All about LAG

Kathi Kellenberger

Redgate Software

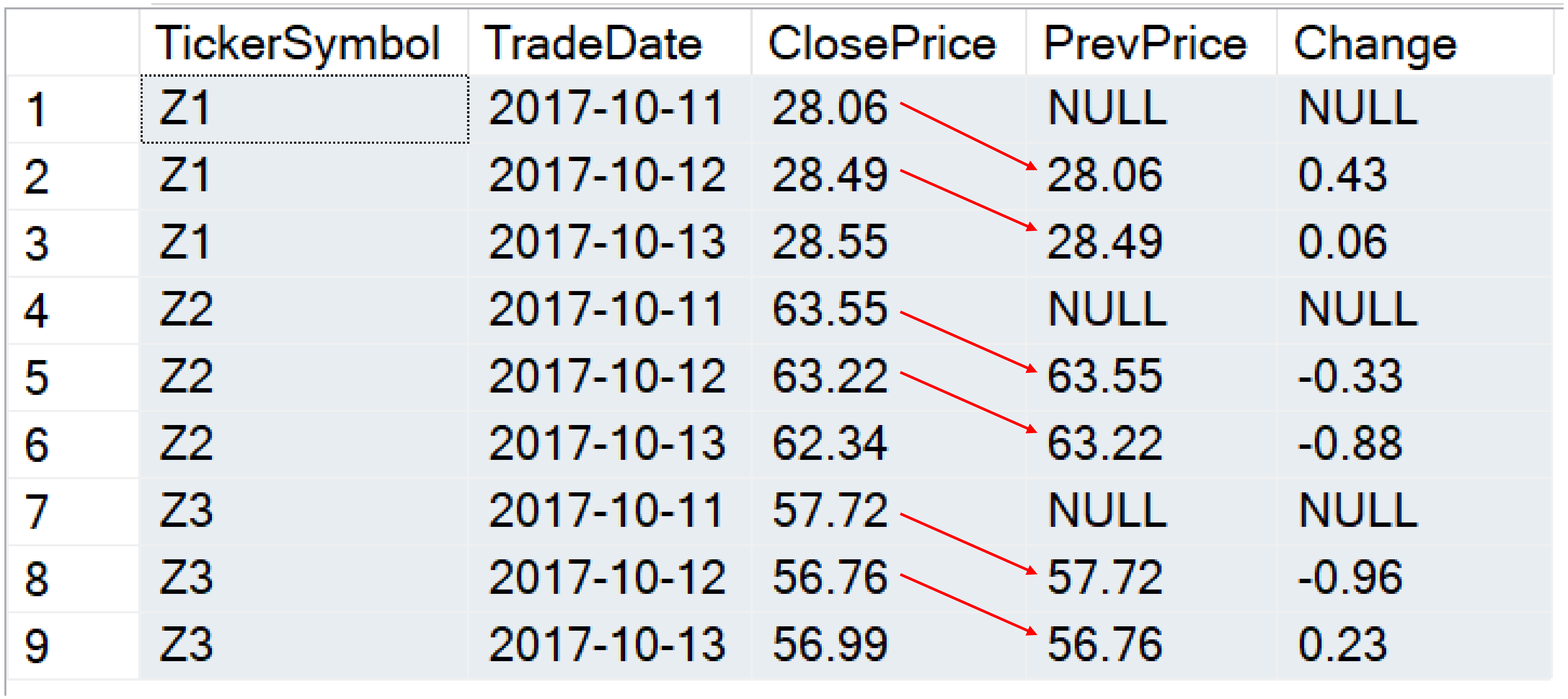
Kathi.Kellenberger@red-gate.com

Kathi Kellenberger

Simple-Talk Editor at Redgate
Instructor at LaunchCode
Lifelong learner and teacher

The stock market problem

Photo by Pixabay from Pexels



	TickerSymbol	TradeDate	ClosePrice	PrevPrice	Change
1	Z1	2017-10-11	28.06	NULL	NULL
2	Z1	2017-10-12	28.49	28.06	0.43
3	Z1	2017-10-13	28.55	28.49	0.06
4	Z2	2017-10-11	63.55	NULL	NULL
5	Z2	2017-10-12	63.22	63.55	-0.33
6	Z2	2017-10-13	62.34	63.22	-0.88
7	Z3	2017-10-11	57.72	NULL	NULL
8	Z3	2017-10-12	56.76	57.72	-0.96
9	Z3	2017-10-13	56.99	56.76	0.23

LAG and LEAD

Window functions new with 2012!

Grab a column from another row, usually the previous (LAG) or next row (LEAD).

```
LAG(expression[,offset] [,default])
OVER([Partition by expression] ORDER BY expression)
```

Divide the rows with PARTITION BY

Line the rows up with ORDER BY, LAG goes back, and LEAD goes forward

Performance is great!!

Many ways to write a query

```sql
SELECT PROD.ProductID, PROD.Name, SOH.OrderDate,
    DATEDIFF(DAY, LAG(SOH.OrderDate)
    OVER(PARTITION BY PROD.ProductID
    ORDER BY SOH.OrderDate),SOH.OrderDate) AS DaysBetweenOrders
FROM Production.Product AS PROD
JOIN Sales.SalesOrderDetail AS SOD
ON SOD.ProductID = PROD.ProductID
JOIN Sales.SalesOrderHeader AS SOH
ON SOH.SalesOrderID = SOD.SalesOrderID
GROUP BY PROD.ProductID, PROD.Name, SOH.OrderDate
ORDER BY PROD.ProductID, SOH.OrderDate;
```

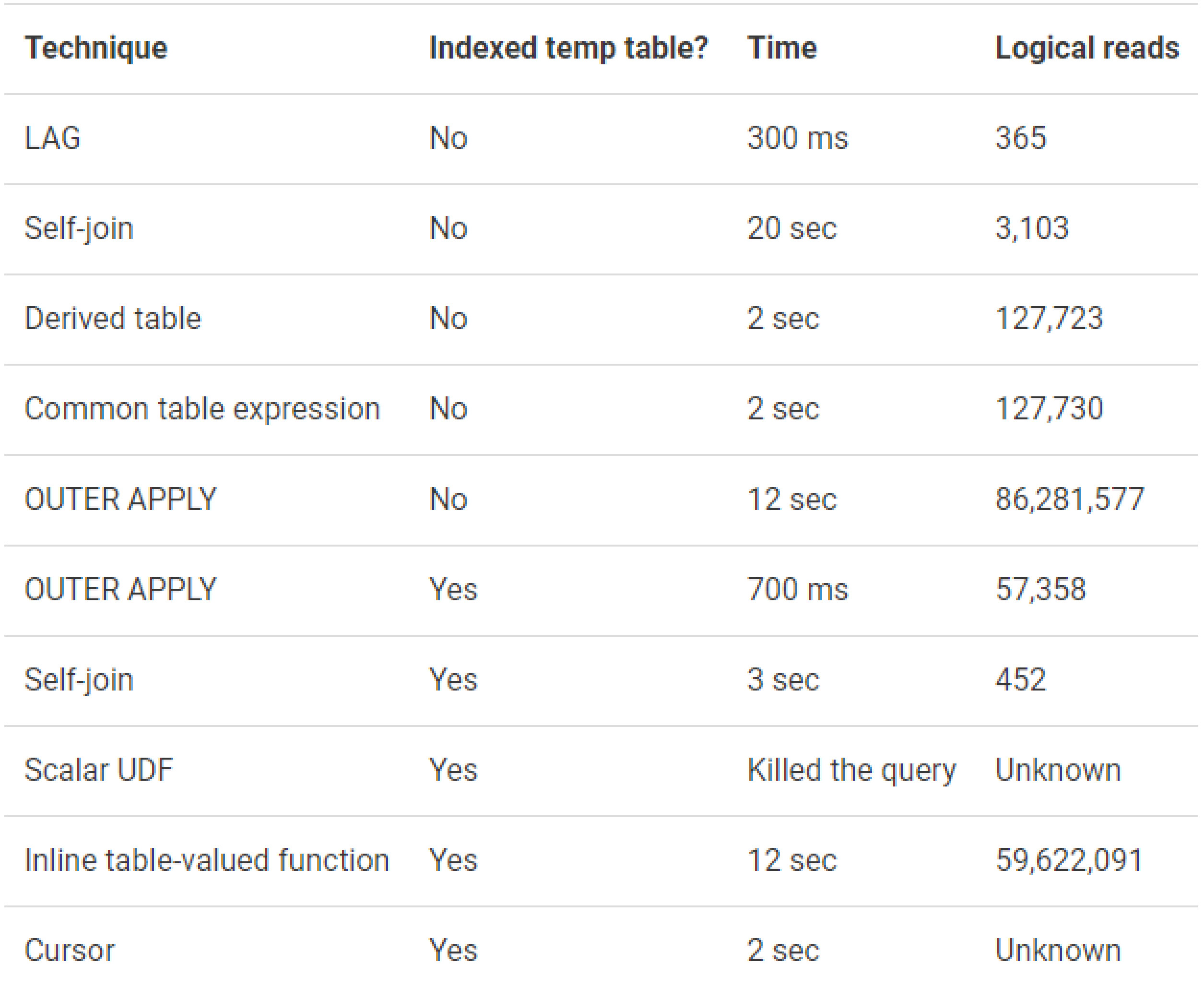
```sql
--Inline table-valued function
GO
CREATE OR ALTER FUNCTION dbo.ITVF_GetPrevDate
(
    @ProductID INT, @OrderDate DATETIME
)
RETURNS TABLE
AS
RETURN
(

    SELECT MAX(SOH.OrderDate) AS PrevOrderDate
    FROM Sales.SalesOrderHeader AS SOH
    JOIN Sales.SalesOrderDetail AS SOD
    ON SOD.SalesOrderID = SOH.SalesOrderID
    WHERE SOD.ProductID = @ProductID
        AND SOH.OrderDate < @OrderDate
)
GO
SELECT PL.ProductID, PL.Name, PL.OrderDate,
   DATEDIFF(DAY,IGPD.PrevOrderDate,PL.OrderDate) AS DaysBetweenOrders
FROM #ProductList AS PL
OUTER APPLY [dbo].[ITVF_GetPrevDate] (PL.ProductID,PL.OrderDate) IGPD
ORDER BY PL.ProductID, PL.OrderDate;
```

```sql
--Self-join
SELECT PROD.ProductID, PROD.Name, SOH.OrderDate,
   DATEDIFF(DAY, MAX(SOH2.OrderDate), SOH.OrderDate) AS DaysBetweenOrders
FROM Production.Product AS PROD
JOIN Sales.SalesOrderDetail AS SOD
ON SOD.ProductID = PROD.ProductID
JOIN Sales.SalesOrderHeader AS SOH
ON SOH.SalesOrderID = SOD.SalesOrderID
JOIN Sales.SalesOrderDetail AS SOD2
ON SOD2.ProductID = PROD.ProductID
JOIN Sales.SalesOrderHeader AS SOH2
ON SOH2.SalesOrderID = SOD2.SalesOrderID
WHERE SOH2.OrderDate < SOH.OrderDate
GROUP BY PROD.ProductID
   , PROD.Name
   , SOH.OrderDate
   , SOD2.ProductID
ORDER BY PROD.ProductID, SOH.OrderDate;
```

```sql
--OUTER APPLY
SELECT PROD.ProductID, PROD.Name, SOH.OrderDate,
   DATEDIFF(DAY, S2.PrevOrderDate, SOH.OrderDate) AS DaysBetweenOrders
FROM Production.Product AS PROD
JOIN Sales.SalesOrderDetail AS SOD
ON SOD.ProductID = PROD.ProductID
JOIN Sales.SalesOrderHeader AS SOH
ON SOH.SalesOrderID = SOD.SalesOrderID
OUTER APPLY (
    SELECT MAX(SOH2.OrderDate) AS PrevOrderDate
    FROM Sales.SalesOrderDetail AS SOD2
    JOIN Sales.SalesOrderHeader AS SOH2
    ON SOH2.SalesOrderID = SOD2.SalesOrderID
    WHERE SOD2.ProductID = PROD.ProductID
        AND SOH2.OrderDate < SOH.OrderDate) S2
GROUP BY  DATEDIFF(DAY, S2.PrevOrderDate, SOH.OrderDate)
        , PROD.ProductID
        , PROD.Name
        , SOH.OrderDate
ORDER BY PROD.ProductID, SOH.OrderDate;
```

Performance comparisons

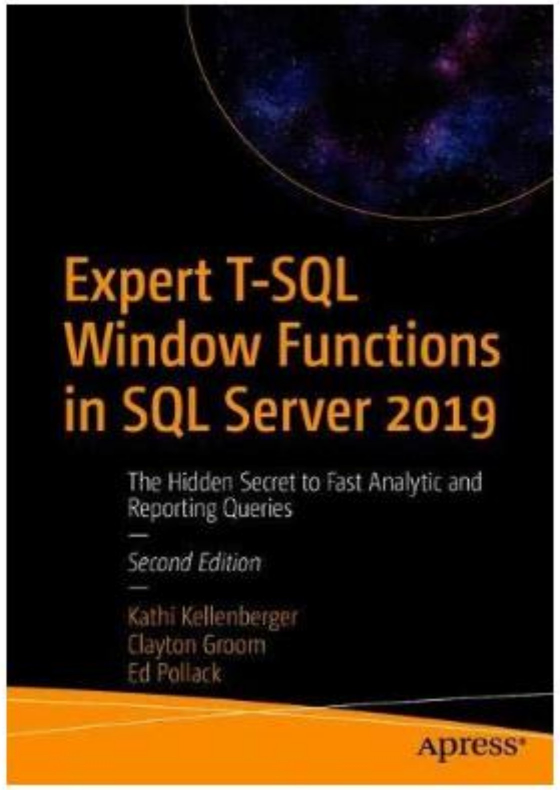


Technique	Indexed temp table?	Time	Logical reads
LAG	No	300 ms	365
Self-join	No	20 sec	3,103
Derived table	No	2 sec	127,723
Common table expression	No	2 sec	127,730
OUTER APPLY	No	12 sec	86,281,577
OUTER APPLY	Yes	700 ms	57,358
Self-join	Yes	3 sec	452
Scalar UDF	Yes	Killed the query	Unknown
Inline table-valued function	Yes	12 sec	59,622,091
Cursor	Yes	2 sec	Unknown

https://rd.gt/3GMGss0

Slides and Code!

https://github.com/KathiKellenberger/AllAboutLag



Expert T-SQL Window Functions in SQL Server 2019

The Hidden Secret to Fast Analytic and Reporting Queries

Second Edition

Kathi Kellenberger
Clayton Groom
Ed Pollack

Apress®