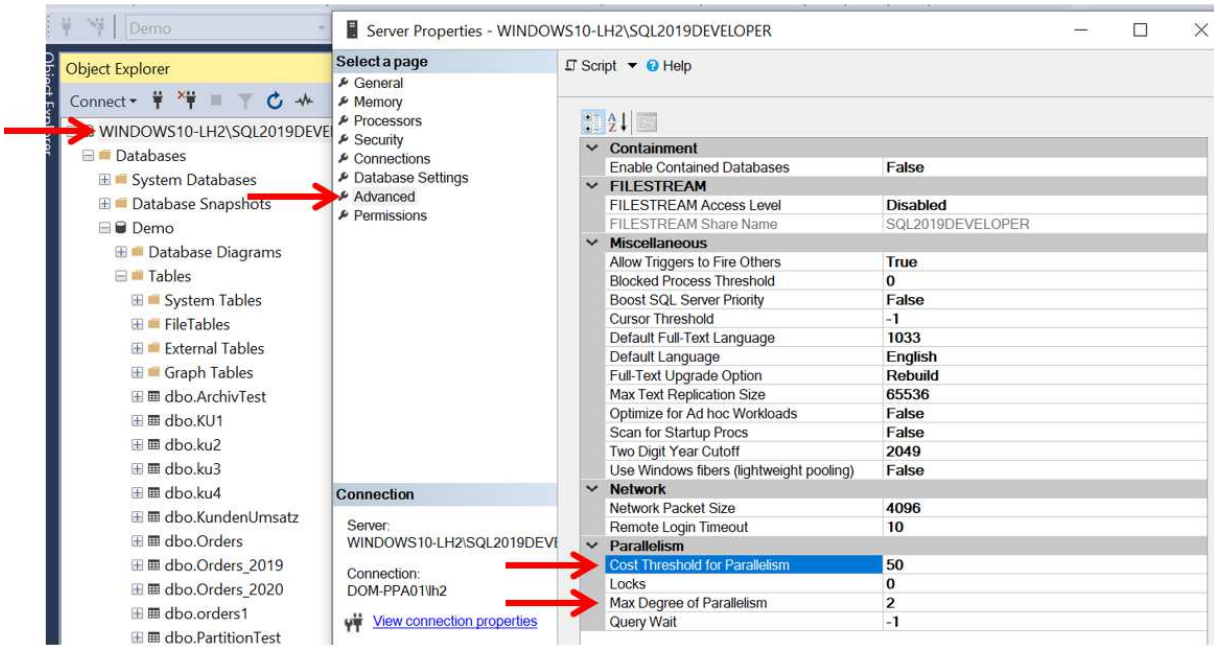
6. Bewegen Sie den Mauszeiger über die SELECT-Abfrage im Execution Plan.



Hier sehen Sie unter „Degree of Parallelism“ wie viele CPUs an der Ausführung der Abfrage beteiligt waren.

Gleich darunter sehen Sie „Estimated Subtree Cost“, die „Kosten“ der Abfrage. Dabei handelt es sich um einen internen Vergleichswert für die Datenbank, der, da es dafür keine Einheit gibt, oft scherzhaft als SQL-Dollar bezeichnet wird.

7. Ändern Sie die Einstellungen für MAXDOP und den Kostenschwellwert.



Im Object Explorer Rechtsklick auf den Server -> Properties -> Advanced finden Sie „Parallelism“. Standardmäßig ist für Max Degree of Parallelism (MAXDOP) der Wert 0 eingetragen. Das bedeutet aber NICHT, dass kein Parallelismus zugelassen ist, sondern nur, dass wir keine Präferenzen eingetragen haben. Es dürfen also ALLE CPUs verwendet werden!

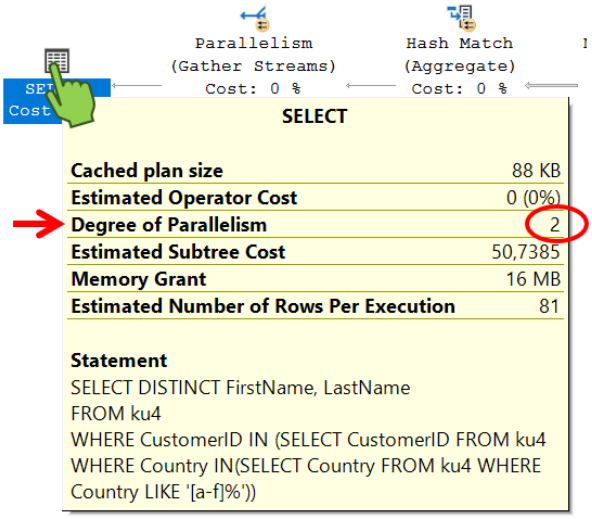
Setzen Sie zum Testen MAXDOP auf 2 (später können Sie das noch mit anderen Einstellungen wiederholen; abhängig von der Anzahl Ihrer CPUs können Sie es auch mit 4, 6 oder 8 ausprobieren).

Beim Kostenschwellwert (Cost Threshold for Parallelism) ist standardmäßig 5 eingetragen. Abhängig von den Kosten Ihrer Abfragen können Sie hier ein sinnvolleres Setting verwenden.

Die Kosten unserer Abfrage waren sehr hoch; wir können testweise den Schwellwert auf 50 setzen.

8. Führen Sie die Abfrage nochmals durch und betrachten Sie wieder den Execution Plan und die Statistik.

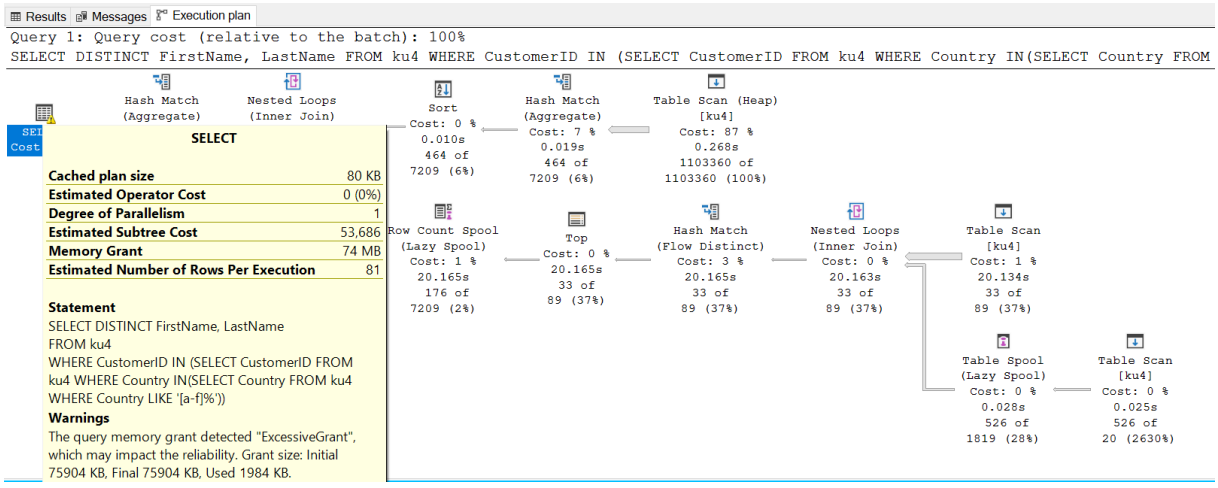
Auf den ersten Blick sieht unser Execution Plan gleich aus; Mauszeiger über dem SELECT zeigt uns die Details:



Da unsere Kosten über 50 (unserem neuen Schwellwert) liegen, durften mehrere CPUs verwendet werden; diesmal allerdings nur zwei.

Bei einem Blick in die Statistik sehen wir auch, dass dadurch unsere Abfragegeschwindigkeit etwas gestiegen ist, die CPU-time allerdings etwas gesunken ist.

9. Setzen Sie MAXDOP auf 4 und den Kostenschwellwert auf 100 und führen Sie die Abfrage noch einmal durch. Betrachten Sie wieder die Details im Execution Plan.



Diesmal hat sich unser Execution Plan geändert. Wir haben zwar 4 CPUs bei MAXDOP gesetzt, aber erst, wenn die geschätzten Kosten 100 übersteigen! Somit durfte für diese Abfrage nur 1 CPU verwendet werden.

Beachten Sie, dass Sie das auch auf den ersten Blick im Execution Plan erkennen können: Die gelb hinterlegten Pfeilchen werden nicht angezeigt, da kein Parallelismus stattgefunden hat.

Die Ausführungszeit ist dadurch natürlich (fast auf das Doppelte) gestiegen.

10. Testen Sie mit mehreren verschiedenen Settings und betrachten Sie jeweils Execution Plan und Statistics.

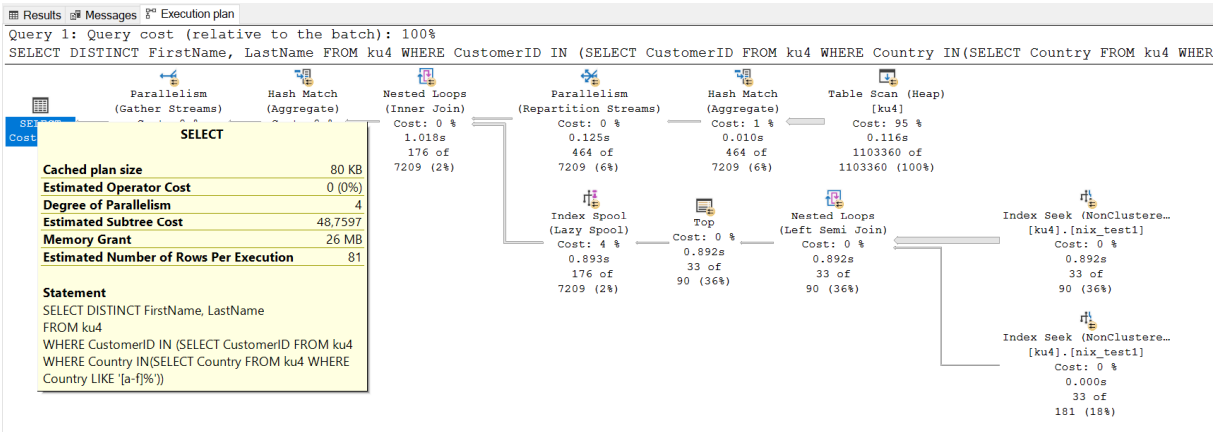
11. Setzen Sie auf die Standardsettings (MAXDOP 0, Schwellwert 5) zurück und vergeben Sie einen Index, z.B.:

```
SET ANSI_PADDING ON

CREATE NONCLUSTERED INDEX nix_ku4 ON [dbo].test1
(
	[Country] ASC
)
INCLUDE([CustomerID]) WITH (SORT_IN_TEMPDB = OFF, DROP_EXISTING =
OFF, ONLINE = OFF) ON [PRIMARY]
```

Sie können einen Index auch über den Object Explorer -> DBName -> Tables -> Table Name, Rechtsklick auf -> Indexes erstellen.

12. Führen Sie die Abfrage noch einmal aus und betrachten Sie wieder Execution Plan und Statistics.



Hier sehen wir, dass durch diesen Index zumindest zweimal ein Index Seek anstatt eines Table Scans gemacht wurde, und dass die geschätzten Kosten in den Details ein wenig gesunken sind.

Vielleicht ist Ihnen aber sofort, auch ohne Blick auf die Statistik, aufgefallen, dass die Abfrage viel schneller geworden ist.

Dafür ist der Index verantwortlich. Statt (durch die Unterabfragen) die Tabelle drei Mal scannen zu müssen, hat sich die Anzahl der logical reads auf `-- 206640 logical reads` reduziert. Die Abfragezeit ist dadurch auf `-- 1057 ms` gesunken (`-- 3091 CPU time`)! Auf Ihrem Rechner werden Sie andere Zeiten beobachten! (Die Zeiten sind auch vom Rechner mitabhängig.)