Databases
Seminary 5

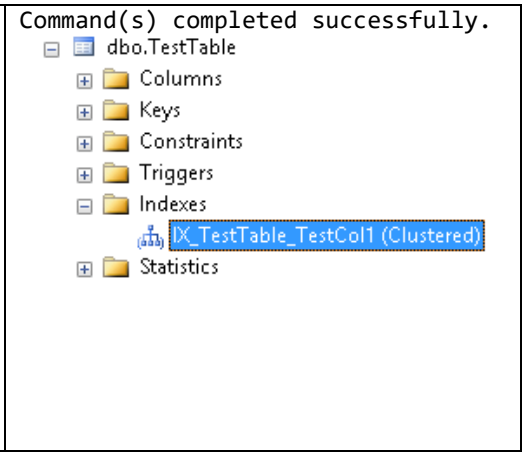
Indexes in SQL Server (I) - Examples

An index is a structure associated to a table or a view that optimize the access time to the records of the table or of the view.

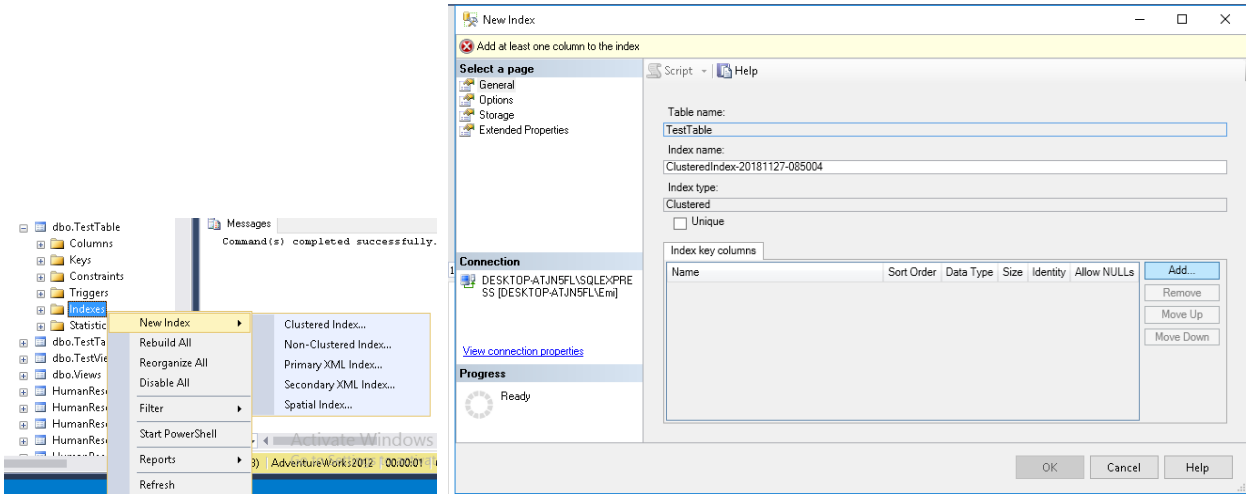
CLUSTERED INDEX - in Transact-SQL (New Query)

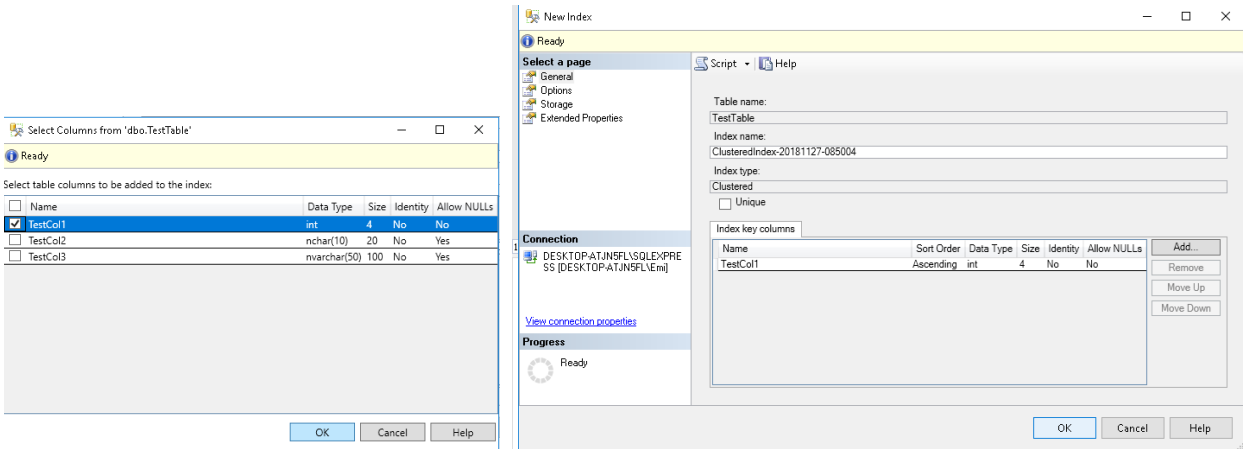
```
-- create a CLUSTERED INDEX
USE AdventureWorks2012;
GO
-- Create a new table with three columns.
CREATE TABLE dbo.TestTable
    (TestCol1 int NOT NULL,
     TestCol2 nchar(10) NULL,
     TestCol3 nvarchar(50) NULL);
GO
-- Create a clustered index called IX_TestTable_TestCol1
-- on the dbo.TestTable table using the TestCol1 column.
CREATE CLUSTERED INDEX IX_TestTable_TestCol1 ON dbo.TestTable (TestCol1)
GO
-- it could be created because we don't have a PRIMARY KEY or UNIQUE
constraint defined in the create table statement
```

```
Command(s) completed successfully.
```



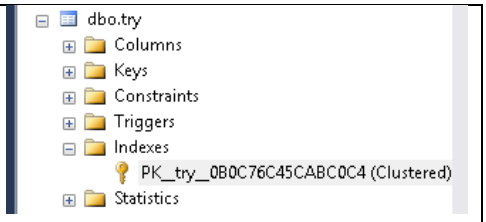
Or, with Object Explorer





Or, automatically, when the primary key is created.

```
-- create a table with primary key and directly also the clustered
index
-- drop table dbo.try
CREATE TABLE dbo.try
    (TestCol1 int PRIMARY KEY,
     TestCol2 nchar(10) NULL,
     TestCol3 nvarchar(50) NULL);
GO
-- the clustered index has been created - PK__try__0B0C76C45CABC0C4
```



Index Clustered versus Non-Clustered

Clustered Index	**Non- Clustered Index**
This will arrange the rows physically in the memory in sorted order	This will not arrange the rows physically in the memory in sorted order.
This will fast in searching for the range of values.	This will be fast in searching for the the values that are not in the range.
Index for table.	You can create a maximum of 999 non clustered indexes for table.
Leaf node of 3 tier of clustered index contains table data.	Leaf nodes of b-tree of non-clustered index contains pointers to get the contains pointers to get that contains two table data, and not the table data directly.

NON-CLUSTERED INDEXES

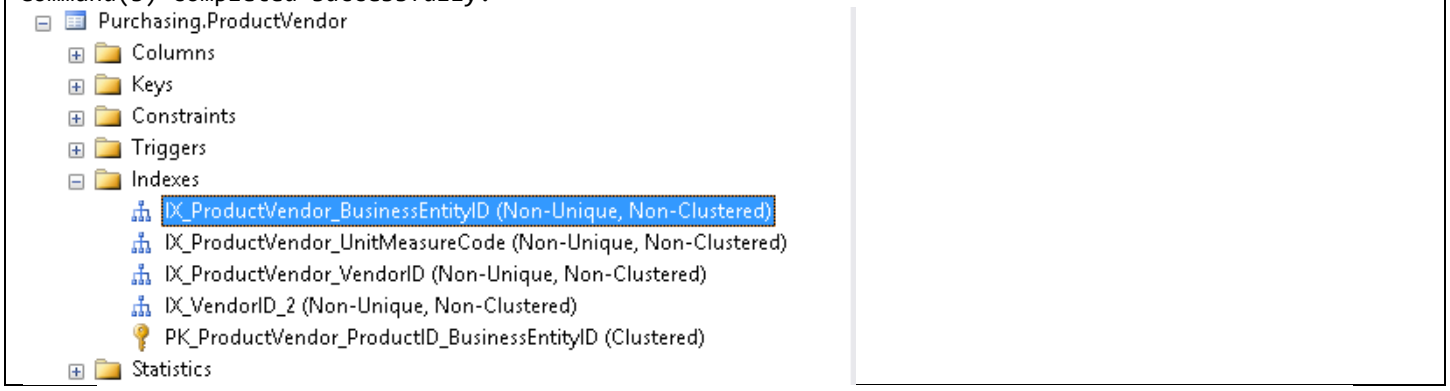
A nonclustered index = an index structure separate from the data stored in a table that reorders one or more selected columns. It helps find data more quickly than searching the underlying table. It improves the performance of frequently used queries not covered by the clustered index or to locate rows in a table without a clustered index (called a heap).

- in Transact-SQL (New Query)

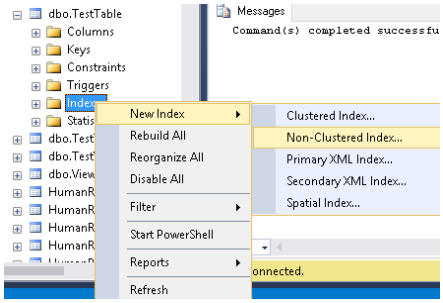
```
-- NON-CLUSTERED INDEX
USE AdventureWorks2012;
GO
-- Find an existing index named IX_ProductVendor_VendorID and delete it if found.
IF EXISTS (SELECT name FROM sys.indexes WHERE name = N'IX_ProductVendor_VendorID')
    DROP INDEX IX_ProductVendor_VendorID ON Purchasing.ProductVendor;
GO
-- Create a nonclustered index called IX_ProductVendor_VendorID
```

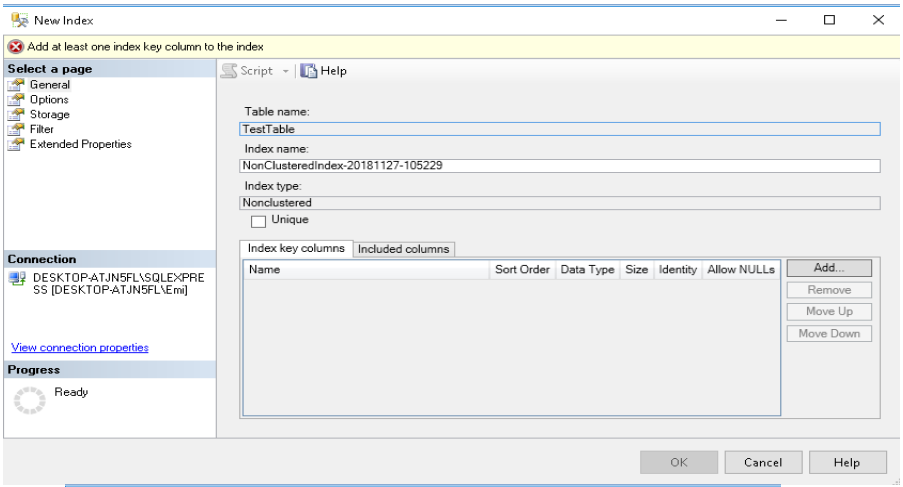
```
-- on the Purchasing.ProductVendor table using the BusinessEntityID column.
CREATE NONCLUSTERED INDEX IX_ProductVendor_VendorID  ON Purchasing.ProductVendor (BusinessEntityID);
GO
```

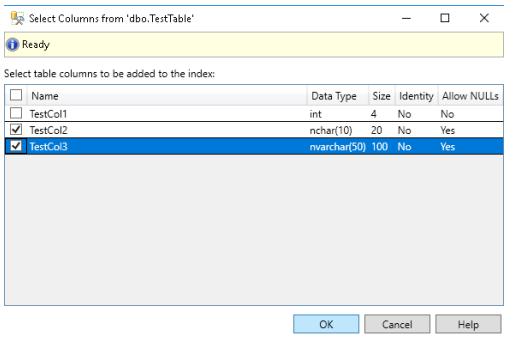
```
Command(s) completed successfully.
```

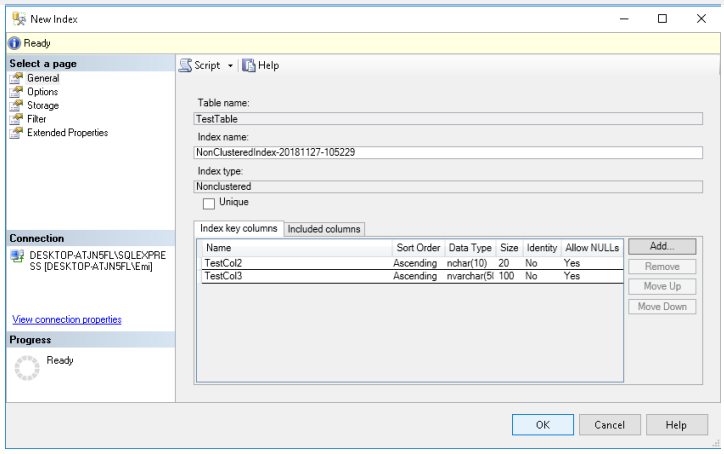


Or, with Object Explorer









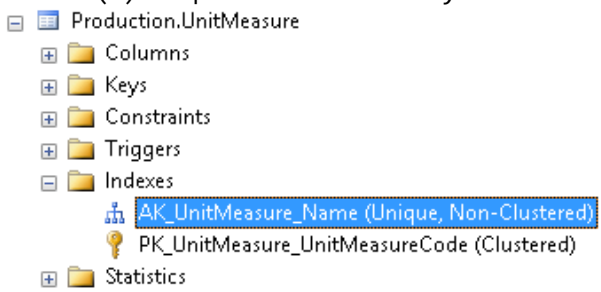
UNIQUE INDEXES

In a unique index the lookup key does not contain duplicate values. The unique index make sense in the case in which the columns from the key are unique.

```
-- UNIQUE INDEX
USE AdventureWorks2012;
```

```sql
GO
-- Find an existing index named AK_UnitMeasure_Name and delete it if found
IF EXISTS (SELECT name from sys.indexes WHERE name = N'AK_UnitMeasure_Name')
   DROP INDEX AK_UnitMeasure_Name ON Production.UnitMeasure;
GO
-- Create a unique index called AK_UnitMeasure_Name on the Production.UnitMeasure table using
the Name column.
CREATE UNIQUE INDEX AK_UnitMeasure_Name ON Production.UnitMeasure (Name);
GO
```

```
Command(s) completed successfully.
```



Primary key vs Unique key

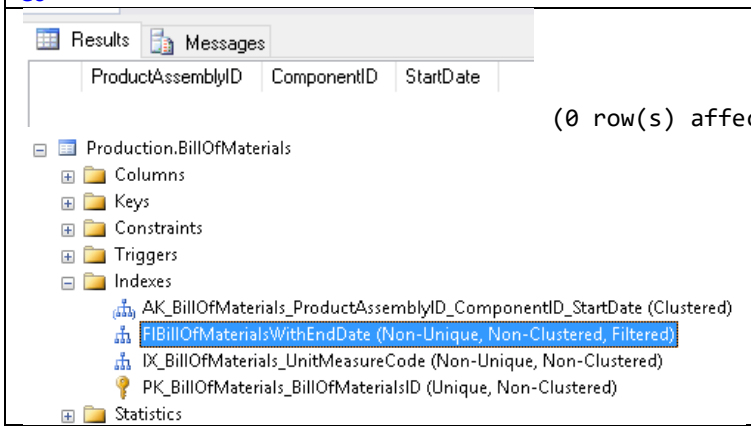
PARAMENTER	**PRIMARY KEY**	**UNIQUE KEY**
Basic	Used to serve as a unique identifier for each row in a table.	Uniquely determines a row which isn't primary key.
NULL value acceptance	Cannot accept NULL values.	Can accept one NULL value.
Number of keys that can be defined in the table	Only one primary key	More than one unique key
Index	Creates clustered index. Data in the database table is physically organized in the sequence of clustered index.	Creates non-clustered index
	Can be made foreign key into another table	Can be made foreign key into another table

Index on primary key = it is created automatically when the primary key is created = index clustered
Index on unique key = it is created automatically when the unique constraint is created = index nonclustered

<u>Filtered indexes</u>
= a non-clustered index optimized, useful for the queries that takes data from a data subset well defined.

```sql
-- FILTERED INDEX

USE AdventureWorks2012;
GO
-- Looks for an existing filtered index named "FIBillOfMaterialsWithEndDate"
-- and deletes it from the table Production.BillOfMaterials if found.
IF EXISTS (SELECT name FROM sys.indexes WHERE name = N'FIBillOfMaterialsWithEndDate'
    AND object_id = OBJECT_ID (N'Production.BillOfMaterials'))
```

```sql
DROP INDEX FIBillOfMaterialsWithEndDate ON Production.BillOfMaterials
GO
-- Creates a filtered index "FIBillOfMaterialsWithEndDate"
-- on the table Production.BillOfMaterials
-- using the columms ComponentID and StartDate.

CREATE NONCLUSTERED INDEX FIBillOfMaterialsWithEndDate ON Production.BillOfMaterials (ComponentID, StartDate)
    WHERE EndDate IS NOT NULL ;
GO
-- The filtered index above is valid for the following query. One can display the query execution plan to
determine if the query optimizer used the filtered index.

USE AdventureWorks2012;
GO
SELECT ProductAssemblyID, ComponentID, StartDate
FROM Production.BillOfMaterials
WHERE EndDate IS NOT NULL AND ComponentID = 5 AND StartDate > '01/01/2008' ;
GO
```




```
(0 row(s) affected)
```

Deactivate indexes

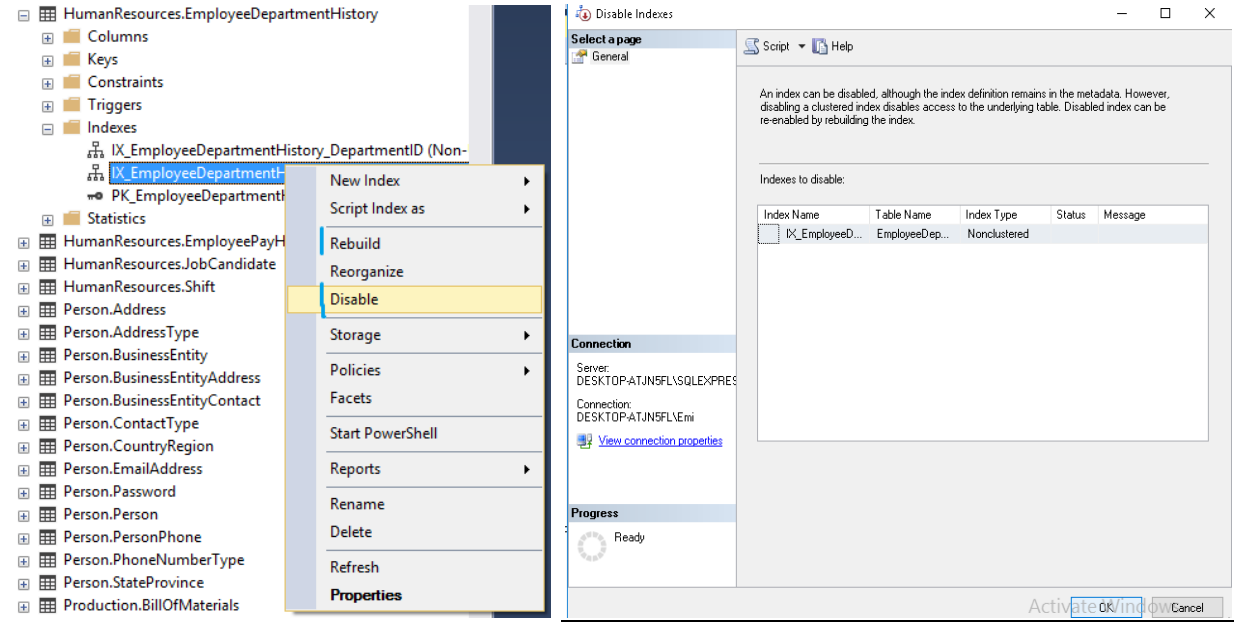
```sql
-- Deactivate indexes
ALTER INDEX IX_EmployeeDepartmentHistory_ShiftID
ON HumanResources.EmployeeDepartmentHistory DISABLE
```

```
Command(s) completed successfully.
```

Re-activate indexes

```sql
-- Re-activate indexes
ALTER INDEX IX_EmployeeDepartmentHistory_ShiftID
ON HumanResources.EmployeeDepartmentHistory REBUILD
```

```
Command(s) completed successfully.
```



EXAMPLES - other

Clustered indexes

```
-- all the indexes from the current database
select * from sys.indexes
```

Results | Messages

	object_id	name	index_id	type	type_desc	is_unique	data_space_id	ignore_dup_key	is_primary_key	is_unique_constraint	fill_factor	is_padded	is_disabled
1	3	clst	1	1	CLUSTERED	1	1	0	0	0	0	0	0
2	5	clust	1	1	CLUSTERED	1	1	0	0	0	0	0	0
3	6	clst	1	1	CLUSTERED	1	1	0	0	0	0	0	0
4	7	clust	1	1	CLUSTERED	1	1	0	0	0	0	0	0
5	7	nc	2	2	NONCLUSTERED	1	1	0	0	0	0	0	0
6	8	NULL	0	0	HEAP	0	1	0	0	0	0	0	0
7	9	clst	1	1	CLUSTERED	1	1	0	0	0	0	0	0
8	17	cl	1	1	CLUSTERED	1	1	0	0	0	0	0	0
9	17	nc	2	2	NONCLUSTERED	1	1	0	0	0	0	0	0
10	17	nc2	3	2	NONCLUSTERED	1	1	0	0	0	0	0	0

Query executed successfully. | DESKTOP-ATJN5FL\SQLEXPRESS ... | DESKTOP-ATJN5FL\Emi (53) | AdventureWorks2012 | 00:00:00 | 396 rows

```
-- drop database Seminary5_test
create database Seminary5_test
go
use Seminary5_test
go

create table Employees(
Eid int not null, EName
varchar(50), Age int)
```

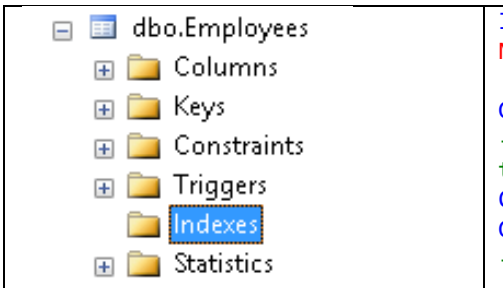
```
select * from Employees -- no index
select * from Products -- PK__Products__C5705938760ABB99 --
automatically created on Primary Key
```

Results | Messages

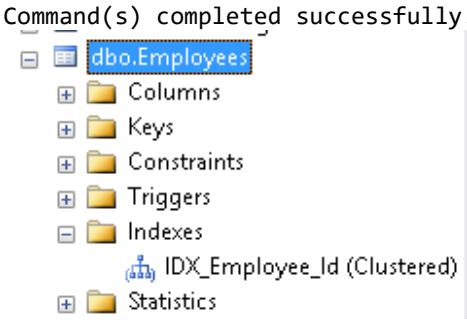
Eid	EName	Age

Pid	PName	Price	Quantity

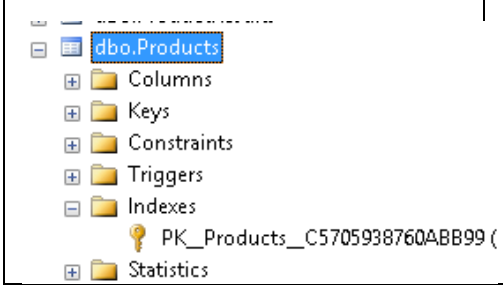
```
-- clustered index on Employees ---------------------------
-- on a posibile primary key, not null
```




```
IF EXISTS (SELECT name FROM sys.indexes WHERE name =
N'IDX_Employee_Id')
    DROP INDEX IDX_Employee_Id ON dbo.Employees;
GO
-- Create a clustered index called IDX_Employee_Id on dbo.Employees
table using the Eid column.
CREATE CLUSTERED INDEX IDX_Employee_Id  ON dbo.Employees (Eid);
GO
-- IDX_Employee_Id

Command(s) completed successfully.
```




```
----------------------------------
create table Products
(Pid int primary key,
PName varchar(50) not null,
Price float,
Quantity int)
```




```
alter table Employees
add constraint pk_E PRIMARY KEY(Eid)

insert into Employees(Eid, EName, Age) values (1,
'Andrei', 20), (2, 'Paula', 40), (3, 'Mihaela', 25)

insert into Products(Pid, PName, Price, Quantity)
values (1, 'toy car', 100, 2), (2, 'laptop', 3000, 1),
(3, 'soda water', 3, 10)

Insert into Orders(Eid, Pid, ODate) values (1, 1,
'12/12/2000'), (1,3, '4/4/2019'), (3,2, '8/8/2020')

select * from Employees
select * from Products
select * from Orders
```

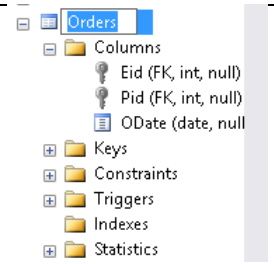
Results | Messages

	Eid	EName	Age
1	1	Andrei	20
2	2	Paula	40
3	3	Mihaela	25

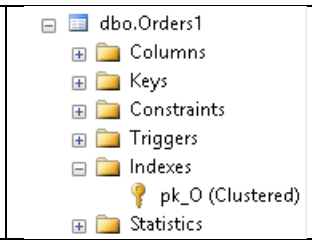
	Pid	PName	Price	Quantity
1	1	toy car	100	2
2	2	laptop	3000	1
3	3	soda ...	3	10

	Eid	Pid	ODate
1	1	1	2000-12-12
2	1	3	2019-04-04
3	3	2	2020-08-08

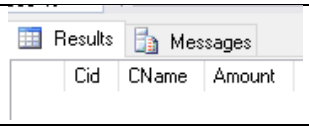
```
create table Orders(
Eid int foreign key references Employees(Eid),
Pid int foreign key references Products(Pid),
ODate date )
-- no index

-- drop table Orders
```



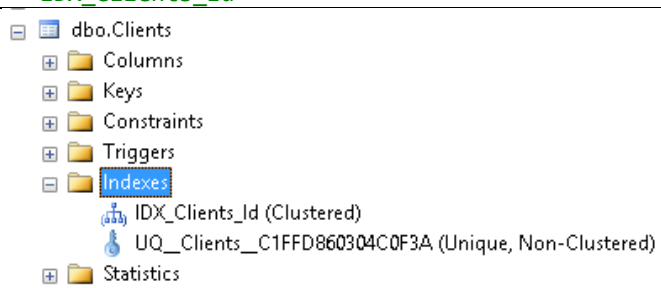
```sql
create table Orders(
Eid int foreign key references Employees(Eid),
Pid int foreign key references Products(Pid),
ODate date
constraint pk_O primary key(Eid, Pid))

select * from Orders1 -- index pk_O
```




```sql
-- on a unique column
create table Clients(Cid int unique, CName varchar(50), Amount float)
select * from Clients
-- UQ__Clients__C1FFD860304C0F3A
```




```sql
IF EXISTS (SELECT name FROM sys.indexes WHERE name = N'IDX_Clients_Id')
    DROP INDEX IDX_Clients_Id ON dbo.Clients;
GO
-- Create a clustered index called IDX_Clients_Id on dbo.Clients table using the Cid column.
CREATE CLUSTERED INDEX IDX_Clients_Id ON dbo.Clients (Cid);
GO
-- IDX_Clients_Id
```

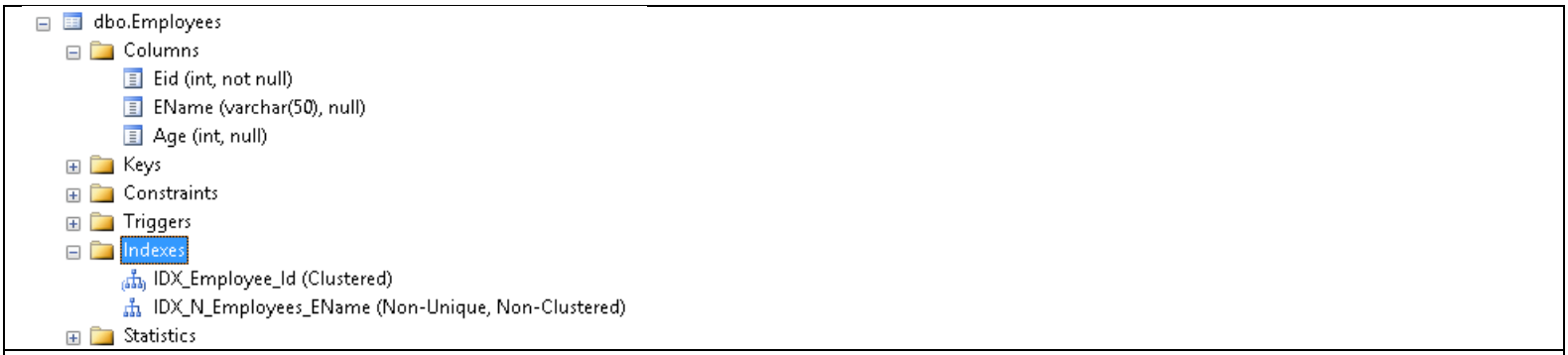



```sql
-- so, the th unique column Cid has a unique index it and also a clustered index, but it is not a
primary key
```

Non-clustered indexes

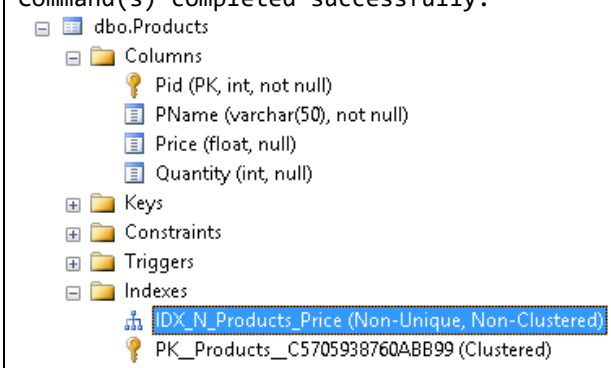
```sql
-- non-clustered index----------------------------------------
-- Find an existing index named IDX_N_Employees_EName and delete it if found.
IF EXISTS (SELECT name FROM sys.indexes WHERE name = N'IDX_N_Employees_EName')
    DROP INDEX IDX_N_Employees_EName ON dbo.Employees;
GO
-- Create a nonclustered index called IDX_N_Employees_EName on the dbo.Employees table using the EName column.
CREATE NONCLUSTERED INDEX IDX_N_Employees_EName ON dbo.Employees (EName);
GO
-- IDX_N_Employees_EName
```

```
Command(s) completed successfully.
```




```sql
-- Find an existing index named IDX_N_Products_Price and delete it if found.
IF EXISTS (SELECT name FROM sys.indexes WHERE name = N'IDX_N_Products_Price')
    DROP INDEX IDX_N_Products_Price ON dbo.Products;
GO
-- Create a nonclustered index called IDX_N_Products_Price on the dbo.Products table using the Price column.
CREATE NONCLUSTERED INDEX IDX_N_Products_Price ON dbo.Products (Price);
GO
```

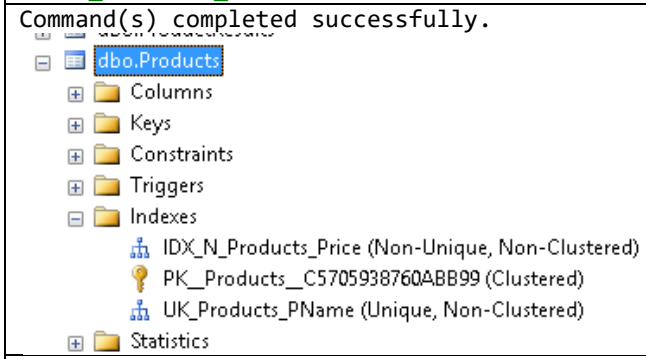
```
Command(s) completed successfully.
```



Unique indexes

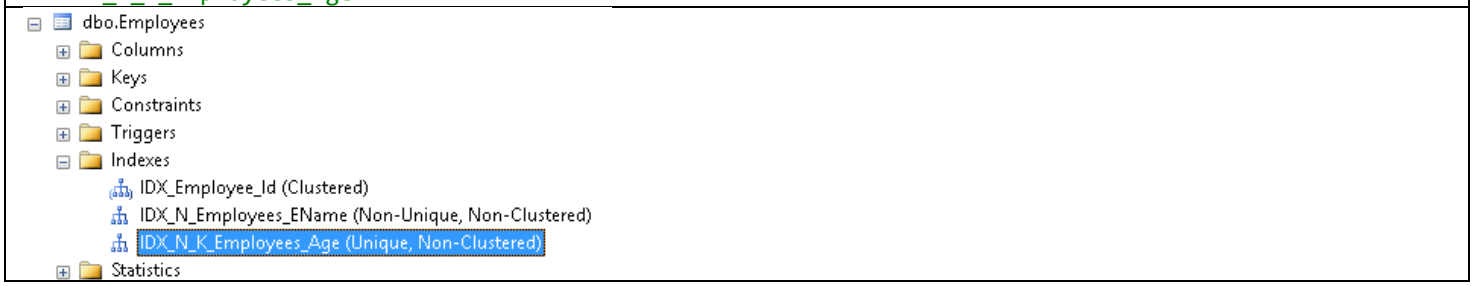
```sql
-- unique index-------------------------------

-- Find an existing index named UK_Products_PName and delete it if found
IF EXISTS (SELECT name from sys.indexes WHERE name = N'UK_Products_PName')
   DROP INDEX UK_Products_PName ON dbo.Products;
GO
-- Create a unique index called UK_Products_PName on the dbo.Products table using the PName column.
CREATE UNIQUE INDEX UK_Products_PName ON dbo.Products(PName);
GO
-- UK_Products_PName
```

```
Command(s) completed successfully.
```



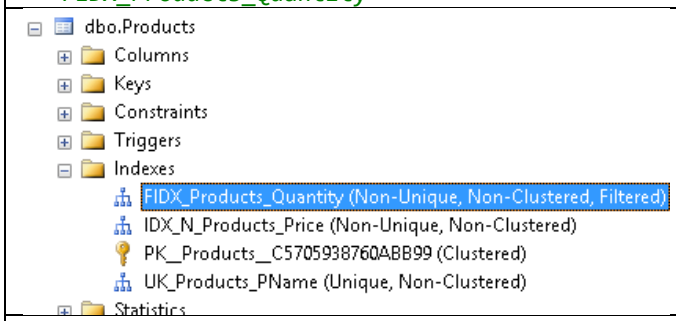

```sql
-- NON-CLUSTERED UNIQUE INDEX
```

```sql
-- Find an existing index named IDX_N_K_Employees_Age and delete it if found
IF EXISTS (SELECT name from sys.indexes WHERE name = N'IDX_N_K_Employees_Age')
   DROP INDEX IDX_N_K_Employees_Age ON dbo.Employees;
GO
-- Create a unique index called IDX_N_K_Employees_Age on the dbo.dbo.Employees table using the Age
column.
CREATE UNIQUE NONCLUSTERED INDEX IDX_N_K_Employees_Age ON dbo.Employees(Age);
GO
-- IDX_N_K_Employees_Age
```

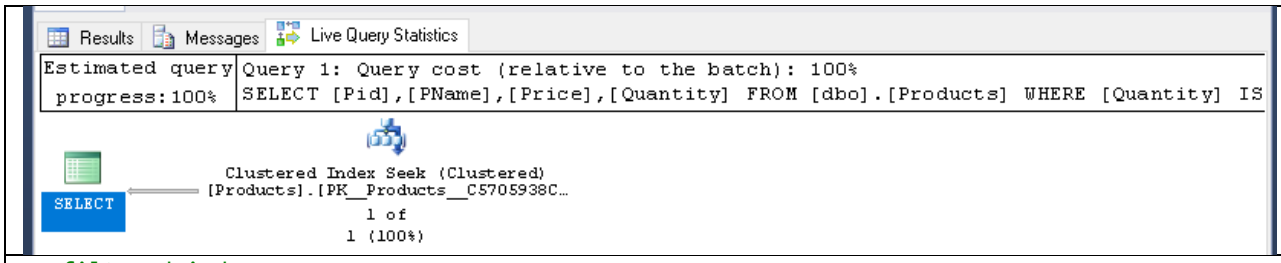


Filtered indexes

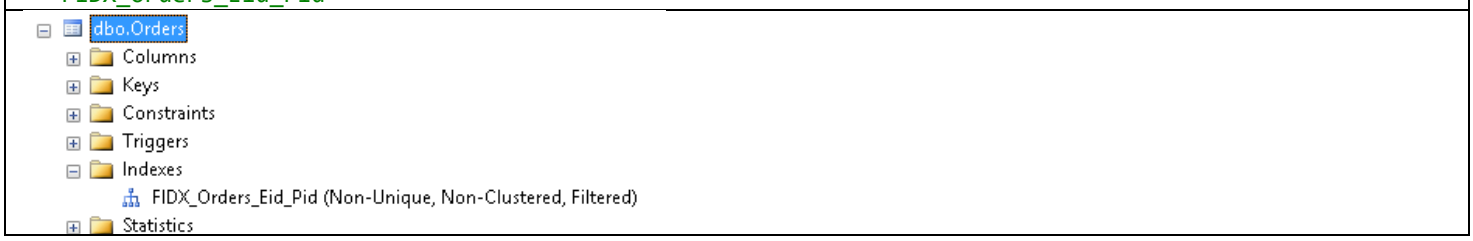
```sql
-- Looks for an existing filtered index named "FIDX_Products_Quantity"  and deletes it from the table
dbo.Products if found.
IF EXISTS (SELECT name FROM sys.indexes WHERE name = N'FIDX_Products_Quantity'
    AND object_id = OBJECT_ID (N'dbo.Products'))
DROP INDEX FIDX_Products_Quantity ON dbo.Products
GO
-- Creates a filtered index "FIDX_Products_Quantity" on the table dbo.Products using the columms
ComponentID and StartDate.

CREATE NONCLUSTERED INDEX FIDX_Products_Quantity ON dbo.Products(Pid, Quantity)
    WHERE Quantity IS NOT NULL ;
GO
-- FIDX_Products_Quantity
```



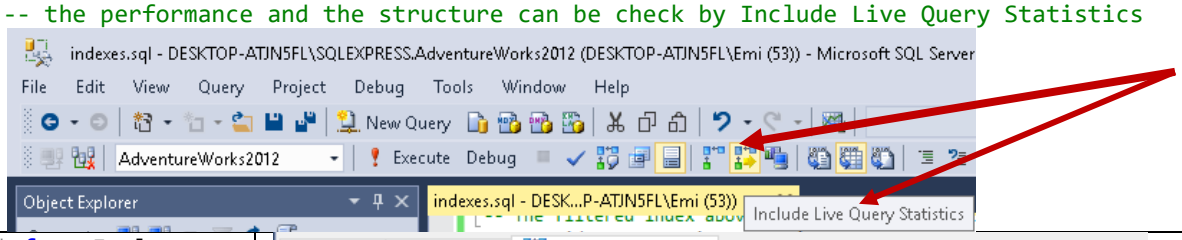

```sql
-- The filtered index above is valid for the following query. One can display the query execution plan
to determine if the query optimizer used the filtered index.
SELECT Pid, PName, Price, Quantity
FROM dbo.Products
WHERE Quantity IS NOT NULL AND Pid = 1 -- AND Quantity > 4
GO
-- Include Live Query Statistics
```



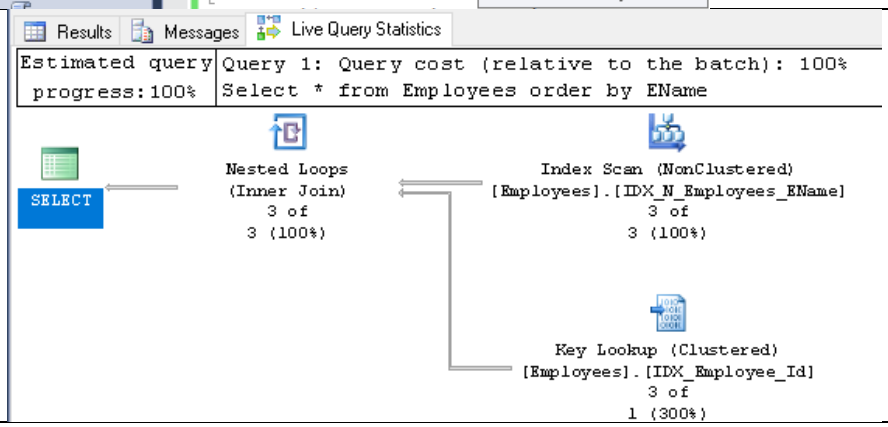

```
-- filtered index
-- Looks for an existing filtered index named "FIDX_Products_Quantity"  and deletes it from the table
dbo.Orders if found.
IF EXISTS (SELECT name FROM sys.indexes WHERE name = N'FIDX_Orders_Eid_Pid' AND object_id = OBJECT_ID
(N'dbo.Orders'))
DROP INDEX FIDX_Orders_Eid_Pid ON dbo.Orders
GO
-- Creates a filtered index "FIDX_Orders_Eid_Pid" on the table dbo.Products using the columms
ComponentID and StartDate.
CREATE NONCLUSTERED INDEX FIDX_Orders_Eid_Pid ON Orders(Eid, Pid) WHERE ODate IS NOT NULL
-- FIDX_Orders_Eid_Pid
```




```
-- the indexes are used espacially in queries with ORDER BY
-- the performance and the structure can be check by Include Live Query Statistics
```




```
Select * from Employees
order by EName
```



Index Scan (NonClustered)

Scan a nonclustered index, entirely or only a range.

Estimated operator progress: 100%

Physical Operation	Index Scan
Logical Operation	Index Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Number of Rows Read	3
Actual Number of Rows	3
Actual Number of Batches	0
Estimated I/O Cost	0.003125
Estimated Operator Cost	0.0032853 (48%)
Estimated Subtree Cost	0.0032853
Estimated CPU Cost	0.0001603
Estimated Number of Executions	1
Number of Executions	1
Estimated Number of Rows	3
Estimated Row Size	44 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	1

Object
[Seminary5_test].[dbo].[Employees].[IDX_N_Employees_EName]

Output List
Uniq1001, [Seminary5_test].[dbo].[Employees].Eid, [Seminary5_test].[dbo].[Employees].EName

Key Lookup (Clustered)

Uses a supplied clustering key to lookup on a table that has a clustered index.

Estimated operator progress: 100%

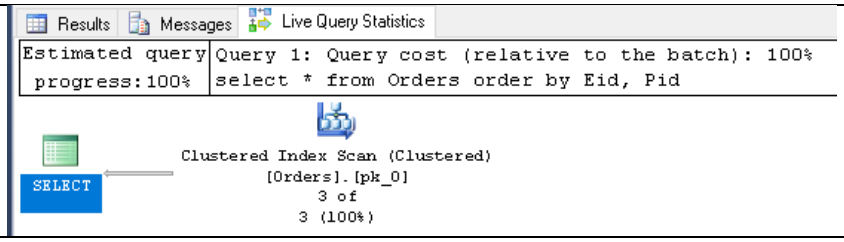
Physical Operation	Key Lookup
Logical Operation	Key Lookup
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Number of Rows Read	3
Actual Number of Rows	3
Actual Number of Batches	0
Estimated I/O Cost	0.003125
Estimated Operator Cost	0.0035993 (52%)
Estimated Subtree Cost	0.0035993
Estimated CPU Cost	0.0001581
Estimated Number of Executions	3
Number of Executions	3
Estimated Number of Rows	1
Estimated Row Size	11 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	3

Object
[Seminary5_test].[dbo].[Employees].[IDX_Employee_Id]

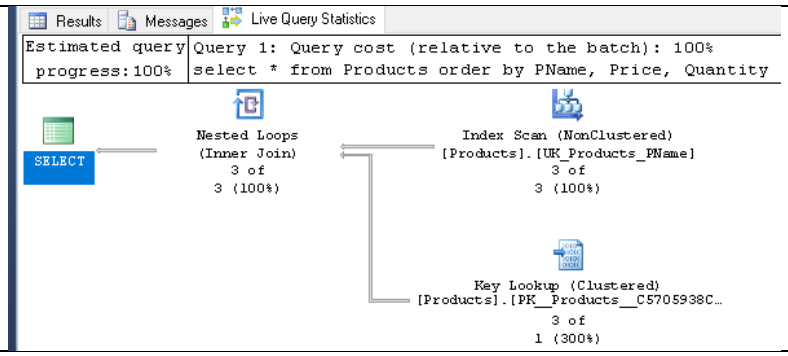
Output List
[Seminary5_test].[dbo].[Employees].Age

Seek Predicates
Seek Keys[1]: Prefix: [Seminary5_test].[dbo].[Employees].Eid, Uniq1001 = Scalar Operator ([Seminary5_test].[dbo].[Employees].[Eid]), Scalar Operator([Uniq1001])

```
select * from Orders
order by Eid, Pid
```




```
select * from Products
order by PName, Price, Quantity
```



Index Scan (NonClustered)

Scan a nonclustered index, entirely or only a range.

Estimated operator progress: 100%

Physical Operation	Index Scan
Logical Operation	Index Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Number of Rows Read	3
Actual Number of Rows	3
Actual Number of Batches	0
Estimated I/O Cost	0.003125
Estimated Operator Cost	0.0032853 (48%)
Estimated Subtree Cost	0.0032853
Estimated CPU Cost	0.0001603
Estimated Number of Executions	1
Number of Executions	1
Estimated Number of Rows	3
Estimated Row Size	40 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	1

Object
[Seminary5_test].[dbo].[Products].[UK_Products_PName]

Output List
[Seminary5_test].[dbo].[Products].Pid, [Seminary5_test].[dbo].[Products].PName

Key Lookup (Clustered)

Uses a supplied clustering key to lookup on a table that has a clustered index.

Estimated operator progress: 100%

Physical Operation	Key Lookup
Logical Operation	Key Lookup
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Number of Rows Read	3
Actual Number of Rows	3
Actual Number of Batches	0
Estimated I/O Cost	0.003125
Estimated Operator Cost	0.0035993 (52%)
Estimated Subtree Cost	0.0035993
Estimated CPU Cost	0.0001581
Estimated Number of Executions	3
Number of Executions	3
Estimated Number of Rows	1
Estimated Row Size	19 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	3

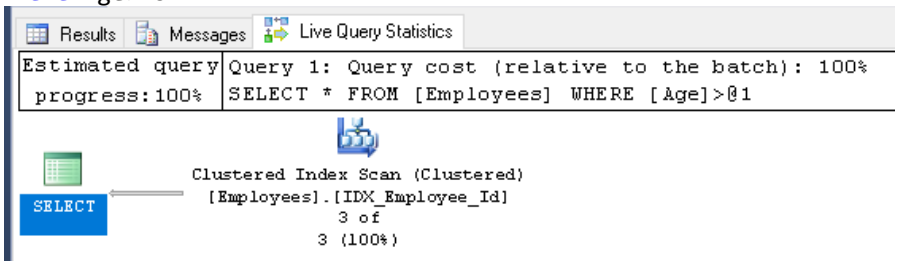
Object
[Seminary5_test].[dbo].[Products].[PK__Products__C5705938CE203326]

Output List
[Seminary5_test].[dbo].[Products].Price, [Seminary5_test].[dbo].[Products].Quantity

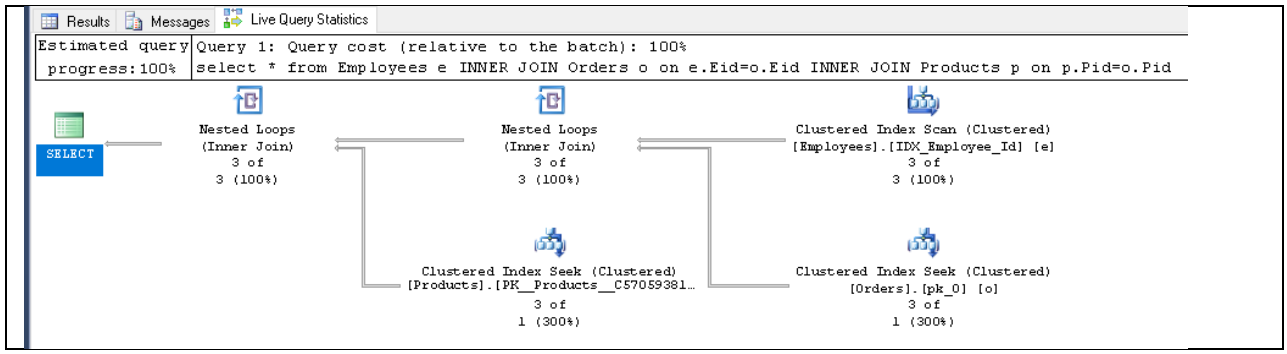
Seek Predicates
Seek Keys[1]: Prefix: [Seminary5_test].[dbo].[Products].Pid = Scalar Operator([Seminary5_test].[dbo].[Products].[Pid])

Also, the indexes can be checked in WHERE clauses and JOIN clauses.

```
-- the indexes are used especially in queries with WHERE
Select * from Employees
Where Age>18
```




```
-- the indexes are used especially in queries with JOIN
select * from Employees e INNER JOIN Orders o on e.Eid=o.Eid
INNER JOIN Products p on p.Pid=o.Pid
```



References:

https://docs.microsoft.com/en-us/sql/relational-databases/indexes/create-clustered-indexes?view=sql-server-2017
https://docs.microsoft.com/en-us/sql/relational-databases/indexes/create-nonclustered-indexes?view=sql-server-2017
https://docs.microsoft.com/en-us/sql/relational-databases/indexes/create-unique-indexes?view=sql-server-2017
https://docs.microsoft.com/en-us/sql/t-sql/statements/create-index-transact-sql?view=sql-server-2017
https://docs.microsoft.com/en-us/sql/relational-databases/indexes/create-filtered-indexes?view=sql-server-2017
https://www.geeksforgeeks.org/difference-between-primary-key-and-unique-key/
http://www.differencebetween.net/technology/difference-between-primary-key-and-unique-key/
https://codingsight.com/the-difference-between-primary-key-and-unique-key/