2015 OSIsoft TechCon

Optimizing SQL Queries for Performance using PI OLEDB Enterprise

Table of Contents

Contents

Optimizing SQL Queries for Performance using PI OLEDB Enterprise

Overview

In this lab, you will learn a few techniques for creating truly performing SQL using PI OLEDB Enterprise. This Lab will also point out what it takes to verify if queries that are automatically generated by 3rd party tools have issues.

Part 1 Optimization walk through

Part 2 Investigating performance issues by a query generated by 3rd party Microsoft product

Not all of the features are completely explained in this tutorial. Please be sure to stay within the workbook instructions for the best learning experience.

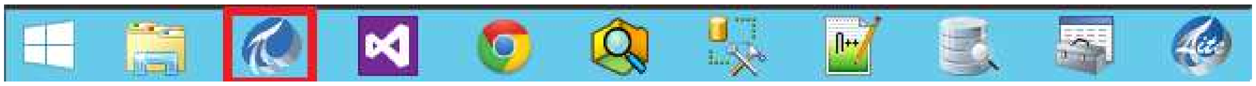
Required Software

- PI OLEDB Enterprise 2012 1.3.15
- MS SQL Server 2012

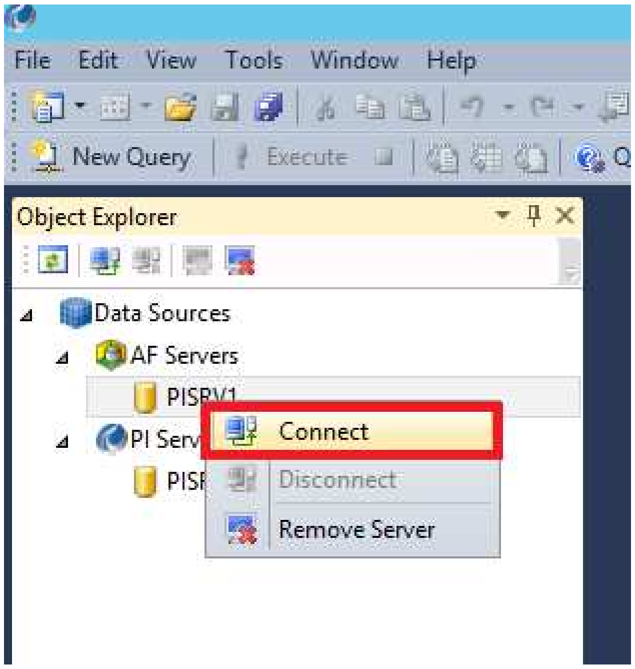
Part 1 Optimization walk through

This section will guide you through some *PI OLEDB Enterprise* query engine details and demonstrate how to write a query in a way it is executed optimally. We will use *PI SQL Commander* to run the queries and inspect the results and the execution times.

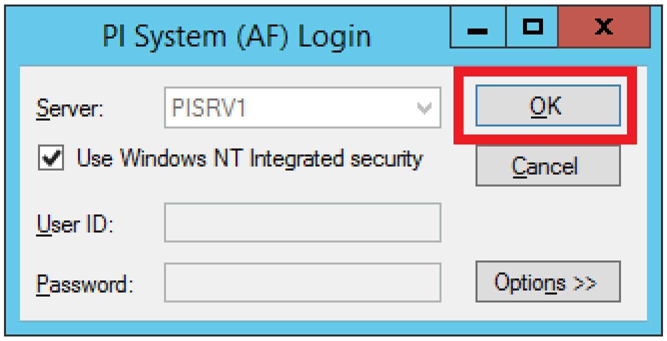
1. Launch *PI SQL Commander* from the windows task bar.



2. Right click on the AF Server *PISVR1* in the *Object Explorer* and select *Connect*.



3. Connect to the AF Server by clicking *OK*.



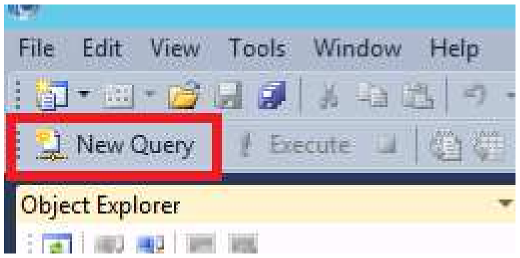
Server: PISRV1

☑ Use Windows NT Integrated security

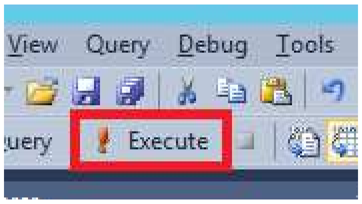
User ID:

Password:

4. You should see a green symbol by the connected server and it should contain four catalogs: *Configuration*, *MDB*, *NuGreen* and *System*. We will now execute several queries to see the differences in the execution time. Click *New Query* toolbar button to open a new query window.

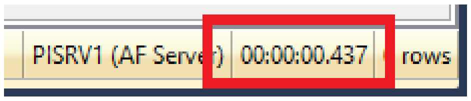


5. Type the following query into the editor window and execute it by clicking *Execute* toolbar button or by pressing *Ctrl+E*.



```
SELECT * FROM NuGreen.Asset.ElementTemplate
```

6. See the execution time in the bottom right corner of the result window.



7. Execute the query again and observe if the execution time changes.

8. Execute the following query twice and observe the execution times.

```
SELECT * FROM NuGreen.Asset.Element
```

Explanation

While the *Element* query took more or less the same, *ElementTemplate* second execution was faster since the table is cached on the client side. The following tables get cached: *Asset.ElementTemplate, Asset.ElementTemplateAttribute, Asset.ElementTemplateCategory, Asset.ElementTemplateAttributeCategory, EventFrame.EventFrameTemplate, EventFrame.EventFrameTemplateAttribute, EventFrame.EventFrameTemplateCategory, EventFrame.EventFrameTemplateAttributeCategory*.

9. Execute the following two queries and compare the execution times.

```
SELECT * FROM NuGreen.EventFrame.EventFrame
-----------------------------------------------------------------------------------------
SELECT Name FROM NuGreen.EventFrame.EventFrame
```

Note: You may want to execute each query multiple times and calculate the average execution time in order to get more appropriate result and minimize the influence of caching. This also applies to all following queries.

Explanation

The amount of data transferred to the client in the first case is greater than in the second one. We often see applications request data they do not need/use. Request always only those columns you need. It may be quite straightforward in the example above, but the same rule applies for subqueries. Try executing the following queries a few times.

```
SELECT eh.Path, eh.Name
FROM NuGreen.Asset.ElementHierarchy eh
JOIN (SELECT * FROM NuGreen.Asset.Element) e
   ON eh.ElementID = e.ID
-----------------------------------------------------------------------------------------
SELECT eh.Path, eh.Name
FROM NuGreen.Asset.ElementHierarchy eh
JOIN (SELECT ID FROM NuGreen.Asset.Element) e
   ON eh.ElementID = e.ID
```

The time difference is really small but in average the second query is a bit faster. The differences can be significant when we would not work on a local environment (everything is on one PC). It would also grow with the size of the database.

10. Execute the following two queries and compare the execution times.

```
SELECT Path, Name
FROM NuGreen.Asset.ElementHierarchy eh
WHERE EXISTS
(
   SELECT 1
   FROM NuGreen.Asset.ElementAttribute
   WHERE ElementID = eh.ElementID
)
-----------------------------------------------------------------------------------------
SELECT DISTINCT eh.Path, eh.Name
FROM NuGreen.Asset.ElementHierarchy eh
JOIN NuGreen.Asset.ElementAttribute ea
   ON ea.ElementID = eh.ElementID
```

Explanation

The correlated subquery (SELECT 1…) in the first query is executed for every row from *Asset.ElementHierarchy* table which makes it really slow. Try not to use correlated subqueries (***correlated queries*** *are queries, which use values from the outer query*).

11. Execute the following two queries and compare the execution times.

```
SELECT e.Name Element
FROM NuGreen.Asset.Element e
JOIN NuGreen.Asset.ElementTemplate et
  ON et.ID = e.ElementTemplateID
JOIN NuGreen.Asset.ElementAttribute ea
  ON ea.ElementID = e.ID
JOIN NuGreen.Data.Snapshot s
  ON s.ElementAttributeID = ea.ID
WHERE et.Name = N'Boiler' AND ea.Name = N'Manufacturer' AND s.ValueStr = N'NATCOM'
-------------------------------------------------------------------------------------------
SELECT e.Name Element
FROM NuGreen.Asset.ElementTemplate et
JOIN NuGreen.Asset.ElementTemplateAttribute eta
  ON eta.ElementTemplateID = et.ID
JOIN NuGreen.Data.Snapshot s
  ON s.ElementTemplateAttributeID = eta.ID
JOIN NuGreen.Asset.ElementAttribute ea
  ON ea.ID = s.ElementAttributeID
JOIN NuGreen.Asset.Element e
  ON e.ID = ea.ElementID
WHERE et.Name = N'Boiler' AND eta.Name = N'Manufacturer' AND s.ValueStr = N'NATCOM'
```

Explanation

Starting with *PI OLEDB Enterprise 2010 R3*, the efficient "by value" searches for non-data reference based template attributes (static attributes inherited from an element template) are supported. This applies to *Data.Snapshot* table, user-created transpose functions, and function tables if access via *Asset.ElementTemplateAttribute* table.
The first query uses access to snapshot data via *Asset.ElementAttribute* table which is not optimized.

12. Execute the following two queries and compare the execution times.

```
SELECT ea.Name, a.Time, a.Value
FROM NuGreen.Asset.ElementTemplate et
JOIN NuGreen.Asset.ElementTemplateAttribute eta
  ON eta.ElementTemplateID = et.ID
JOIN NuGreen.Asset.ElementAttribute ea
  ON ea.ElementTemplateAttributeID = eta.ID
JOIN NuGreen.Data.Archive a
  ON a.ElementAttributeID = ea.ID
WHERE et.Name = N'Heater' AND (a.Time > N'1-Feb-2015' AND a.Time < N'10-Feb-
2015' OR a.Time > N'12-Feb-2015' AND a.Time < N'22-Feb-2015')
-----------------------------------------------------------------------------------------
SELECT ea.Name, a.Time, a.Value
FROM NuGreen.Asset.ElementTemplate et
JOIN NuGreen.Asset.ElementTemplateAttribute eta
  ON eta.ElementTemplateID = et.ID
JOIN NuGreen.Asset.ElementAttribute ea
  ON ea.ElementTemplateAttributeID = eta.ID
JOIN NuGreen.Data.Archive a
  ON a.ElementAttributeID = ea.ID
WHERE et.Name = N'Heater' AND a.Time > N'1-Feb-2015' AND a.Time < N'10-Feb-2015'
UNION ALL
SELECT ea.Name, a.Time, a.Value
FROM NuGreen.Asset.ElementTemplate et
JOIN NuGreen.Asset.ElementTemplateAttribute eta
  ON eta.ElementTemplateID = et.ID
JOIN NuGreen.Asset.ElementAttribute ea
  ON ea.ElementTemplateAttributeID = eta.ID
JOIN NuGreen.Data.Archive a
  ON a.ElementAttributeID = ea.ID
WHERE et.Name = N'Heater' AND a.Time > N'12-Feb-2015' AND a.Time < N'22-Feb-2015'
```

Explanation

The *UNION* and *UNION ALL* queries are executed in parallel. The *WHERE* condition can be split into two where the first takes the first part of the OR clause and the second one the second one. The queries are then executed in parallel and the results are concatenated at the end.

13. Execute the following two queries and compare the execution times.

```
SELECT e.Name, ea.Name, s.Value
FROM NuGreen.Asset.Element e
JOIN NuGreen.Asset.ElementAttribute ea
   ON e.ID = ea.ElementID
JOIN NuGreen.Data.Snapshot s
   ON ea.ID = s.ElementAttributeID
WHERE s.Value = N'NATCOM' and ea.Name LIKE N'Manu%' AND e.Name LIKE N'B%'
OPTION (ALLOW EXPENSIVE, IGNORE ERRORS)
----------------------------------------------------------------------------------------
SELECT e.Name, ea.Name, s.Value
FROM NuGreen.Asset.Element e
JOIN NuGreen.Asset.ElementAttribute ea
   ON e.ID = ea.ElementID
JOIN NuGreen.Data.Snapshot s
   ON ea.ID = s.ElementAttributeID
WHERE s.Value = N'NATCOM' and ea.Name LIKE N'Manu%' AND e.Name LIKE N'B%'
OPTION (FORCE ORDER, ALLOW EXPENSIVE, IGNORE ERRORS)
```

Explanation

Based on the restrictions defined by the *WHERE* clause the query engine makes a decision on how to execute the join. It first requests the data from one table and then from the other one using the data from the first table as an additional restriction. The query engine tries to estimate which of the restriction is more restrictive (will lead to less rows in the result) and request the data from such table first. It made here a wrong decision and requested data from *Snapshot* and *Element Attribute* table first. You may help the query engine with the decision and use *FORCE ORDER* option which forces the same order as the order of the tables in join.
The *OPTION (FORCE ORDER)* clause must be used with caution. If used inappropriately, the execution plan might be less optimal when compared to the one that the query engine would generate otherwise.

14. Execute the following two queries and compare the execution times.

```
SELECT Name, Path
FROM NuGreen.Asset.ElementHierarchy
WHERE Path LIKE N'%NuGreen\Houston\%'
----------------------------------------------------------------------------------------
SELECT Name, Path
FROM NuGreen.Asset.ElementHierarchy
WHERE Path LIKE N'\NuGreen\Houston\%'
```

Explanation

If *%* is used at the beginning of the search pattern, the query engine cannot index the values in the search column and therefore needs to inspect all values. Try to avoid *LIKE* conditions with *%* at the beginning.

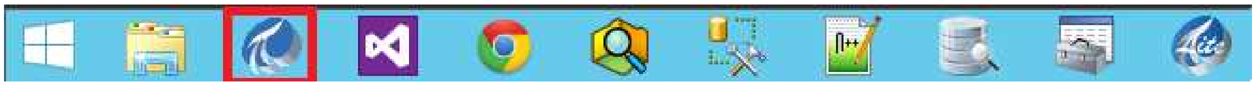
Tip

PI OLEDB Enterprise optimization concepts are described in detail in ***PI OLEDB Enterprise SQL Optimization*** white paper that can be downloaded from OSIsoft Tech Support web site (https://techsupport.osisoft.com/). You can also download it using the direct link.

Part 2 Linked Server

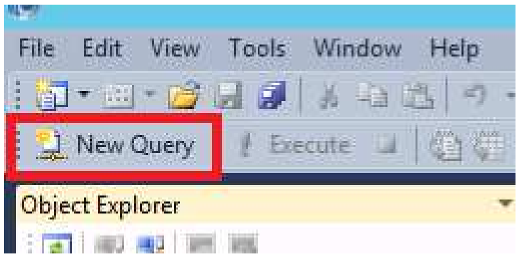
In this part we will execute a query in a Microsoft product. The sample query we will use was developed in SQL Commander. We will take a look at the execution time in PI SQL Commander and MS SQL Server. We will investigate the execution time difference in both products. At the end we will take a look at possible solutions to improve the performance of this query.

1. Launch or navigate to *PI SQL Commander*



2. Connect to the *PISRV1* AF Server (the same we used in the first part of this learning lab)

3. Select the *New Query* toolbar button to open a new query window in *PI SQL Commander*

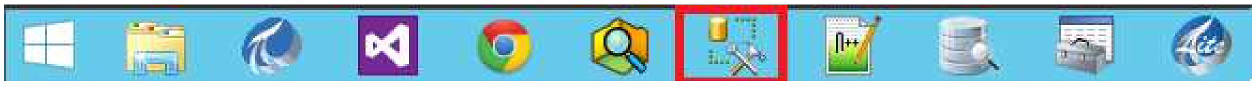


4. Copy the query below to the opened query editor in *PI SQL Commander*

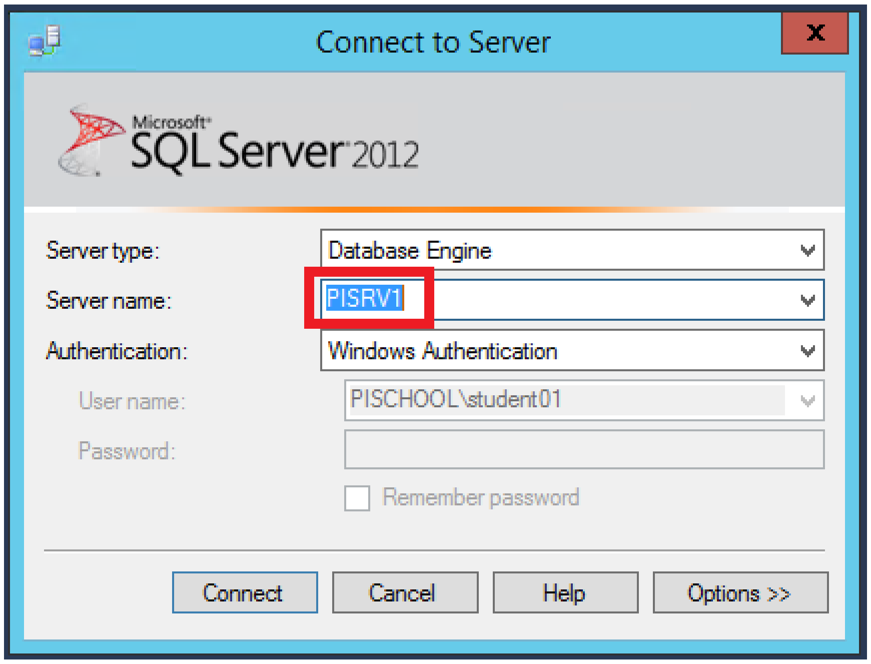
```
SELECT ef2.Name EF_Name, e.Name E_Name, ea.Name EA_Name, v.Value
FROM [NuGreen].[EventFrame].[EventFrame] ef2
INNER JOIN [NuGreen].[Asset].[Element] e
 ON e.ID = ef2.PrimaryReferencedElementID
INNER JOIN [NuGreen].[Asset].[ElementAttribute] ea
 ON e.ID = ea.ElementID
INNER JOIN [NuGreen].[Data].[ft_InterpolateDiscrete] v
 ON ea.ID = v.ElementAttributeID
WHERE ef2.PrimaryParentID IN
    (SELECT TOP 1 ef1.ID
     FROM [NuGreen].[EventFrame].[EventFrame] ef1
     WHERE IsRoot = True
     AND StartTime > '1-Feb-2015'
     ORDER BY StartTime ASC)
AND v.Time = ef2.endtime
AND ef2.Name Like '%A' -- we just want the 'xxxx_A' child Event Frames
```

5. Execute the query and check the execution time

6. Launch *SQL Server Management Studio (SSMS)* from the windows bar



7. Connect to the default selected server (*PISRV1*)



Connect to Server

Microsoft SQL Server 2012

Server type: Database Engine

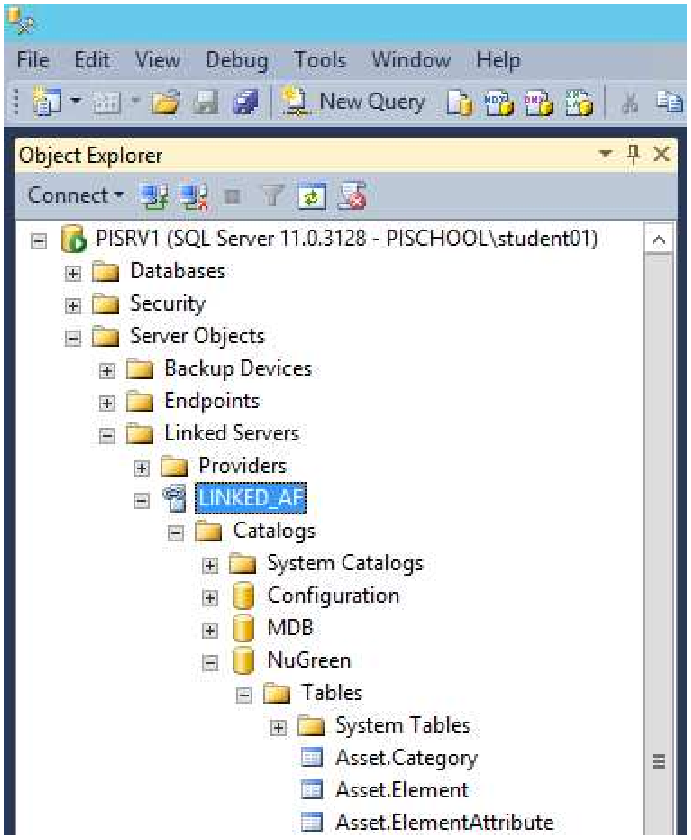
Server name: PISRV1

Authentication: Windows Authentication

User name: PISCHOOL\student01

Password:

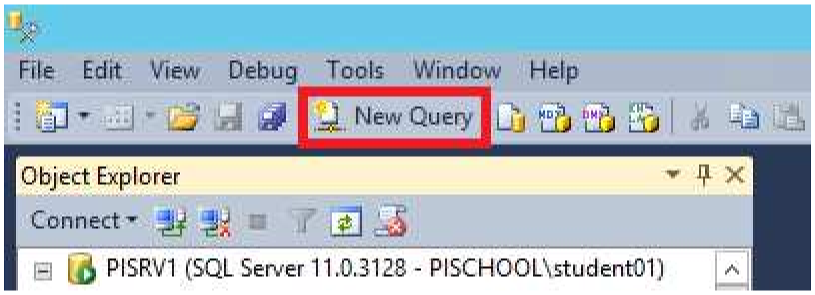
8. Navigate to *Object Explorer* in SSMS and expand the *Tables* node in *PISRV1->Server Objects->Linked Servers->LinkedAF->Catalogs->NuGreen* node



PISRV1 (SQL Server 11.0.3128 - PISCHOOL\student01)
Databases
Security
Server Objects
Backup Devices
Endpoints
Linked Servers
Providers
LINKED_AF
Catalogs
System Catalogs
Configuration
MDB
NuGreen
Tables
System Tables
Asset.Category
Asset.Element
Asset.ElementAttribute

You may notice that the PI System is already exposed to a SQL Server. This configuration enables SQL Server to execute commands against PIOLEDB Enterprise.

9. Select the *New Query* toolbar button to open a new query window in *SSMS*



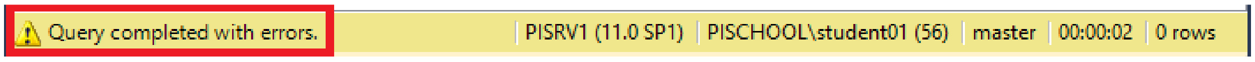
10. Take a look at the query below and check what are the differences with the query we used in *PI SQL Commander*

```
SELECT ef2.Name EF_Name, e.Name E_Name, ea.Name EA_Name, v.Value
FROM [LINKED_AF].[NuGreen].[EventFrame].[EventFrame] ef2
INNER JOIN [LINKED_AF].[NuGreen].[Asset].[Element] e
 ON e.ID = ef2.PrimaryReferencedElementID
INNER JOIN [LINKED_AF].[NuGreen].[Asset].[ElementAttribute] ea
 ON e.ID = ea.ElementID
INNER JOIN [LINKED_AF].[NuGreen].[Data].[ft_InterpolateDiscrete] v
 ON ea.ID = v.ElementAttributeID
WHERE ef2.PrimaryParentID IN
(SELECT TOP 1 ef1.ID
 FROM [LINKED_AF].[NuGreen].[EventFrame].[EventFrame] ef1
 WHERE IsRoot = 1
 AND StartTime > '1-Feb-2015'
 ORDER BY StartTime ASC)
AND v.Time = ef2.endtime
AND ef2.Name Like '%A' -- we just want the 'xxxx_A' child Event Frames
```

To make it work in SQL Server we had to avoid any PI specific time literals and PI specific functions (we could not use condition like "*StartTime > 'y'*"), which are not known to SQL Server. We also need to make sure that SQL Server knows where the database is coming from. The changes made are:

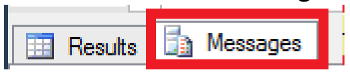
a. the linked server name needs to precede the AF database name i.e. **[NuGreen].[Asset].[ElementAttribute] => [LINKED_AF]. [NuGreen].[Asset].[ElementAttribute]**
b. the condition "**IsRoot = True**" needed to be changed to "**IsRoot = 1**"

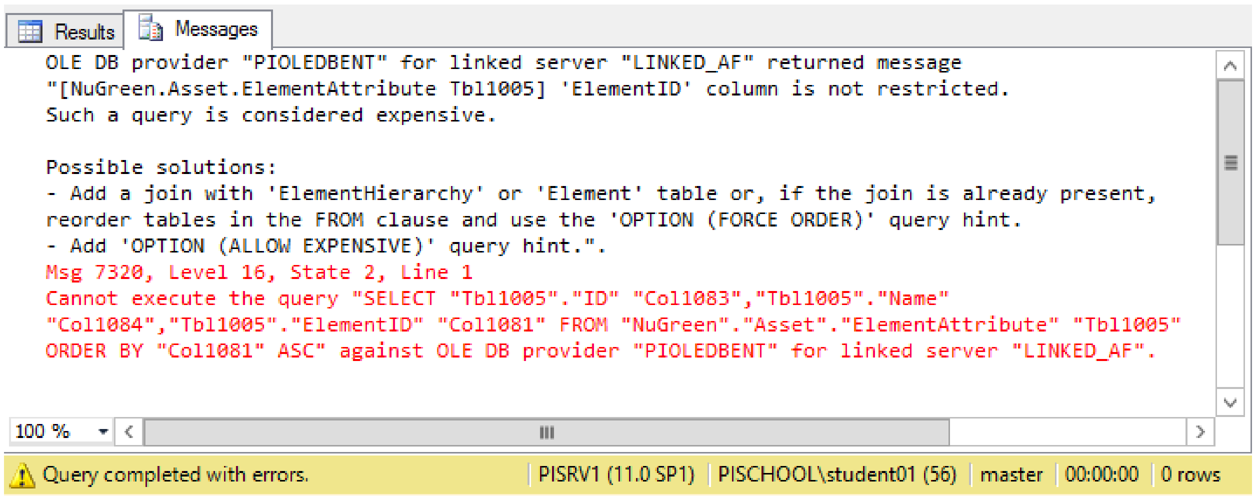
11. Now copy the above query and paste it to previously opened query editor in SSMS. Execute the query by pressing **F5**.


Query completed with errors. | PISRV1 (11.0 SP1) | PISCHOOL\student01 (56) | master | 00:00:02 | 0 rows

The same query that we used in PI SQL Commander is now completing with errors. It can mean only one thing: SQL Server is doing something different.

12. Take a look at the message returned by the execution of the query in the message tab:





```
OLE DB provider "PIOLEDBENT" for linked server "LINKED_AF" returned message
"[NuGreen.Asset.ElementAttribute Tbl1005] 'ElementID' column is not restricted.
Such a query is considered expensive.

Possible solutions:
- Add a join with 'ElementHierarchy' or 'Element' table or, if the join is already present,
reorder tables in the FROM clause and use the 'OPTION (FORCE ORDER)' query hint.
- Add 'OPTION (ALLOW EXPENSIVE)' query hint.".
Msg 7320, Level 16, State 2, Line 1
Cannot execute the query "SELECT "Tbl1005"."ID" "Col1083","Tbl1005"."Name"
"Col1084","Tbl1005"."ElementID" "Col1081" FROM "NuGreen"."Asset"."ElementAttribute" "Tbl1005"
ORDER BY "Col1081" ASC" against OLE DB provider "PIOLEDBENT" for linked server "LINKED_AF".
```

Query completed with errors. | PISRV1 (11.0 SP1) | PISCHOOL\student01 (56) | master | 00:00:00 | 0 rows

First we notice that the error was returned by PIOLEDB Enterprise itself. The second thing we discover is that the *ElementAttribute* table was not restricted. In the message marked red we can look at further details:

```
SELECT "Tbl1005"."ID" "Col1120","Tbl1005"."Name" "Col1121","Tbl1005"."ElementID"
"Col1118"
FROM "NuGreen"."Asset"."ElementAttribute" "Tbl1005"
ORDER BY "Col1118" ASC
```

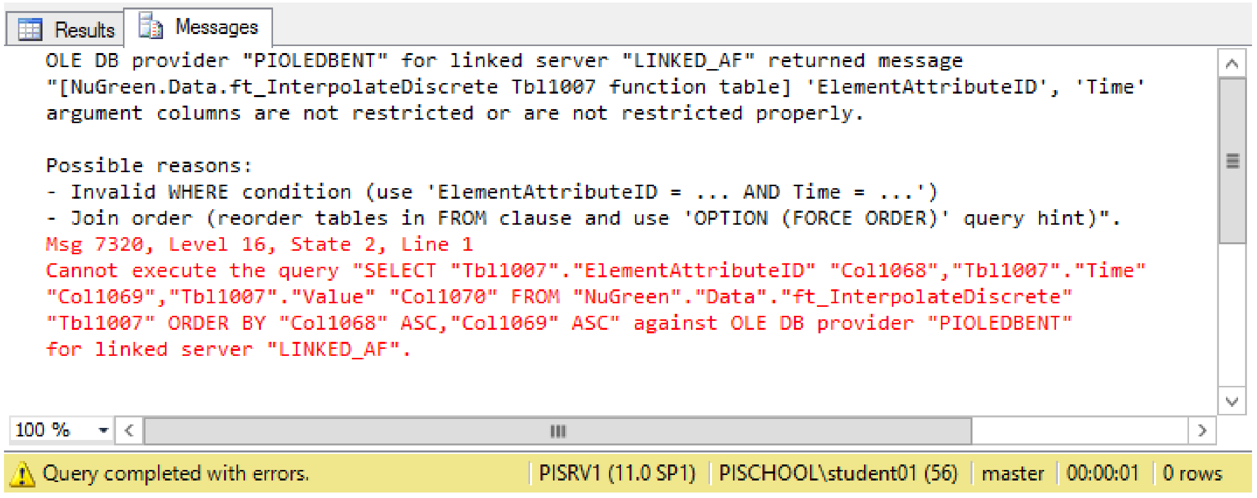
Note: The table and column names may differ since it is generated by the SQL Server.

Optimization hint 1

13. Let's instruct the MS SQL Server to perform the join operation on the remote data source. To do that, we can use a join hint for T-SQL (*INNER REMOTE JOIN* instead of *INNER JOIN*)

```
SELECT ef2.Name EF_Name, e.Name E_Name, ea.Name EA_Name, v.Value
FROM [LINKED_AF].[NuGreen].[EventFrame].[EventFrame] ef2
INNER JOIN [LINKED_AF].[NuGreen].[Asset].[Element] e
 ON e.ID = ef2.PrimaryReferencedElementID
INNER REMOTE JOIN [LINKED_AF].[NuGreen].[Asset].[ElementAttribute] ea
 ON e.ID = ea.ElementID
INNER JOIN [LINKED_AF].[NuGreen].[Data].[ft_InterpolateDiscrete] v
 ON ea.ID = v.ElementAttributeID
WHERE ef2.PrimaryParentID IN
(SELECT TOP 1 ef1.ID
 FROM [LINKED_AF].[NuGreen].[EventFrame].[EventFrame] ef1
 WHERE IsRoot = 1
 AND StartTime > '1-Feb-2015'
 ORDER BY StartTime ASC)
AND v.Time = ef2.endtime
AND ef2.Name Like '%A' -- we just want the 'xxxx_A' child Event Frames
```

14. Copy the above modified query and execute it in SSMS. The query returned another error:



```
OLE DB provider "PIOLEDBENT" for linked server "LINKED_AF" returned message
"[NuGreen.Data.ft_InterpolateDiscrete Tbl1007 function table] 'ElementAttributeID', 'Time'
argument columns are not restricted or are not restricted properly.

Possible reasons:
- Invalid WHERE condition (use 'ElementAttributeID = ... AND Time = ...')
- Join order (reorder tables in FROM clause and use 'OPTION (FORCE ORDER)' query hint)".
Msg 7320, Level 16, State 2, Line 1
Cannot execute the query "SELECT "Tbl1007"."ElementAttributeID" "Col1068","Tbl1007"."Time"
"Col1069","Tbl1007"."Value" "Col1070" FROM "NuGreen"."Data"."ft_InterpolateDiscrete"
"Tbl1007" ORDER BY "Col1068" ASC,"Col1069" ASC" against OLE DB provider "PIOLEDBENT"
for linked server "LINKED_AF".
```

```
Query completed with errors.  | PISRV1 (11.0 SP1) | PISCHOOL\student01 (56) | master | 00:00:01 | 0 rows
```

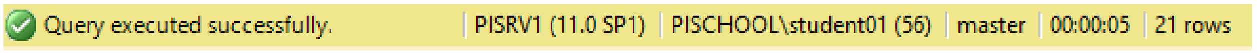
We recognize that MS SQL Server wants to perform another join operation locally and it tries to get the entire data source. This time it wants to pull the NuGreen.Data.ft_InterpolateDiscrete data source to perform local filtering:

```
SELECT "Tbl1007"."ElementAttributeID" "Col1052","Tbl1007"."Time"
"Col1053","Tbl1007"."Value" "Col1054"
FROM "NuGreen"."Data"."ft_InterpolateDiscrete" "Tbl1007"
ORDER BY "Col1052" ASC,"Col1053" ASC
```

15. Let's modify all ***inner joins*** to ***inner remote joins*** and let's execute the query again

```
SELECT ef2.Name EF_Name, e.Name E_Name, ea.Name EA_Name, v.Value
FROM [LINKED_AF].[NuGreen].[EventFrame].[EventFrame] ef2
INNER REMOTE JOIN [LINKED_AF].[NuGreen].[Asset].[Element] e
 ON e.ID = ef2.PrimaryReferencedElementID
INNER REMOTE JOIN [LINKED_AF].[NuGreen].[Asset].[ElementAttribute] ea
 ON e.ID = ea.ElementID
INNER REMOTE JOIN [LINKED_AF].[NuGreen].[Data].[ft_InterpolateDiscrete] v
 ON ea.ID = v.ElementAttributeID
WHERE ef2.PrimaryParentID IN
(SELECT TOP 1 ef1.ID
 FROM [LINKED_AF].[NuGreen].[EventFrame].[EventFrame] ef1
 WHERE IsRoot = 1
 AND StartTime > '1-Feb-2015'
 ORDER BY StartTime ASC)
AND v.Time = ef2.endtime
AND ef2.Name Like '%A' -- we just want the 'xxxx_A' child Event Frames
```

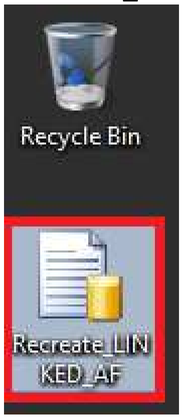
16. The query executed properly but the performance is very poor compared to *PI SQL Commander*, even we instructed MS SQL Server to perform all join operation on the remote data sources



```
Query executed successfully.  | PISRV1 (11.0 SP1) | PISCHOOL\student01 (56) | master | 00:00:05 | 21 rows
```

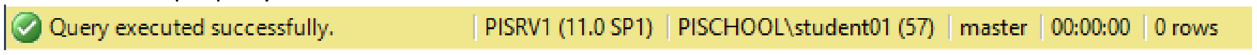
17. We have the result but we do not have any feedback why the query was slower. We need to check the PIOLEDB Enterprise log file to see further details. The log file help us to understand how the query was sent to the OLE DB provider and how the query pieces were executed.

18. First we need to recreate the linked server with a new connection string specifying that the logs should be collected. To do that please navigate to your desktop and open the "Recreate_LINKED_AF.sql" file:

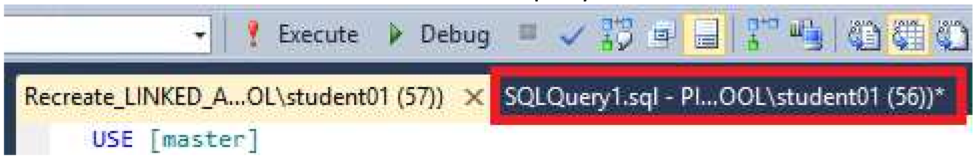


*Please refer to the **PI OLEDB Enterprise 2012 User Guide.pdf** to learn how to setup the logs.*

The file will be opened in *SSMS* in a new tab. Press **F5** to execute the script. Make sure the script was executed properly:


Query executed successfully. | PISRV1 (11.0 SP1) | PISCHOOL\student01 (57) | master | 00:00:00 | 0 rows

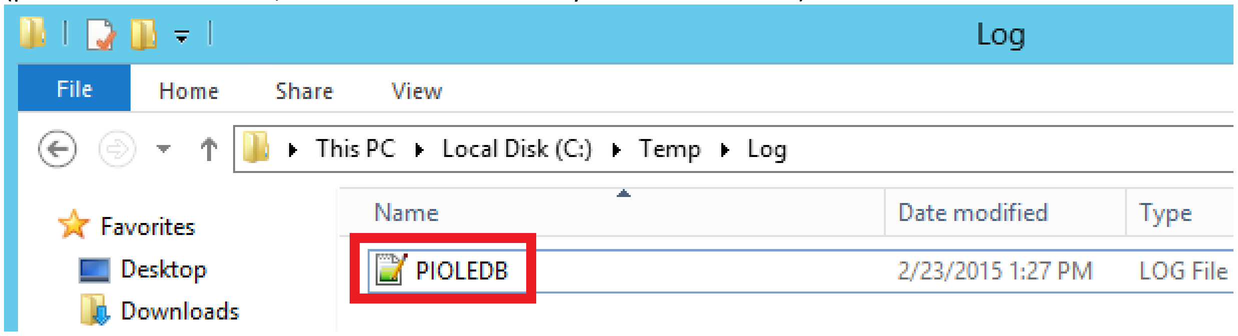
19. Now we can move back to the tab with our query:


Execute
Debug
Recreate_LINKED_A...OL\student01 (57))
SQLQuery1.sql - PI...OOL\student01 (56))*

```
USE [master]
```

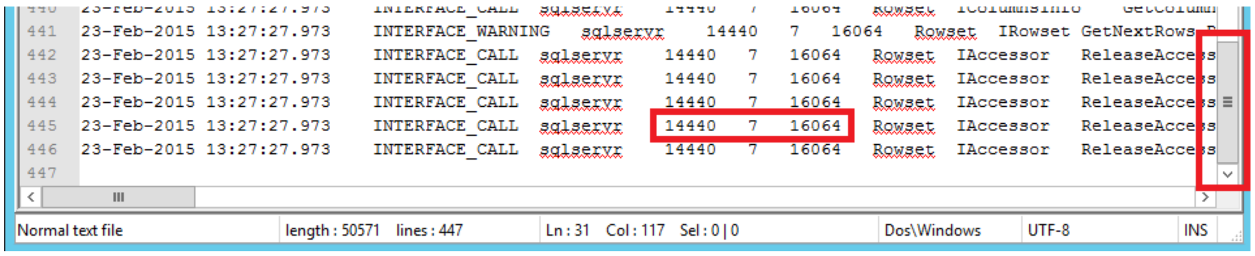
Execute the query again by hitting F5 (this time PIOLEDB Enterprise is collecting logs).

20. Open the windows file explorer and navigate to *C:\Temp\Log* and open the *PIOLEDB.log* file (please use *NotePad++,* which should be used by default in this lab).



Name	Date modified	Type
PIOLEDB	2/23/2015 1:27 PM	LOG File

Navigate to the bottom of the file to find the latest logs:

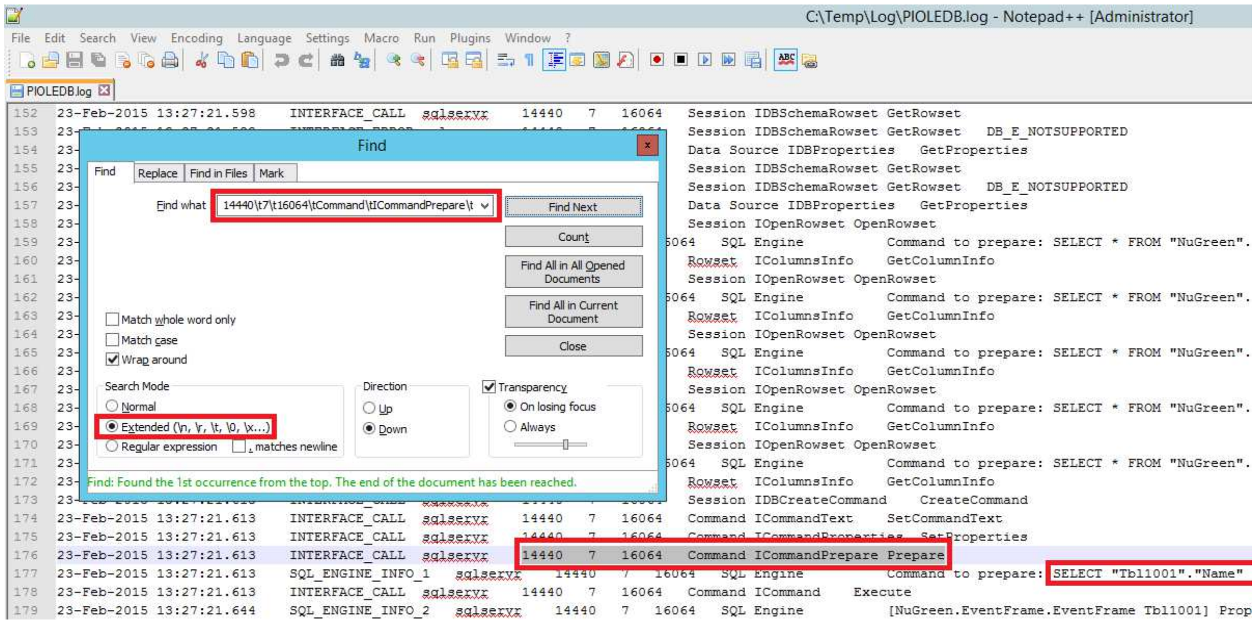


```
441  23-Feb-2015 13:27:27.973  INTERFACE_WARNING  sqlservr  14440  7  16064  Rowset  IRowset GetNextRows
442  23-Feb-2015 13:27:27.973  INTERFACE_CALL  sqlservr  14440  7  16064  Rowset  IAccessor  ReleaseAccess
443  23-Feb-2015 13:27:27.973  INTERFACE_CALL  sqlservr  14440  7  16064  Rowset  IAccessor  ReleaseAccess
444  23-Feb-2015 13:27:27.973  INTERFACE_CALL  sqlservr  14440  7  16064  Rowset  IAccessor  ReleaseAccess
445  23-Feb-2015 13:27:27.973  INTERFACE_CALL  sqlservr  14440  7  16064  Rowset  IAccessor  ReleaseAccess
446  23-Feb-2015 13:27:27.973  INTERFACE_CALL  sqlservr  14440  7  16064  Rowset  IAccessor  ReleaseAccess
447
Normal text file   length : 50571   lines : 447   Ln : 31   Col : 117   Sel : 0 | 0   Dos\Windows   UTF-8   INS
```

The outlined by red color numbers (*14440, 7, 16064*) are the *process ID, thread ID and instance ID*. We will use this numbers as a reference, if a particular line is part of our query. Please make sure you refer to your *process ID, thread ID and instance ID*, which you will find in your local log file.

You can check the process ID of the MS SQL Server instance in the task manager.

21. Let's look for all instances of a prepare query command in the log file. We can use the following text as search pattern to find it (*copy and paste it, and replace the highlighted numbers*): *"**14440**\t**7**\t**16064**\t**Command**\t**ICommandPrepare**\t**Prepare**"* (the **\t** symbol is equal to a **tab** key). To open the find window in *Notepad++* press **Ctrl+F**. In the next step make sure you are using the **extended** *search mode* like in the screenshot below:



C:\Temp\Log\PIOLEDB.log - Notepad++ [Administrator]

```
152  23-Feb-2015 13:27:21.598  INTERFACE_CALL  sqlservr  14440  7  16064  Session IDBSchemaRowset GetRowset
153  23-                                                                Session IDBSchemaRowset GetRowset  DB_E_NOTSUPPORTED
154  23-                                                                Data Source IDBProperties  GetProperties
155  23-                                                                Session IDBSchemaRowset GetRowset
156  23-                                                                Session IDBSchemaRowset GetRowset  DB_E_NOTSUPPORTED
157  23-                                                                Data Source IDBProperties  GetProperties
158  23-                                                                Session IOpenRowset OpenRowset
159  23-                                                            5064  SQL Engine  Command to prepare: SELECT * FROM "NuGreen".
160  23-                                                                Rowset  IColumnsInfo  GetColumnInfo
161  23-                                                                Session IOpenRowset OpenRowset
162  23-                                                            5064  SQL Engine  Command to prepare: SELECT * FROM "NuGreen".
163  23-                                                                Rowset  IColumnsInfo  GetColumnInfo
164  23-                                                                Session IOpenRowset OpenRowset
165  23-                                                            5064  SQL Engine  Command to prepare: SELECT * FROM "NuGreen".
166  23-                                                                Rowset  IColumnsInfo  GetColumnInfo
167  23-                                                                Session IOpenRowset OpenRowset
168  23-                                                            5064  SQL Engine  Command to prepare: SELECT * FROM "NuGreen".
169  23-                                                                Rowset  IColumnsInfo  GetColumnInfo
170  23-                                                                Session IOpenRowset OpenRowset
171  23-                                                            5064  SQL Engine  Command to prepare: SELECT * FROM "NuGreen".
172  23-                                                                Rowset  IColumnsInfo  GetColumnInfo
173  23-                                                                Session IDBCreateCommand  CreateCommand
174  23-Feb-2015 13:27:21.613  INTERFACE_CALL  sqlservr  14440  7  16064  Command ICommandText  SetCommandText
175  23-Feb-2015 13:27:21.613  INTERFACE_CALL  sqlservr  14440  7  16064  Command ICommandProperties  SetProperties
176  23-Feb-2015 13:27:21.613  INTERFACE_CALL  sqlservr  14440  7  16064  Command ICommandPrepare Prepare
177  23-Feb-2015 13:27:21.613  SQL_ENGINE_INFO_1  sqlservr  14440  7  16064  SQL Engine  Command to prepare: SELECT "Tbl1001"."Name"
178  23-Feb-2015 13:27:21.613  INTERFACE_CALL  sqlservr  14440  7  16064  Command ICommand  Execute
179  23-Feb-2015 13:27:21.644  SQL_ENGINE_INFO_2  sqlservr  14440  7  16064  SQL Engine  [NuGreen.EventFrame.EventFrame Tbl1001] Prop
```

Find what: 14440\t7\t16064\tCommand\tICommandPrepare\t

Match whole word only [ ]
Match case [ ]
Wrap around [✓]

Search Mode: Normal [✓ Extended (\n, \r, \t, \0, \x...)] Regular expression
Direction: Up [✓ Down]
Transparency [✓]: [✓ On losing focus] Always

Find: Found the 1st occurrence from the top. The end of the document has been reached.

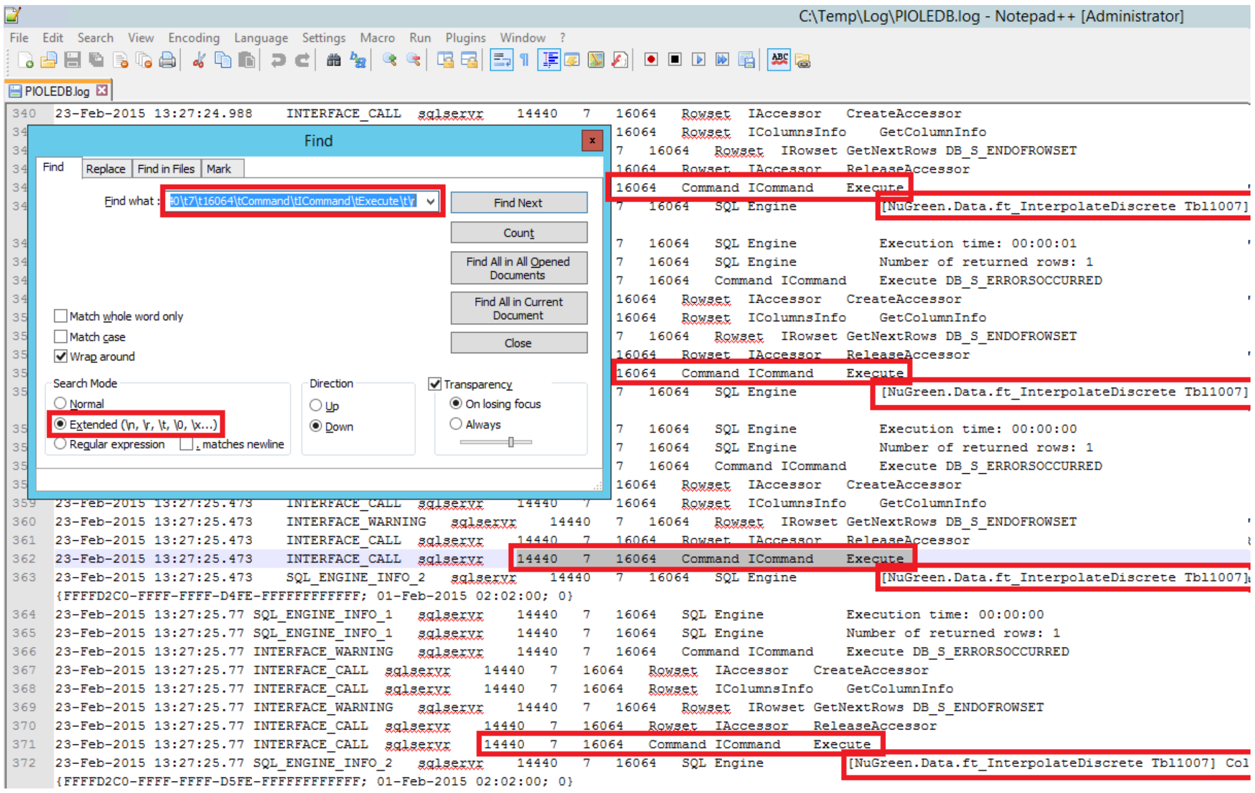
For this particular query we managed to find five instances of our search pattern:

a. ***SELECT** "Tbl1001"."Name" "Col1035","Tbl1001"."EndTime" "Col1036","Tbl1001"."PrimaryParentID" "Col1037","Tbl1001"."PrimaryReferencedElementID" "Col1038" **FROM "NuGreen"."EventFrame"."EventFrame"** "Tbl1001" **WHERE** "Tbl1001"."Name" like '%A' **ORDER BY** "Col1037" **ASC***

b. ***SELECT** "Tbl1009"."ID" "Col1027","Tbl1009"."StartTime" "Col1028" **FROM** **"NuGreen"."EventFrame"."EventFrame"** "Tbl1009" **WHERE** "Tbl1009"."IsRoot"=(1) AND "Tbl1009"."StartTime">'2015-02-16 00:00:00.0000000' ORDER BY "Col1028" ASC*
c. ***SELECT** "Tbl1003"."ID" "Col1043","Tbl1003"."Name" "Col1044" **FROM** **"NuGreen"."Asset"."Element"** "Tbl1003" **WHERE** "Tbl1003"."ID"=?*
d. ***SELECT** "Tbl1005"."ID" "Col1056","Tbl1005"."Name" "Col1057" **FROM** **"NuGreen"."Asset"."ElementAttribute"** "Tbl1005" **WHERE** "Tbl1005"."ElementID"=?*
e. ***SELECT** "Tbl1007"."Value" "Col1068" FROM **"NuGreen"."Data"."ft_InterpolateDiscrete"** "Tbl1007" WHERE "Tbl1007"."ElementAttributeID"=? AND "Tbl1007"."Time"=?*

We can see that the query was sent from MS SQL Server to PIOLEDB Enterprise in five subqueries.

22. In the next step we may look at the number of executions (how many times each subquery was executed by PIOLEDB Enterprise). We can find it by looking for this search pattern (*copy and paste it, and replace the highlighted numbers*):
*"14440**\t**7**\t**16064**\t**Command**\t**ICommand**\t**Execute**\t\r**"*


C:\Temp\Log\PIOLEDB.log - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
PIOLEDB.log
Find
Find Replace Find in Files Mark
Find Next
Count
Find All in All Opened Documents
Find All in Current Document
Close
Direction
Up
Down

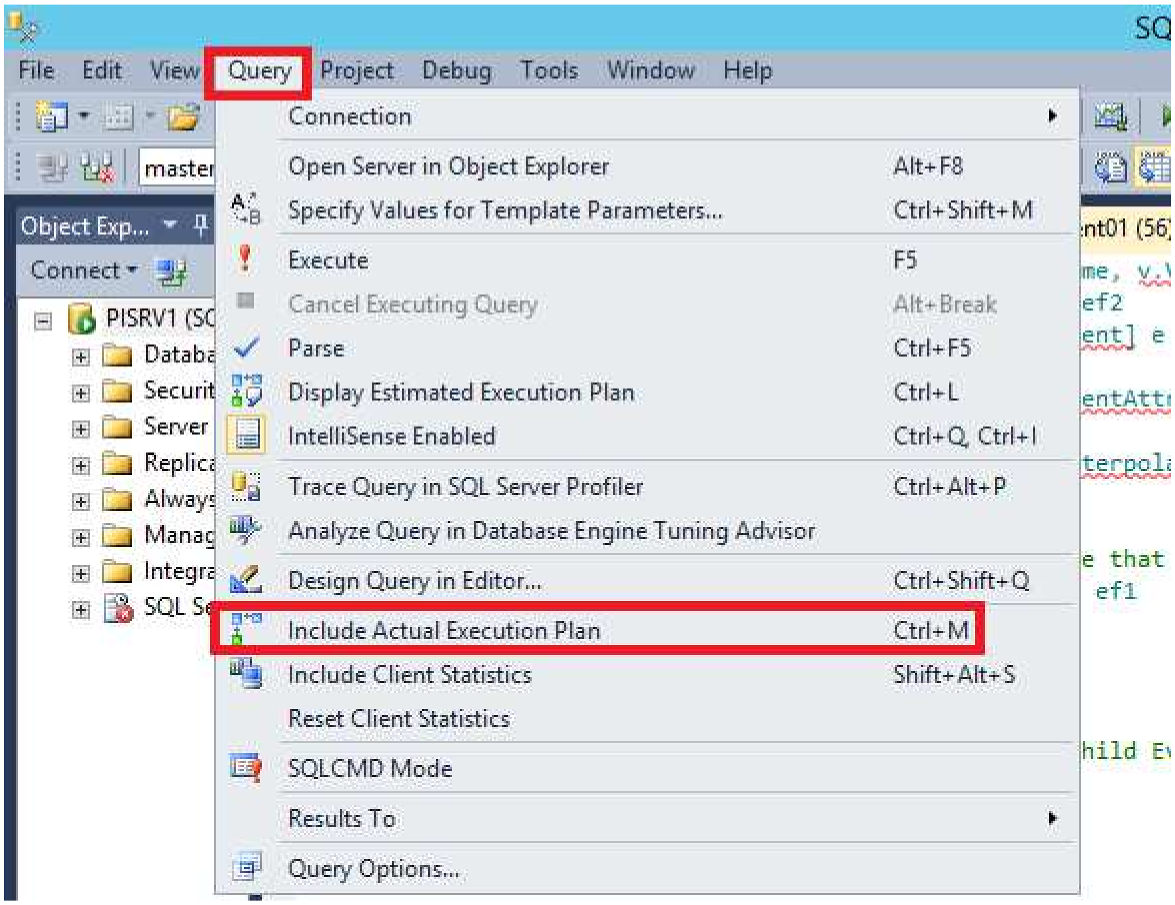
Find what : 0\t7\t16064\tCommand\tICommand\tExecute\t\r
[ ] Match whole word only
[ ] Match case
[✓] Wrap around
Search Mode: ( ) Normal [✓ Extended (\n, \r, \t, \0, \x...)] ( ) Regular expression [ ] . matches newline
Direction: ( ) Up [✓ Down]
[✓] Transparency: [✓ On losing focus] ( ) Always

```
23-Feb-2015 13:27:24.988   INTERFACE_CALL  sqlservr    14440   7   16064   Rowset  IAccessor   CreateAccessor
                                                                   16064   Rowset  IColumnsInfo    GetColumnInfo
                                                                   7   16064   Rowset  IRowset GetNextRows DB_S_ENDOFROWSET
                                                                   16064   Rowset  IAccessor   ReleaseAccessor
                                                                   16064   Command ICommand    Execute
                                                                   7   16064   SQL Engine      [NuGreen.Data.ft_InterpolateDiscrete Tbl1007]

                                                                   7   16064   SQL Engine      Execution time: 00:00:01
                                                                   7   16064   SQL Engine      Number of returned rows: 1
                                                                   7   16064   Command ICommand    Execute DB_S_ERRORSOCCURRED
                                                                   16064   Rowset  IAccessor   CreateAccessor
                                                                   16064   Rowset  IColumnsInfo    GetColumnInfo
                                                                   7   16064   Rowset  IRowset GetNextRows DB_S_ENDOFROWSET
                                                                   16064   Rowset  IAccessor   ReleaseAccessor
                                                                   16064   Command ICommand    Execute
                                                                   7   16064   SQL Engine      [NuGreen.Data.ft_InterpolateDiscrete Tbl1007]

                                                                   7   16064   SQL Engine      Execution time: 00:00:00
                                                                   7   16064   SQL Engine      Number of returned rows: 1
                                                                   7   16064   Command ICommand    Execute DB_S_ERRORSOCCURRED
                                                                   16064   Rowset  IAccessor   CreateAccessor
23-Feb-2015 13:27:25.473   INTERFACE_CALL  sqlservr    14440   7   16064   Rowset  IColumnsInfo    GetColumnInfo
23-Feb-2015 13:27:25.473   INTERFACE_WARNING  sqlservr    14440   7   16064   Rowset  IRowset GetNextRows DB_S_ENDOFROWSET
23-Feb-2015 13:27:25.473   INTERFACE_CALL  sqlservr    14440   7   16064   Rowset  IAccessor   ReleaseAccessor
23-Feb-2015 13:27:25.473   INTERFACE_CALL  sqlservr    14440   7   16064   Command ICommand    Execute
23-Feb-2015 13:27:25.473   SQL_ENGINE_INFO_2  sqlservr    14440   7   16064   SQL Engine      [NuGreen.Data.ft_InterpolateDiscrete Tbl1007]
{FFFFD2C0-FFFF-FFFF-D4FE-FFFFFFFFFFFF; 01-Feb-2015 02:02:00; 0}
23-Feb-2015 13:27:25.77 SQL_ENGINE_INFO_1    sqlservr    14440   7   16064   SQL Engine      Execution time: 00:00:00
23-Feb-2015 13:27:25.77 SQL_ENGINE_INFO_1    sqlservr    14440   7   16064   SQL Engine      Number of returned rows: 1
23-Feb-2015 13:27:25.77 INTERFACE_WARNING    sqlservr    14440   7   16064   Command ICommand    Execute DB_S_ERRORSOCCURRED
23-Feb-2015 13:27:25.77 INTERFACE_CALL  sqlservr    14440   7   16064   Rowset  IAccessor   CreateAccessor
23-Feb-2015 13:27:25.77 INTERFACE_CALL  sqlservr    14440   7   16064   Rowset  IColumnsInfo    GetColumnInfo
23-Feb-2015 13:27:25.77 INTERFACE_WARNING    sqlservr    14440   7   16064   Rowset  IRowset GetNextRows DB_S_ENDOFROWSET
23-Feb-2015 13:27:25.77 INTERFACE_CALL  sqlservr    14440   7   16064   Rowset  IAccessor   ReleaseAccessor
23-Feb-2015 13:27:25.77 INTERFACE_CALL  sqlservr    14440   7   16064   Command ICommand    Execute
23-Feb-2015 13:27:25.77 SQL_ENGINE_INFO_2    sqlservr    14440   7   16064   SQL Engine      [NuGreen.Data.ft_InterpolateDiscrete Tbl1007] Col
{FFFFD2C0-FFFF-FFFF-D5FE-FFFFFFFFFFFF; 01-Feb-2015 02:02:00; 0}
```

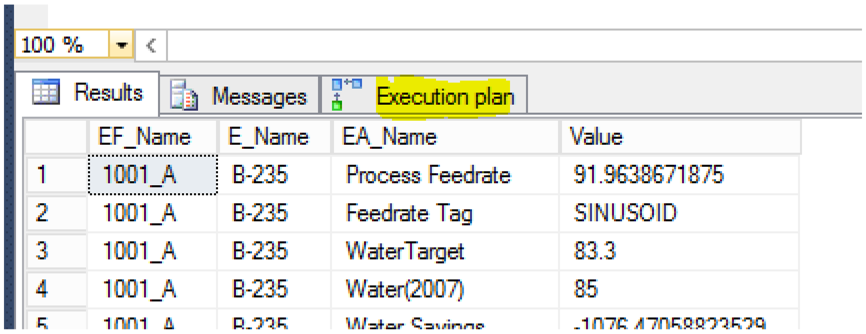
23. When we look at the number of executions for each query, we discover that the first four subqueries were executed only once (that is correct) but the last command was executed 21 times. We would expect that this command is executed only once but MS SQL Server passes the filter parameter one by one because it has performed local filtering.

24. Second observation from the captured queries, is that we never saw the “Top” keyword sent to PIOLEDB Enterprise. This means that the MS SQL Server did not pass this expression to the OLE DB provider and it pulls the data source to perform the operation locally.

25. Let’s navigate back to ***SSMS***. We will now look if we can find the same information in the *SSMS* tools. To do that, navigate to *Query* toolbar menu and select the “*Include Actual Execution Plan*”

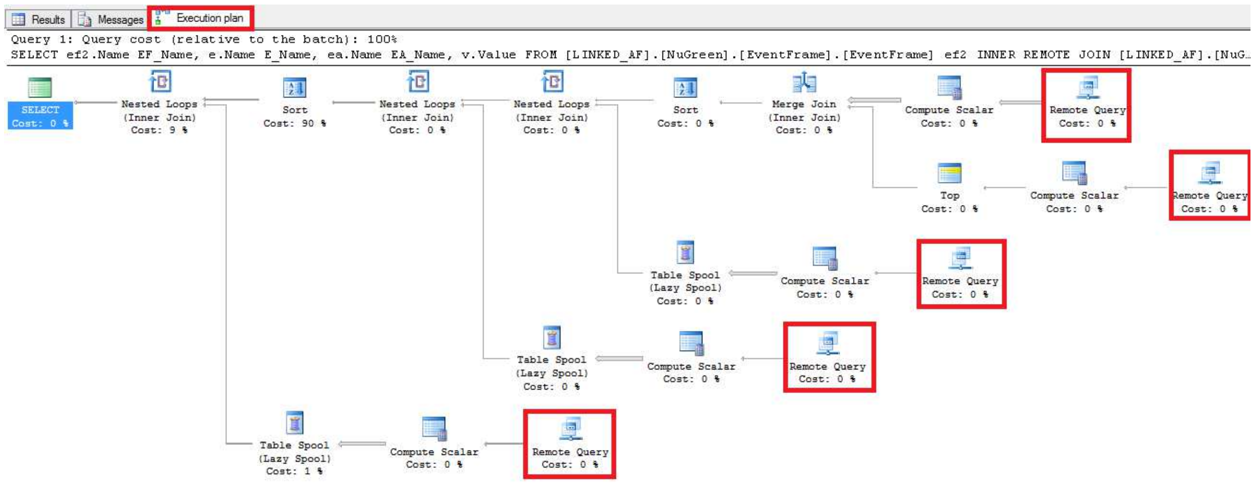

File
Edit
View
Query
Project
Debug
Tools
Window
Help
master
Object Exp...
Connect
PISRV1 (SQ
Databa
Securit
Server
Replica
Always
Manag
Integra
SQL Se
Connection
Open Server in Object Explorer Alt+F8
Specify Values for Template Parameters... Ctrl+Shift+M
Execute F5
Cancel Executing Query Alt+Break
Parse Ctrl+F5
Display Estimated Execution Plan Ctrl+L
IntelliSense Enabled Ctrl+Q, Ctrl+I
Trace Query in SQL Server Profiler Ctrl+Alt+P
Analyze Query in Database Engine Tuning Advisor
Design Query in Editor... Ctrl+Shift+Q
Include Actual Execution Plan Ctrl+M
Include Client Statistics Shift+Alt+S
Reset Client Statistics
SQLCMD Mode
Results To
Query Options...

26. Execute the query again by hitting **F5** and take a look at the additional tab “*Execution Plan*”



	EF_Name	E_Name	EA_Name	Value
1	1001_A	B-235	Process Feedrate	91.9638671875
2	1001_A	B-235	Feedrate Tag	SINUSOID
3	1001_A	B-235	WaterTarget	83.3
4	1001_A	B-235	Water(2007)	85
5	1001_A	B-235	Water Savings	-1076.47058823529

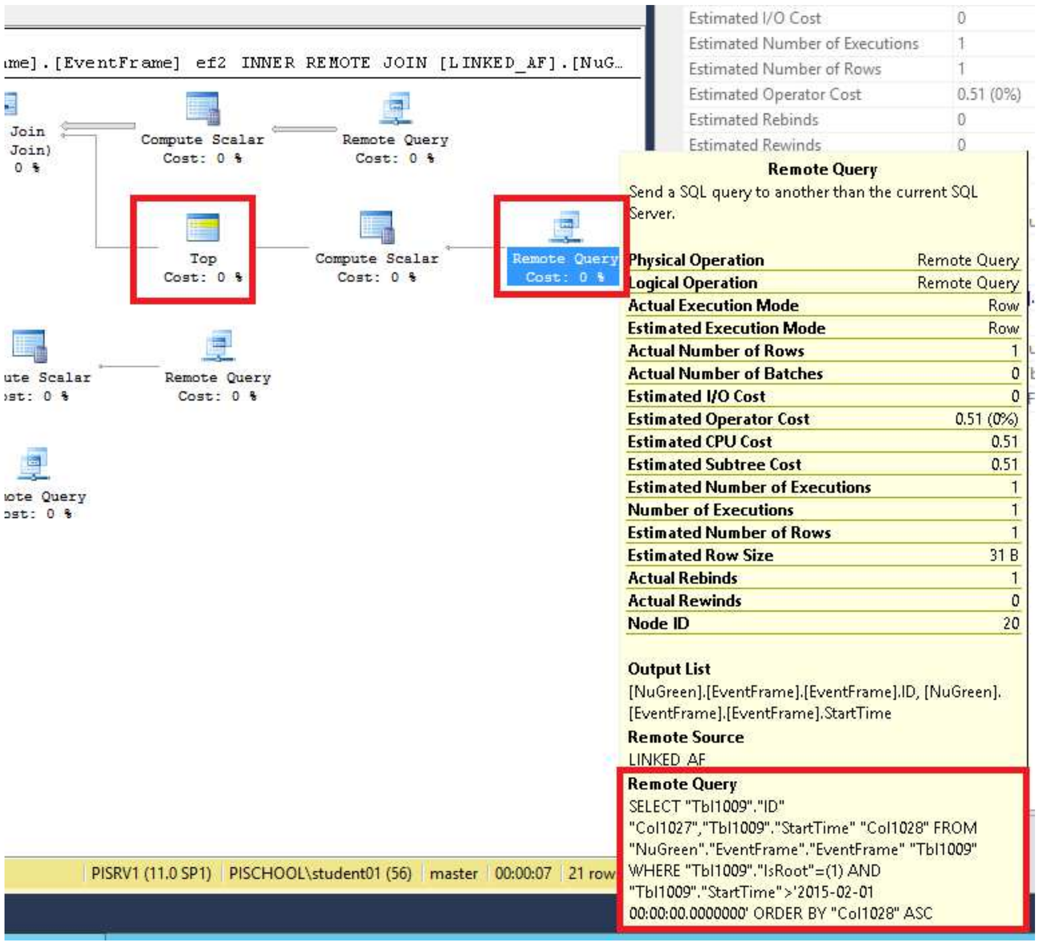
27. Let’s quickly review the execution plan



```
Query 1: Query cost (relative to the batch): 100%
SELECT ef2.Name EF_Name, e.Name E_Name, ea.Name EA_Name, v.Value FROM [LINKED_AF].[NuGreen].[EventFrame].[EventFrame] ef2 INNER REMOTE JOIN [LINKED_AF].[NuG…
```

All nodes with the name “Remote Query” (outlined red) are executed by the OLE DB provider. The rest is executed on the SQL Server side.

28. We realize that all remote queries displayed in this view represent the same queries we captured in the log file. Let’s hover over the second “*Remote Query*” node from the top. We can see that the remote query does not contain the *Top* expression and that the *Top* expression is present on MS SQL Server side:



Estimated I/O Cost	0
Estimated Number of Executions	1
Estimated Number of Rows	1
Estimated Operator Cost	0.51 (0%)
Estimated Rebinds	0
Estimated Rewinds	0

Remote Query

Send a SQL query to another than the current SQL Server.

Physical Operation	Remote Query
Logical Operation	Remote Query
Actual Execution Mode	Row
Estimated Execution Mode	Row
Actual Number of Rows	1
Actual Number of Batches	0
Estimated I/O Cost	0
Estimated Operator Cost	0.51 (0%)
Estimated CPU Cost	0.51
Estimated Subtree Cost	0.51
Estimated Number of Executions	1
Number of Executions	1
Estimated Number of Rows	1
Estimated Row Size	31 B
Actual Rebinds	1
Actual Rewinds	0
Node ID	20

Output List

[NuGreen].[EventFrame].[EventFrame].ID, [NuGreen].[EventFrame].[EventFrame].StartTime

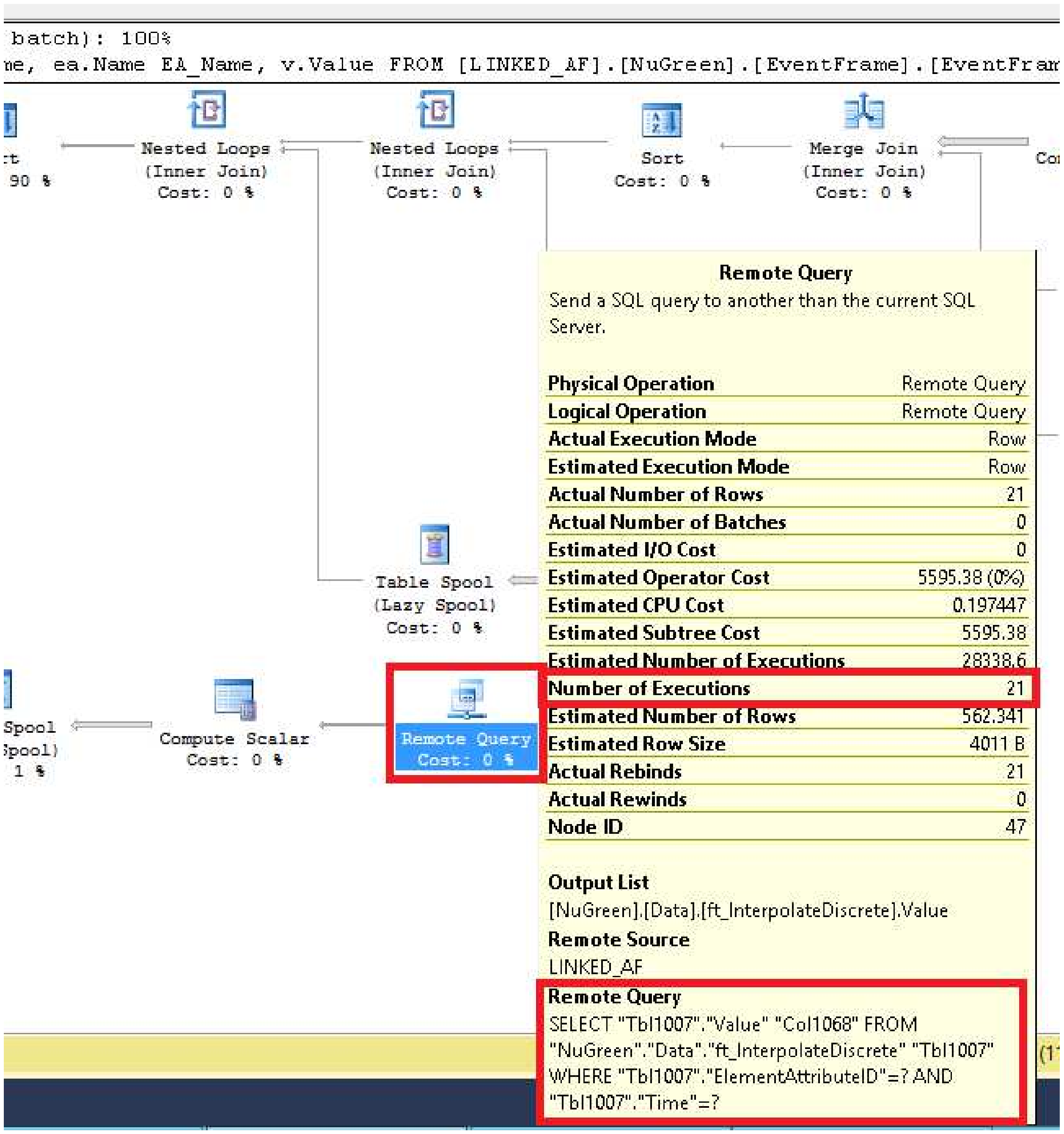
Remote Source

LINKED_AF

Remote Query

```
SELECT "Tbl1009"."ID"
"Col1027","Tbl1009"."StartTime" "Col1028" FROM
"NuGreen"."EventFrame"."EventFrame" "Tbl1009"
WHERE "Tbl1009"."IsRoot"=(1) AND
"Tbl1009"."StartTime">'2015-02-01
00:00:00.0000000' ORDER BY "Col1028" ASC
```

29. Hover now over the bottom “Remote Query” node


batch): 100%
ne, ea.Name EA_Name, v.Value FROM [LINKED_AF].[NuGreen].[EventFrame].[EventFram
Nested Loops (Inner Join) Cost: 0 %
Nested Loops (Inner Join) Cost: 0 %
Sort Cost: 0 %
Merge Join (Inner Join) Cost: 0 %
Table Spool (Lazy Spool) Cost: 0 %
Compute Scalar Cost: 0 %
Remote Query Cost: 0 %

Remote Query
Send a SQL query to another than the current SQL Server.

Physical Operation	Remote Query
Logical Operation	Remote Query
Actual Execution Mode	Row
Estimated Execution Mode	Row
Actual Number of Rows	21
Actual Number of Batches	0
Estimated I/O Cost	0
Estimated Operator Cost	5595.38 (0%)
Estimated CPU Cost	0.197447
Estimated Subtree Cost	5595.38
Estimated Number of Executions	28338.6
Number of Executions	21
Estimated Number of Rows	562.341
Estimated Row Size	4011 B
Actual Rebinds	21
Actual Rewinds	0
Node ID	47

Output List
[NuGreen].[Data].[ft_InterpolateDiscrete].Value
Remote Source
LINKED_AF
Remote Query
```
SELECT "Tbl1007"."Value" "Col1068" FROM
"NuGreen"."Data"."ft_InterpolateDiscrete" "Tbl1007"
WHERE "Tbl1007"."ElementAttributeID"=? AND
"Tbl1007"."Time"=?
```

We recognize that the remote query was executed 21 times.

Optimization hint 2

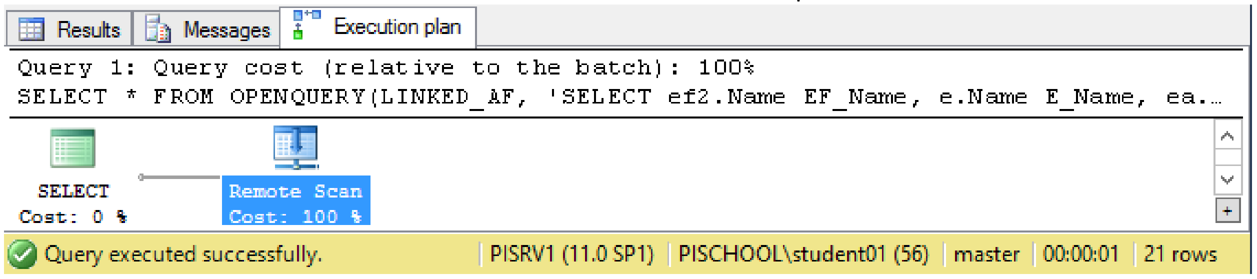
30. Let’s check the possibilities to work around the issues we found in the PIOLEDB Enterprise log and the *SSMS* tool. First we will try to use a T-SQL function which will pass the entire query as is to the OLE DB provider:

```
SELECT * FROM OPENQUERY(LINKED_AF, 'SELECT ef2.Name EF_Name, e.Name E_Name,
ea.Name EA_Name, v.Value
FROM [NuGreen].[EventFrame].[EventFrame] ef2
INNER JOIN [NuGreen].[Asset].[Element] e
 ON e.ID = ef2.PrimaryReferencedElementID
INNER JOIN [NuGreen].[Asset].[ElementAttribute] ea
 ON e.ID = ea.ElementID
INNER JOIN [NuGreen].[Data].[ft_InterpolateDiscrete] v
 ON ea.ID = v.ElementAttributeID
WHERE ef2.PrimaryParentID IN
    (SELECT TOP 1 ef1.ID -- we want the first event frame that started yesterday
     FROM [NuGreen].[EventFrame].[EventFrame] ef1
     WHERE IsRoot = True
     AND StartTime > ''1-Feb-2015''
     ORDER BY StartTime ASC)
AND v.Time = ef2.endtime
AND ef2.Name Like ''%A'' -- we just want the ''xxxx_A'' child Event Frames')
```

The OPENQUERY T-SQL function executes the specified pass-through query in second parameter on the specified linked server in the first parameter. *The OPENQUERY can be referenced in the FROM clause of a query as if it were a table name.*

31. Copy the query from above, paste it to SSMS query editor and execute it by hitting **F5**.

32. Notice the execution time. And also take a look and the execution plan tab:



```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM OPENQUERY(LINKED_AF, 'SELECT ef2.Name EF_Name, e.Name E_Name, ea...
```

Query executed successfully. | PISRV1 (11.0 SP1) | PISCHOOL\student01 (56) | master | 00:00:01 | 21 rows

We can see, that the only operation that MS SQL Server performed is the scan operation.

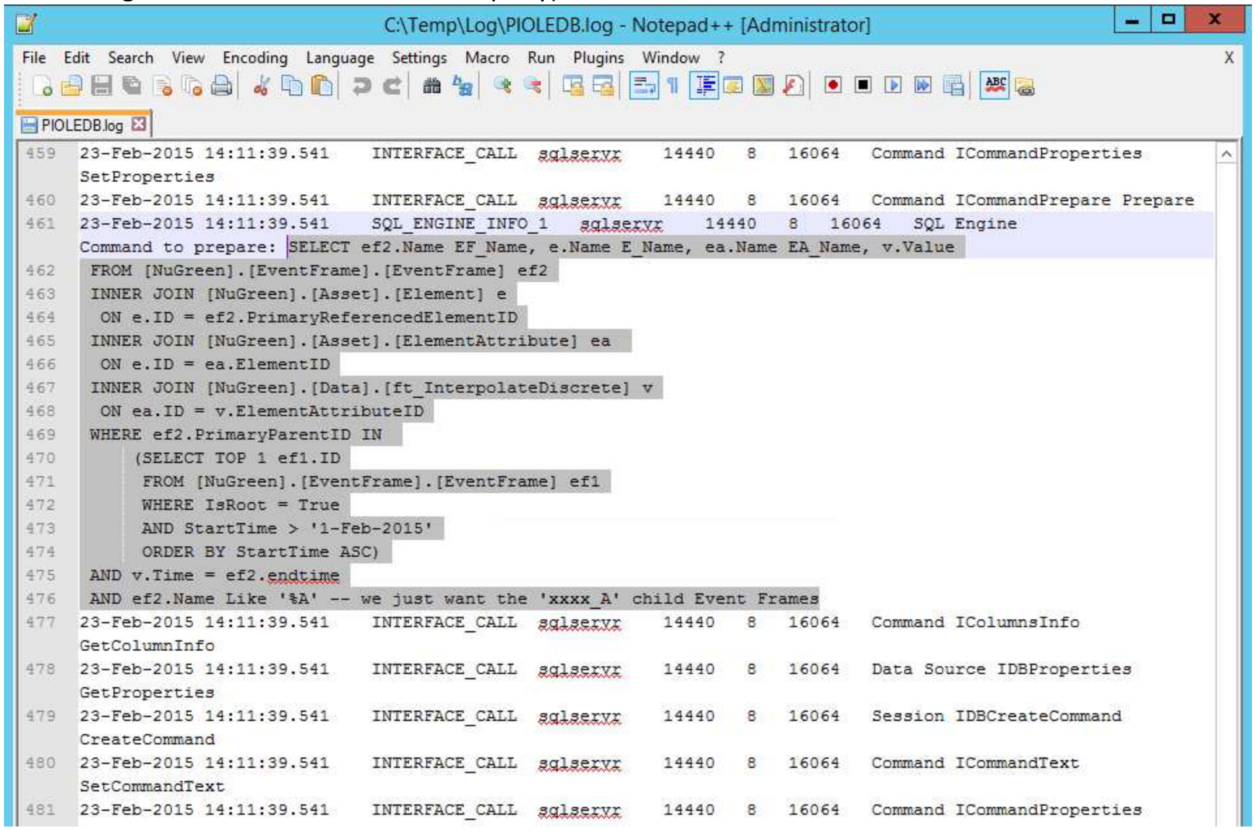
Advantages of this technique:

1. The entire query is executed remotely. No unnecessary data is pulled local beside the actual result.
2. We can use PI specific functions and literals.

Disadvantages of this technique:

1. We cannot join local MS SQL Server tables with the tables from AF (in that case you need to use the previous heterogeneous query example we were investigating and use the Inner Remote Join hint to increase performance)

33. If we take a look at the log file, we will see that the entire query was passed to the OLE DB (we even recognize the comment inside the query).


C:\Temp\Log\PIOLEDB.log - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
PIOLEDB.log

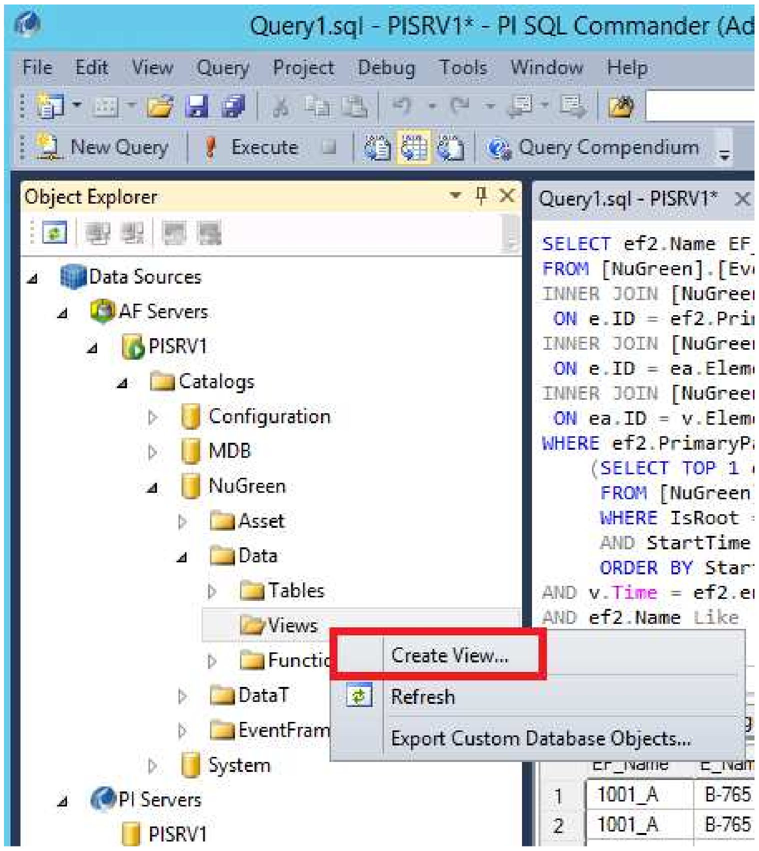
```
23-Feb-2015 14:11:39.541    INTERFACE_CALL  sqlservr    14440   8   16064   Command ICommandProperties
SetProperties
23-Feb-2015 14:11:39.541    INTERFACE_CALL  sqlservr    14440   8   16064   Command ICommandPrepare Prepare
23-Feb-2015 14:11:39.541    SQL_ENGINE_INFO_1   sqlservr    14440   8   16064   SQL Engine
Command to prepare: SELECT ef2.Name EF_Name, e.Name E_Name, ea.Name EA_Name, v.Value
 FROM [NuGreen].[EventFrame].[EventFrame] ef2
 INNER JOIN [NuGreen].[Asset].[Element] e
  ON e.ID = ef2.PrimaryReferencedElementID
 INNER JOIN [NuGreen].[Asset].[ElementAttribute] ea
  ON e.ID = ea.ElementID
 INNER JOIN [NuGreen].[Data].[ft_InterpolateDiscrete] v
  ON ea.ID = v.ElementAttributeID
 WHERE ef2.PrimaryParentID IN
     (SELECT TOP 1 ef1.ID
      FROM [NuGreen].[EventFrame].[EventFrame] ef1
      WHERE IsRoot = True
      AND StartTime > '1-Feb-2015'
      ORDER BY StartTime ASC)
 AND v.Time = ef2.endtime
 AND ef2.Name Like '%A' -- we just want the 'xxxx_A' child Event Frames
23-Feb-2015 14:11:39.541    INTERFACE_CALL  sqlservr    14440   8   16064   Command IColumnsInfo
GetColumnInfo
23-Feb-2015 14:11:39.541    INTERFACE_CALL  sqlservr    14440   8   16064   Data Source IDBProperties
GetProperties
23-Feb-2015 14:11:39.541    INTERFACE_CALL  sqlservr    14440   8   16064   Session IDBCreateCommand
CreateCommand
23-Feb-2015 14:11:39.541    INTERFACE_CALL  sqlservr    14440   8   16064   Command ICommandText
SetCommandText
23-Feb-2015 14:11:39.541    INTERFACE_CALL  sqlservr    14440   8   16064   Command ICommandProperties
```

If we had another look at the internals of the execution in the log file, we would also see that the query internally was executed in five pieces. The main difference beside that the *Top* expression is present in PI OLEDB Enterprise command, each of this subqueries we have seen previously would be executed only once.

Optimization hint 3

34. The second solution could be to wrap the query or parts of it into a PI SQL View and use this View on the SQL Server side. This can for example help with data type incompatibilities or query structures where you cannot give SQL Server a hint (the INNER REMOTE keyword is such a hint) that prevents local filtering. Or if you would like to use PI specific functions or literals.

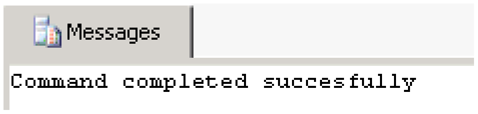
35. Navigate to ***PI SQL Commander*** and right click on the *NuGreen->Data->Views* node and select the "*Create View…*" option:



36. Replace the <*view name*> with a name "MyData" and replace the <*query*> with the query we used before in *PI SQL Commander.* You can also copy and paste the query below.

```
CREATE VIEW [NuGreen].[Data].[MyData]
AS
SELECT ef2.Name EF_Name, e.Name E_Name, ea.Name EA_Name, v.Value
FROM [NuGreen].[EventFrame].[EventFrame] ef2
INNER JOIN [NuGreen].[Asset].[Element] e
 ON e.ID = ef2.PrimaryReferencedElementID
INNER JOIN [NuGreen].[Asset].[ElementAttribute] ea
 ON e.ID = ea.ElementID
INNER JOIN [NuGreen].[Data].[ft_InterpolateDiscrete] v
 ON ea.ID = v.ElementAttributeID
WHERE ef2.PrimaryParentID IN
   (SELECT TOP 1 ef1.ID -- we want the first event frame that started yesterday
    FROM [NuGreen].[EventFrame].[EventFrame] ef1
    WHERE IsRoot = True
    AND StartTime > '1-Feb-2015'
    ORDER BY StartTime ASC)
AND v.Time = ef2.endtime
AND ef2.Name Like '%A' -- we just want the 'xxxx_A' child Event Frames
```

37. Execute the create statement.
Result will look similar to this:



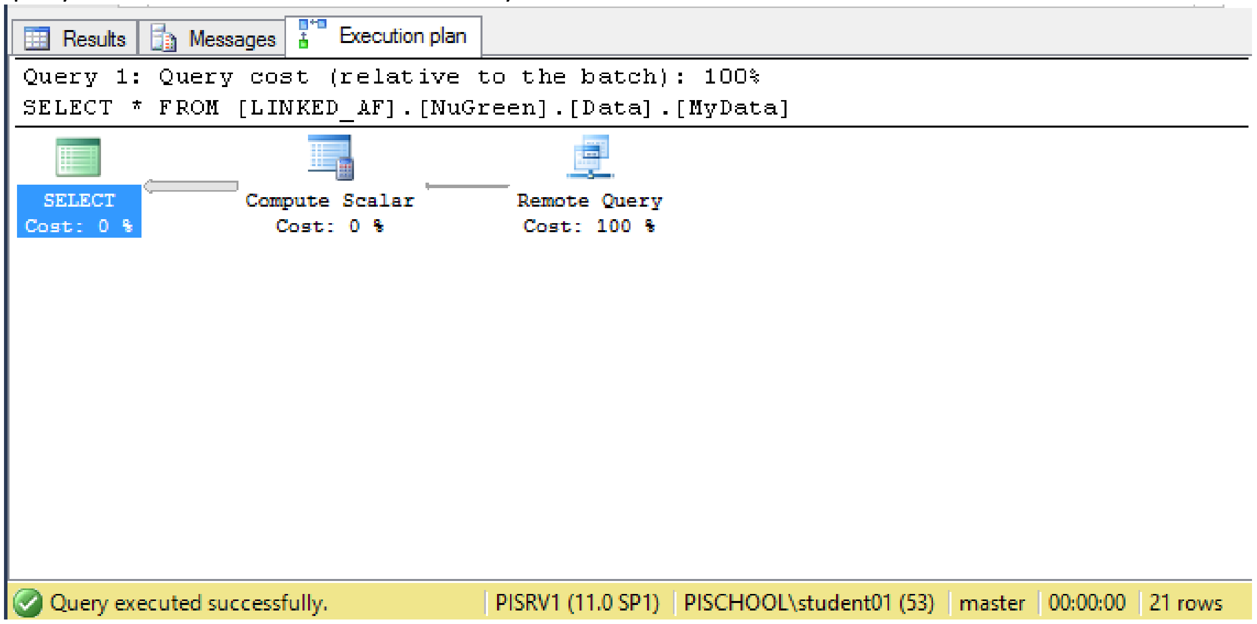
```
Command completed succesfully
```

Note: If you get the following error
[OSIsoft.AFSDK] Cannot modify Element 'Database Objects' in Element 'PI SQL' in Element 'OSIsoft' because the current user does not have Write permission.
use PI System Explorer to verify/set write permission for your user to the Configuration database and to the elements mentioned in the error message. You may also need to restart PI SQL Commander using "Run as Administrator" option.

38. After the view was created we can now navigate back to ***SSMS*** and try to pull data from the view and check the execution plan. To do that enter the query below in the query editor and hit **F5**:

```
SELECT * FROM [LINKED_AF].[NuGreen].[Data].[MyData]
```

The execution time should be as quick as in PI SQL Commander and we will realize that the query was executed in one "Remote Query":



```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM [LINKED_AF].[NuGreen].[Data].[MyData]
```

Query executed successfully. | PISRV1 (11.0 SP1) | PISCHOOL\student01 (53) | master | 00:00:00 | 21 rows

Tip

Independent which method you use to pull data with a third party tool, have always in mind the query hints you learned in the first part of this learning lab.

OSIsoft Virtual Learning Environment

The OSIsoft Virtual Environment provides you with virtual machines where you can complete the exercises contained in this workbook. After you launch the Virtual Learning Environment, connect to **PISRV1** with the credentials: **pischool\student01, student**.

The environment contains the following machines:

PISRV1: a windows server that runs the PI System and that contains all the software and configuration necessary to perform the exercises on this workbook. This is the machine you need to connect to. This machine cannot be accessed from the outside except by rdp, however, from inside the machine, you can access Coresight and other applications with the url: http://pisrv1/, (i.e. http://pisrv1/coresight).

PIDC: a domain controller that provides network and authentication functions.

The system will create these machines for you upon request and this process may take between 5 to 10 minutes. During that time you can start reading the workbook to understand what you will be doing in the machine.

After you launch the virtual learning environment your session will run for up to 8 hours, after which your session will be deleted. You can save your work by using a cloud storage solution like onedrive or box. From the virtual learning environment you can access any of these cloud solutions and upload the files you are interested in saving.

System requirements: the Virtual Learning Environment is composed of virtual machines hosted on Microsoft Azure that you can access remotely. In order to access these virtual machines you need a Remote Desktop Protocol (RDP) Client and you will also need to be able to access the domain cloudapp.net where the machines are hosted. A typical connection string has the form cloudservicename.cloudapp.net:xxxxx, where the cloud service name is specific to a group of virtual machines and xxxxx is a port in the range 41952-65535. Therefore users connecting to Azure virtual machines must be allowed to connect to the domain *.cloudapp.net throughout the port range 41952-65535. If you cannot connect, check your company firewall policies and ensure that you can connect to this domain on the required ports.