Laboratory 5

CRUD Operations

Create =Create new table or INSERT

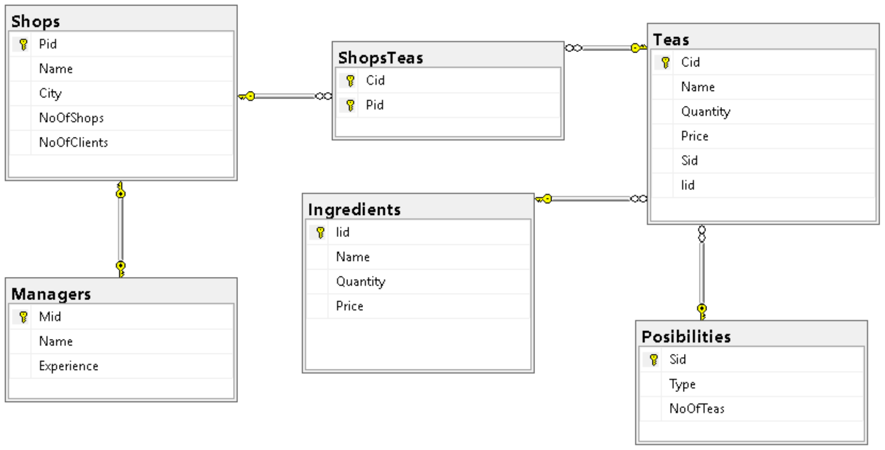
Read=SELECT

Update=UPDATE

Delete=DELETE

You have to create stored procedures for CRUD operations for 5 tables (5 stored procedures for CRUD on each table and a main procedure, or 4 operations*5 tables =20 stored procedures, or …).

You will have to consider your own database and work on it. Here, we consider the database



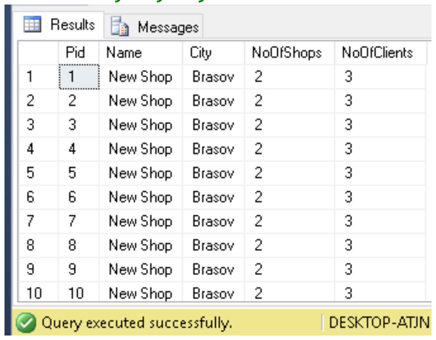
We can choose tables Shops, ShopsTeas, Teas, Ingredients and Posibilities.

The way that you implement your **stored procedures** is up to you!!!

1. An example of crud operation on a table can be (similar for Shops, Ingredients, Posibilities – these tables have only primary keys, no foreign keys)

```
USE [Example_Lab1]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[CRUD_Shops]
       @table_name Varchar(50),
       @name varchar(50),
       @city varchar(50),
       @nos int,
       @noc int,
       @noOfRows int
AS
BEGIN
       SET NOCOUNT ON;
       -- verify the parameters - at least one from the list -
with the help of a scalar function or a stored procedure with
output parameter

       -- CREATE=INSERT
       declare @n int =1
       -- we add as many rows as the parameter indicate us
       -- not all the fields must be given as parameters
       while @n<=@noOfRows begin
        insert into Shops(Name, City, NoOfShops, NoOfClients)
              Values(@name, @city, @nos, @noc)
        set @n=@n+1
       end
```

```
-- execute
EXEC CRUD_Shops 'Shops', 'New Shop',
'Brasov', 2, 3, 10
```

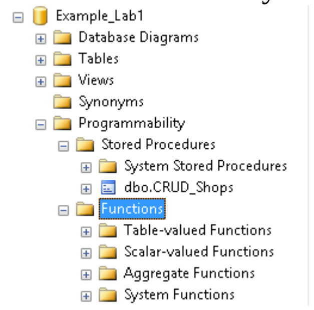


	Pid	Name	City	NoOfShops	NoOfClients
1	1	New Shop	Brasov	2	3
2	2	New Shop	Brasov	2	3
3	3	New Shop	Brasov	2	3
4	4	New Shop	Brasov	2	3
5	5	New Shop	Brasov	2	3
6	6	New Shop	Brasov	2	3
7	7	New Shop	Brasov	2	3
8	8	New Shop	Brasov	2	3
9	9	New Shop	Brasov	2	3
10	10	New Shop	Brasov	2	3

Query executed successfully. DESKTOP-ATJN

```
        -- READ=SELECT
        select * from Shops

        -- UPDATE
        update Shops set City='Cluj-Napoca' where NoOfClients>10
and Name LIKE 'A%'

        -- DELETE
        delete from Shops where NoOfClients=0

        print 'CRUD operations for table ' + @table_name
END
```

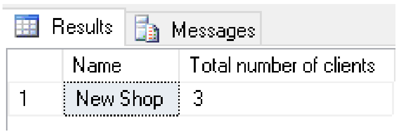
For the instruction SELECT, one can also use a **function** that returns a table. There are also scalar functions that can be used for verify the parameters.



- *inline Table-Valued Functions – RETURNS a TABLE*

```
USE Example_Lab1
GO
IF OBJECT_ID (N'ShopsF', N'IF') IS NOT NULL
    DROP FUNCTION ShopsF;
GO
CREATE FUNCTION ShopsF (@Pid int)
RETURNS TABLE
AS
RETURN
(
    SELECT Name, SUM(NoOfClients) AS 'Total number of clients'
    FROM Shops
    WHERE Pid=@Pid
    GROUP BY Name
);
GO
```

```
-- EXECUTE
SELECT * FROM ShopsF (2);
```



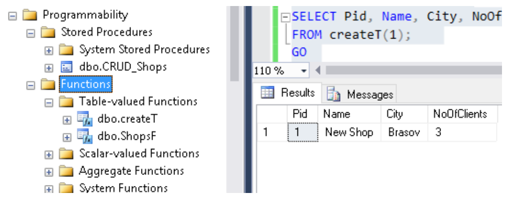
	Name	Total number of clients
1	New Shop	3

Return the Name and the Number of Clients for a Shop given with ID

- *multi-statement Table-Valued Functions – RETURNS a TABLE*

```
USE Example_Lab1
IF OBJECT_ID (N'dbo.createT', N'TF') IS NOT NULL
    DROP FUNCTION dbo.createT;
GO
CREATE FUNCTION dbo.createT (@Pid int)
RETURNS @rett TABLE
(
    Pid int primary key,
    Name varchar(50) NOT NULL,
    City varchar(50) NOT NULL,
    NoOfClients int
)
--Returns a result set that lists the shops with the Pid given
AS
BEGIN
WITH IntermediateT(Pid, Name, City, NoOfClients)
-- IntermediateT - the Intermediate table for Shops -  name and
columns
```

```
-- Example execute
SELECT Pid, Name, City, NoOfClients
FROM createT(1);
GO
```



```
SELECT Pid, Name, City, NoOf
FROM createT(1);
GO
```

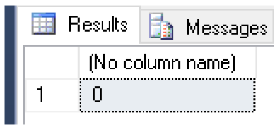
	Pid	Name	City	NoOfClients
1	1	New Shop	Brasov	3

```
    AS (
        -- Get the initial list of Shops
        SELECT Pid, Name, City, NoOfClients
        FROM Shops
        WHERE Pid=@Pid
       )
-- copy the required columns to the result of the function
   INSERT @rett
   SELECT Pid, Name, City, NoOfClients
   FROM IntermediateT
   RETURN
END;
GO
```

- *Scalar-Valued Functions*

```
Use Example_Lab1
IF OBJECT_ID (N'dbo.Sfunction', N'FN') IS NOT NULL
    DROP FUNCTION dbo.Sfunction;
GO
-- return the Total number of clients for a given Shops Pid
CREATE FUNCTION dbo.Sfunction(@Pid int)
RETURNS int
AS
BEGIN
    DECLARE @r int;
    SELECT @r = SUM(NoOfClients)
    FROM Shops
    WHERE Pid = @Pid AND Name LIKE 'S%';
    IF (@r IS NULL) SET @r = 0;
    RETURN @r;
END;
GO
```

```
-- EXECUTION
SELECT dbo.Sfunction(2)
```



```
Results   Messages
   (No column name)
1  0
```

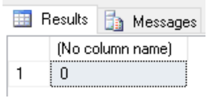
One **must create at least one function for each table** in which will verify some conditions related to a field of the table (that will appear as a parameter in the crud stored procedure)

For example: verify that the name of the Shop start with a letter, verify that the quantity for Ingredient to be positive, verify the e-mail address to respect a format, …

TestPrice – check for table Ingredients the field Price - when will be executed

```
Create function dbo.TestPrice(@p int)
RETURNS INT
AS
  BEGIN
       IF @p BETWEEN 10 AND 20 SET @p=1
       ELSE SET @p=0
       RETURN @p
END
```

```
-- execute
SELECT dbo.TestPrice(2)
```



```
Results   Messages
   (No column name)
1  0
```

--or verification like

```
if dbo.TestPrice(@p)=1
   insert into...
else print 'no insertion..'
```

Function that add a constraint with check

```
CREATE TABLE CheckTbl (col1 int, col2 int);
GO
CREATE FUNCTION CheckFnctn()
RETURNS int
AS
BEGIN
  DECLARE @retval int
  SELECT @retval = COUNT(*) FROM CheckTbl
  RETURN @retval
END;
GO
ALTER TABLE CheckTbl
ADD CONSTRAINT chkRowCount CHECK (dbo.CheckFnctn() >= 1 );
GO
```

Stored Procedure with INPUT/OUTPUT parameters

```sql
CREATE PROCEDURE test_InsertShops
@flag bit OUTPUT,-- return 0 for fail,1 for success
@Name varchar(50),
@City varchar(100),
@NoOfShops int,
@NoOfClients int
AS
BEGIN
 Insert into Shops(Name, City, NoOfShops, NoOfClients) Values(@Name, @City, @NoOfShops, @NoOfClients)
 IF @@TRANCOUNT > 0 SET @flag=1;
 ELSE SET @flag=0;
END
```

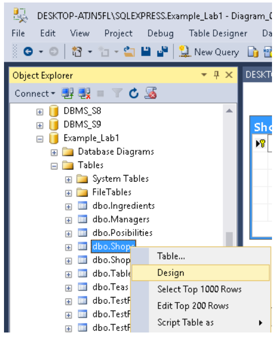
```sql
--Execute above created procedure to insert rows into table
Declare @flag bit
EXEC test_InsertShops @flag OUTPUT,'Shop 1','Bucuresti', 14, 12
if @flag=1 print 'Successfully inserted'
else print 'There is some error'
```

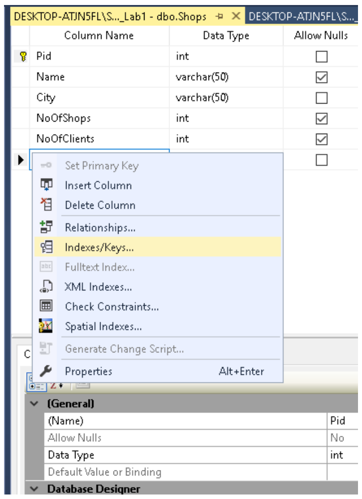
```
(1 row(s) affected)
There is some error
```

<u>Views</u> - Must be created on the tables used for the CRUD operations and be relevant.

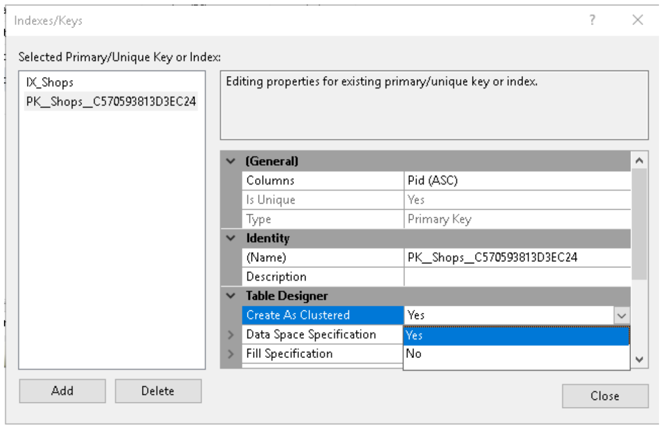
<u>Non-Clustered Indexes</u>

- *by using the Table Designer:* Choose the database -> Tables -> right click on the table used to create a non-clustered index -> Design -> Indexes/Keys -> Add -> Select the new index in the Selected Primary/Unique Key or Index text box. -> In the grid -> Create as Clustered: No (for nonclustered indexes) -> Close -> Save table




DESKTOP-ATJN5FL\S..._Lab1 - dbo.Shops

Column Name	Data Type	Allow Nulls
Pid	int	☐
Name	varchar(50)	☑
City	varchar(50)	☐
NoOfShops	int	☑
NoOfClients	int	☑

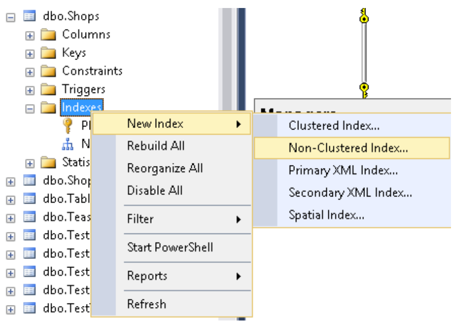

Indexes/Keys

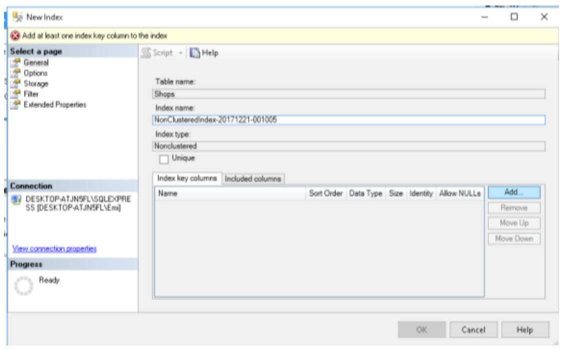
Selected Primary/Unique Key or Index:
IX_Shops
PK_Shops__C570593813D3EC24

Editing properties for existing primary/unique key or index.

(General)	
Columns	Pid (ASC)
Is Unique	Yes
Type	Primary Key
Identity	
(Name)	PK_Shops__C570593813D3EC24
Description	
Table Designer	
Create As Clustered	Yes
Data Space Specification	Yes
Fill Specification	No

- *by using Object Explorer:* Choose the database -> Tables (folder) -> expand the table that will be used to create a non-clustered index -> Right-click the Indexes folder -> New Index -> select Non-Clustered Index…



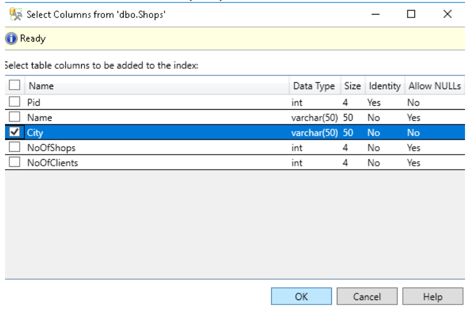

New Index

Add at least one index key column to the index

Table name: Shops
Index name: NonClusteredIndex-20171221-001005
Index type: Nonclustered
☐ Unique

Connection: DESKTOP-ATJN5FL\SQLEXPRESS [DESKTOP-ATJN5FL\Emi]

In the New Index dialog box -> General page -> Index name box (=enter the name of the new index) -> Under Index key columns -> click Add… -> In the Select Columns from table_name dialog box -> select the check box(es) of the table column(s) to be added to the nonclustered index ->Ok -> Ok.

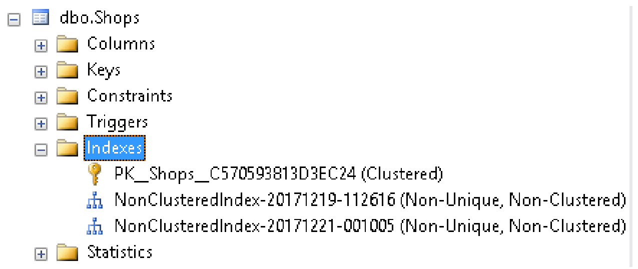


Select Columns from 'dbo.Shops'

Ready

Select table columns to be added to the index:

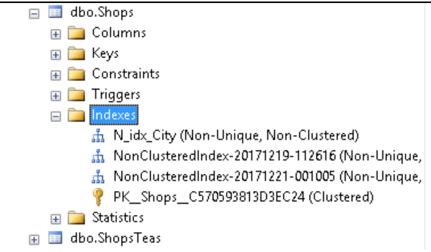
	Name	Data Type	Size	Identity	Allow NULLs
☐	Pid	int	4	Yes	No
☐	Name	varchar(50)	50	No	Yes
✓	City	varchar(50)	50	No	No
☐	NoOfShops	int	4	No	Yes
☐	NoOfClients	int	4	No	Yes



```
dbo.Shops
  Columns
  Keys
  Constraints
  Triggers
  Indexes
    PK__Shops__C570593813D3EC24 (Clustered)
    NonClusteredIndex-20171219-112616 (Non-Unique, Non-Clustered)
    NonClusteredIndex-20171221-001005 (Non-Unique, Non-Clustered)
  Statistics
```

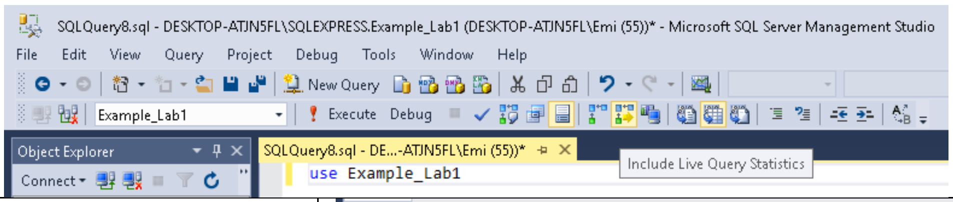
- *by using Transact-SQL:* Choose the database -> New Query -> write the code -> Execute

```sql
USE Example_Lab1
GO
-- Find an existing index named Nix_Name and delete it if found.
IF EXISTS (SELECT name FROM sys.indexes WHERE name = N'N_idx_City')
    DROP INDEX N_idx_City ON Shops;
GO
-- Create a nonclustered index called N_idx_City on the Shops table
using the City column.
CREATE NONCLUSTERED INDEX N_idx_City ON Shops (City);
GO
```



```
dbo.Shops
  Columns
  Keys
  Constraints
  Triggers
  Indexes
    N_idx_City (Non-Unique, Non-Clustered)
    NonClusteredIndex-20171219-112616 (Non-Unique,
    NonClusteredIndex-20171221-001005 (Non-Unique,
    PK__Shops__C570593813D3EC24 (Clustered)
  Statistics
dbo.ShopsTeas
```

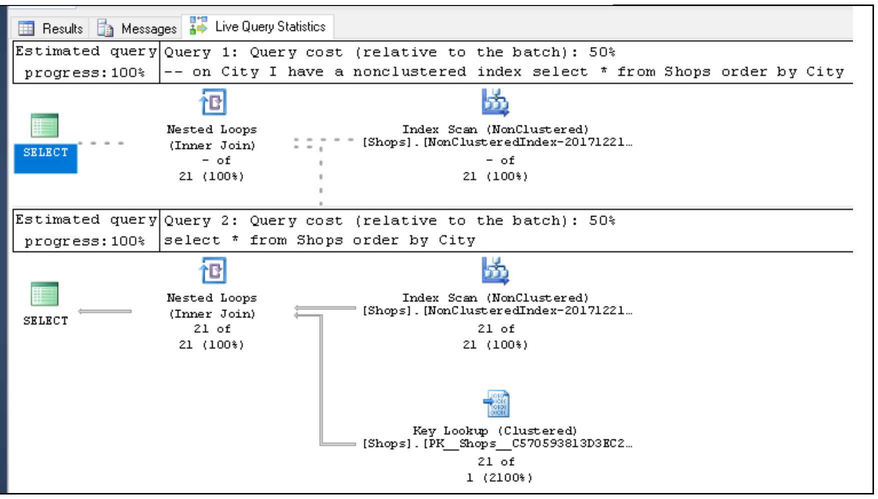
Check Clustered / Non-Clustered Indexes (Instead of Dynamic Management Views and Functions)
Check the indexes – check **Include Live Query Statistics -** when run a query.
After an update / order by / …, the order of the records is changed. For example, the indexes become 'un-ordered' (1, 2, 3, … -> 3, 1, 2, …). To choose the 'best' index, verify with the menu Include Live Query Statistics.



SQLQuery8.sql - DESKTOP-ATJN5FL\SQLEXPRESS.Example_Lab1 (DESKTOP-ATJN5FL\Emi (55))* - Microsoft SQL Server Management Studio

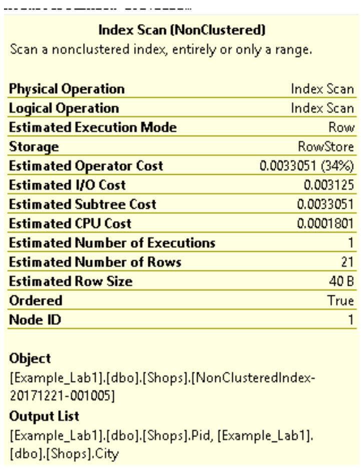
```sql
use Example_Lab1
```

```sql
-- check Include Live Query Statistics
-- on Pid (the primary key) there is a
clustered index
select * from Shops order by Pid

-- on City I have a nonclustered index
select * from Shops order by City
```



```
Estimated query progress:100%   Query 1: Query cost (relative to the batch): 50%
                                -- on City I have a nonclustered index select * from Shops order by City

SELECT    Nested Loops (Inner Join)   Index Scan (NonClustered)
          - of                        [Shops].[NonClusteredIndex-20171221…
          21 (100%)                   - of
                                      21 (100%)

Estimated query progress:100%   Query 2: Query cost (relative to the batch): 50%
                                select * from Shops order by City

SELECT    Nested Loops (Inner Join)   Index Scan (NonClustered)
          21 of                       [Shops].[NonClusteredIndex-20171221…
          21 (100%)                   21 of
                                      21 (100%)

                                      Key Lookup (Clustered)
                                      [Shops].[PK__Shops__C570593813D3EC2…
                                      21 of
                                      1 (2100%)
```

By moving the mouse through the indexes, one can check the properties…

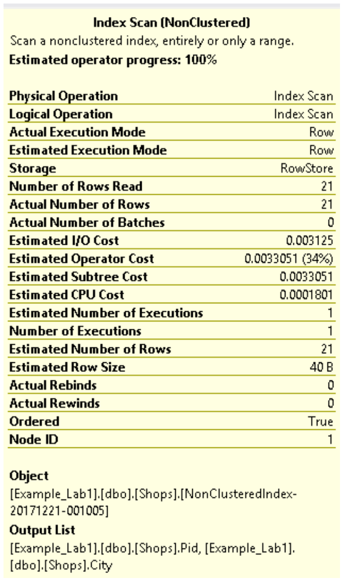


Index Scan (NonClustered)

Scan a nonclustered index, entirely or only a range.

Physical Operation	Index Scan
Logical Operation	Index Scan
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.0033051 (34%)
Estimated I/O Cost	0.003125
Estimated Subtree Cost	0.0033051
Estimated CPU Cost	0.0001801
Estimated Number of Executions	1
Estimated Number of Rows	21
Estimated Row Size	40 B
Ordered	True
Node ID	1

Object
[Example_Lab1].[dbo].[Shops].[NonClusteredIndex-20171221-001005]
Output List
[Example_Lab1].[dbo].[Shops].Pid, [Example_Lab1].[dbo].[Shops].City

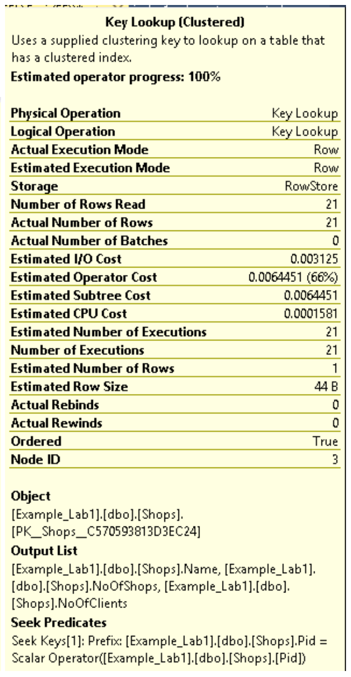


Index Scan (NonClustered)

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	21
Actual Number of Rows	21
Actual Number of Batches	0
Estimated I/O Cost	0.003125
Estimated Operator Cost	0.0033051 (34%)
Estimated Subtree Cost	0.0033051
Estimated CPU Cost	0.0001801
Estimated Number of Executions	1
Number of Executions	1
Estimated Number of Rows	21
Estimated Row Size	40 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	1

Object
[Example_Lab1].[dbo].[Shops].[NonClusteredIndex-20171221-001005]
Output List
[Example_Lab1].[dbo].[Shops].Pid, [Example_Lab1].[dbo].[Shops].City



Key Lookup (Clustered)

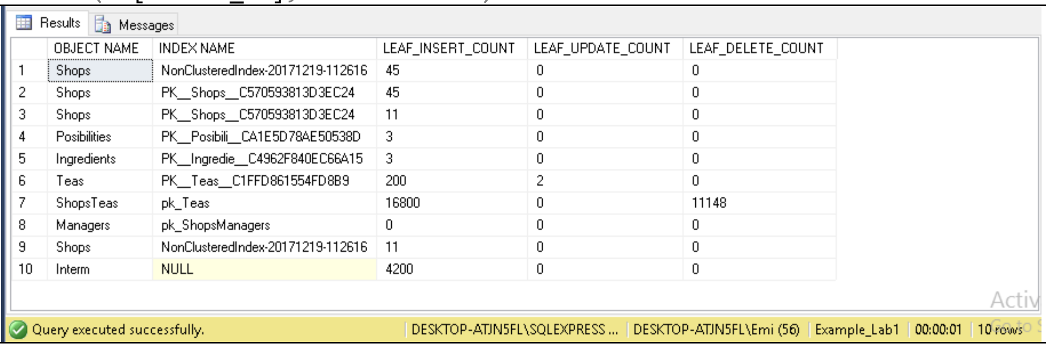
Uses a supplied clustering key to lookup on a table that has a clustered index.
Estimated operator progress: 100%

Physical Operation	Key Lookup
Logical Operation	Key Lookup
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Number of Rows Read	21
Actual Number of Rows	21
Actual Number of Batches	0
Estimated I/O Cost	0.003125
Estimated Operator Cost	0.0064451 (66%)
Estimated Subtree Cost	0.0064451
Estimated CPU Cost	0.0001581
Estimated Number of Executions	21
Number of Executions	21
Estimated Number of Rows	1
Estimated Row Size	44 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	3

Object
[Example_Lab1].[dbo].[Shops].[PK__Shops__C570593813D3EC24]
Output List
[Example_Lab1].[dbo].[Shops].Name, [Example_Lab1].[dbo].[Shops].NoOfShops, [Example_Lab1].[dbo].[Shops].NoOfClients
Seek Predicates
Seek Keys[1]: Prefix: [Example_Lab1].[dbo].[Shops].Pid = Scalar Operator([Example_Lab1].[dbo].[Shops].[Pid])

Check all indexes and some other properties (leaf number after Insert, update, delete)

```
use Example_Lab1
GO
SELECT OBJECT_NAME(A.[OBJECT_ID]) AS [OBJECT NAME],
       I.[NAME] AS [INDEX NAME],
       A.LEAF_INSERT_COUNT,
       A.LEAF_UPDATE_COUNT,
       A.LEAF_DELETE_COUNT
FROM   SYS.DM_DB_INDEX_OPERATIONAL_STATS (NULL,NULL,NULL,NULL ) A INNER JOIN SYS.INDEXES AS I
         ON I.[OBJECT_ID] = A.[OBJECT_ID] AND I.INDEX_ID = A.INDEX_ID
WHERE  OBJECTPROPERTY(A.[OBJECT_ID],'IsUserTable') = 1
```



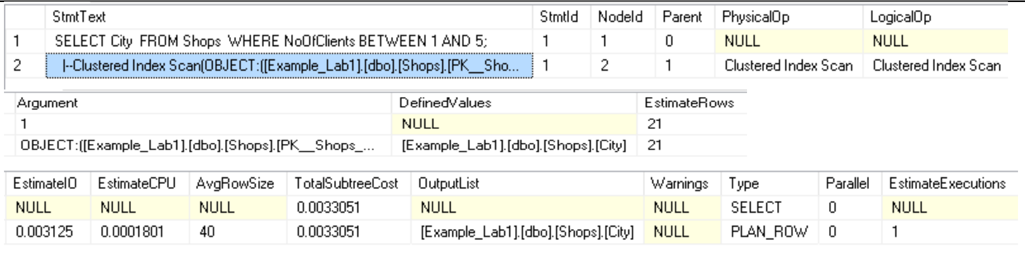
Results | Messages

	OBJECT NAME	INDEX NAME	LEAF_INSERT_COUNT	LEAF_UPDATE_COUNT	LEAF_DELETE_COUNT
1	Shops	NonClusteredIndex-20171219-112616	45	0	0
2	Shops	PK__Shops__C570593813D3EC24	45	0	0
3	Shops	PK__Shops__C570593813D3EC24	11	0	0
4	Posibilities	PK__Posibili__CA1E5D78AE50538D	3	0	0
5	Ingredients	PK__Ingredie__C4962F840EC66A15	3	0	0
6	Teas	PK__Teas__C1FFD861554FD8B9	200	2	0
7	ShopsTeas	pk_Teas	16800	0	11148
8	Managers	pk_ShopsManagers	0	0	0
9	Shops	NonClusteredIndex-20171219-112616	11	0	0
10	Interm	NULL	4200	0	0

```
Query executed successfully.   DESKTOP-ATJN5FL\SQLEXPRESS ... | DESKTOP-ATJN5FL\Emi (56) | Example_Lab1 | 00:00:01 | 10 rows
```

Show index plan

```
use Example_Lab1
go
SET NOCOUNT ON;
GO
SET SHOWPLAN_ALL ON;
GO
SELECT City
FROM Shops
WHERE NoOfClients BETWEEN 1
AND 5;
GO
SET SHOWPLAN_ALL OFF;
```



	StmtText	StmtId	NodeId	Parent	PhysicalOp	LogicalOp
1	SELECT City FROM Shops WHERE NoOfClients BETWEEN 1 AND 5;	1	1	0	NULL	NULL
2	|--Clustered Index Scan(OBJECT:([Example_Lab1].[dbo].[Shops].[PK__Sho...	1	2	1	Clustered Index Scan	Clustered Index Scan

Argument	DefinedValues	EstimateRows
1	NULL	21
OBJECT:([Example_Lab1].[dbo].[Shops].[PK__Shops_...	[Example_Lab1].[dbo].[Shops].[City]	21

EstimateIO	EstimateCPU	AvgRowSize	TotalSubtreeCost	OutputList	Warnings	Type	Parallel	EstimateExecutions
NULL	NULL	NULL	0.0033051	NULL	NULL	SELECT	0	NULL
0.003125	0.0001801	40	0.0033051	[Example_Lab1].[dbo].[Shops].[City]	NULL	PLAN_ROW	0	1