Indexing Technicals

ISM 6218

Due on October 31st

The Avengers Team

"We will avenge every problem on our way"

Aitemir Yeskenov (Team Lead)

Nagarjuna Kanneganti

Sai Suraj Argula

Vinay Kumar Reddy Baradi

Table of Contents

No index 5

PK index 5

Non PK clustered index 6

Both Clustered Index and non-clustered Index 7

Only a Non-clustered index 7

Composite index (can only be non-clustered) 8

Table Variable with an index 9

Temp table with a covering and non-covering index 10

Create a view with and without an index 14

Use a filtered index for any of the above 15

Create Execution plans using hints 16

Analyze performance of Index Choices and Choices of Hints 16

Demonstrate use of: Composite, Covering, Use, Include, With, Filter 17

Demonstrate use of: disable, rebuild, drop. 18

Requirements

Create separate tables with the following configurations:

- No index
- PK index
- Non PK clustered index
- Both Clustered Index and non-clustered Index
- Only a Non-clustered index
- Composite index (can only be non-clustered)
- Temp table with a covering and non-covering index
- Table Variable with an index
- Create a view with and without an index
- Use a filtered index for any of the above
- Create Execution plans using hints
- Analyze performance of Index Choices and Choices of Hints
- Demonstrate use of: Composite, Covering, Use, Include, With, Filter
- Demonstrate use of: disable, rebuild, drop.
- You must run queries with execution plans showing the difference between index and disabled index performance.

Deliverables:

You must design and execute a series of experiments covering the above requirements and report the results in your team's lab book.

You will need to upload MDF and LDF files and your report.

Your experiments need to identify a stakeholder, user story and use case.

Business Process Supported

We have created the Employee database which consists of the employee details such as EMployeeId, EmpoyeeFirstname, EmployeeLastName, ZipCode.

<u>USER STORY</u>:

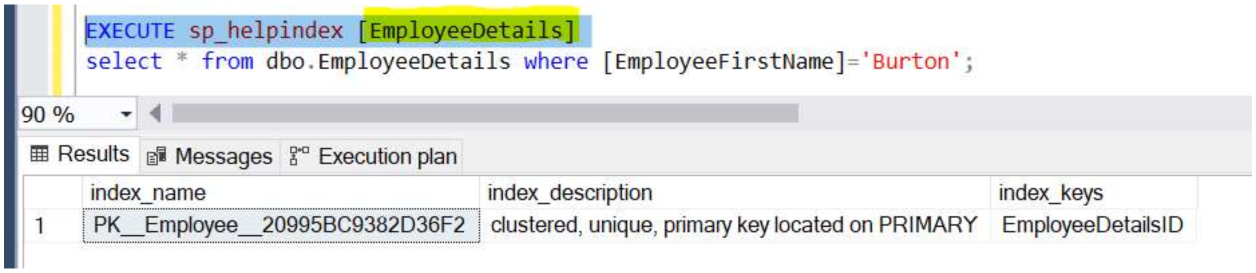
HR of the company XYZ wants to know the details of the Employee with FirstName ‘Burton’ .

No index

Here first we created the simple Employee table [EmployeeDetails] which stores the details of the employees.

Syntax:

```
CREATE TABLE [dbo].[EmployeeDetails](
        [EmployeeId] [int] NOT NULL,
        [EmployeeFirstName] [varchar](50) NULL,
        [EmployeeLastName] [varchar](50) NULL,
        [EmployeeZip] [varchar](50) NULL
)
```



```
EXECUTE sp_helpindex [EmployeeDetails]
select * from dbo.EmployeeDetails where [EmployeeFirstName]='Burton';
```

	index_name	index_description	index_keys
1	PK__Employee__20995BC9382D36F2	clustered, unique, primary key located on PRIMARY	EmployeeDetailsID

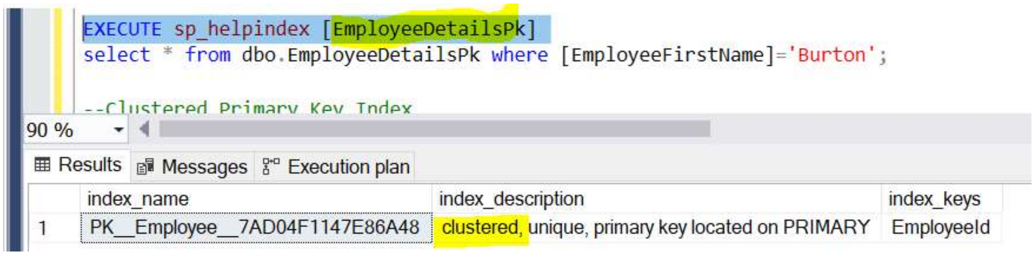
PK index

Here we created a table EmployeeDetailsPk with the EmployeeId as the primary key column.

Also the clustered index is automatically created on the primary key column in the table.

Syntax:

```
CREATE TABLE [dbo].[EmployeeDetailsPk](
        [EmployeeId] [int] NOT NULL,
        [EmployeeFirstName] [varchar](50) NULL,
        [EmployeeLastName] [varchar](50) NULL,
        [EmployeeZip] [varchar](50) NULL
        Primary key ([EmployeeId])
```



```
EXECUTE sp_helpindex [EmployeeDetailsPk]
select * from dbo.EmployeeDetailsPk where [EmployeeFirstName]='Burton';

--Clustered Primary Key Index
```

90 %

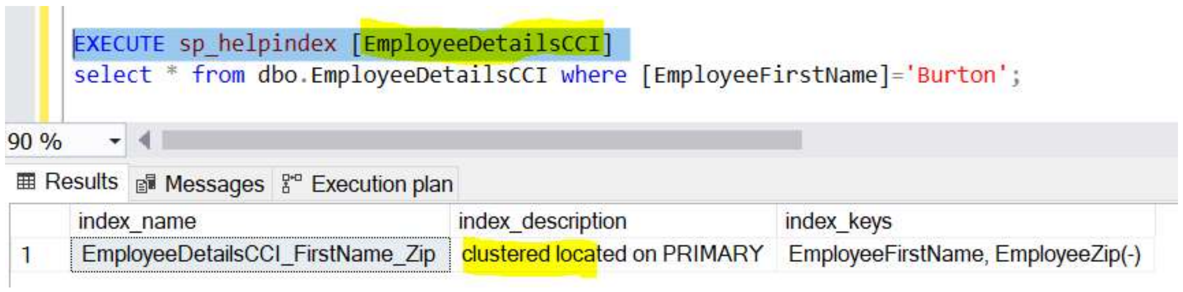
Results | Messages | Execution plan

	index_name	index_description	index_keys
1	PK__Employee__7AD04F1147E86A48	clustered, unique, primary key located on PRIMARY	EmployeeId

Non PK clustered index

Syntax:

```
CREATE TABLE [dbo].[EmployeeDetailsCCI](
	[EmployeeId] [int] NOT NULL,
	[EmployeeFirstName] [varchar](50) NULL,
	[EmployeeLastName] [varchar](50) NULL,
	[EmployeeZip] [varchar](50) NULL

)


CREATE CLUSTERED INDEX EmployeeDetailsCCI_FirstName_Zip
ON [dbo].[EmployeeDetailsCCI]([EmployeeFirstName] ASC, [EmployeeZip] DESC)
```



```
EXECUTE sp_helpindex [EmployeeDetailsCCI]
select * from dbo.EmployeeDetailsCCI where [EmployeeFirstName]='Burton';
```

90 %

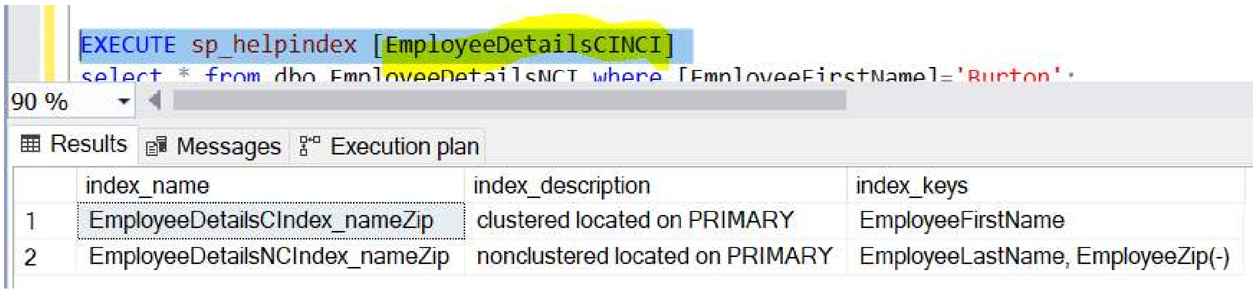
Results | Messages | Execution plan

	index_name	index_description	index_keys
1	EmployeeDetailsCCI_FirstName_Zip	clustered located on PRIMARY	EmployeeFirstName, EmployeeZip(-)

Both Clustered Index and non-clustered Index

Here we created the clustered index on the employeefirstname and non-clustered index on the employeelast name and the employeeZip.

```
CREATE TABLE [dbo].[EmployeeDetailsCINCI](
	[EmployeeId] [int] NOT NULL,
	[EmployeeFirstName] [varchar](50) NULL,
	[EmployeeLastName] [varchar](50) NULL,
	[EmployeeZip] [varchar](50) NULL

)

create NONCLUSTERED INDEX EmployeeDetailsNCIndex_nameZip
ON dbo.[EmployeeDetailsCINCI]([EmployeeLastName] ASC,[EmployeeZip] DESC)
create CLUSTERED INDEX EmployeeDetailsCIndex_nameZip
ON dbo.[EmployeeDetailsCINCI]([EmployeeFirstName] ASC)
```



```
EXECUTE sp_helpindex [EmployeeDetailsCINCI]
select * from dbo.EmployeeDetailsNCI where [EmployeeFirstName]='Burton';
```

90 %

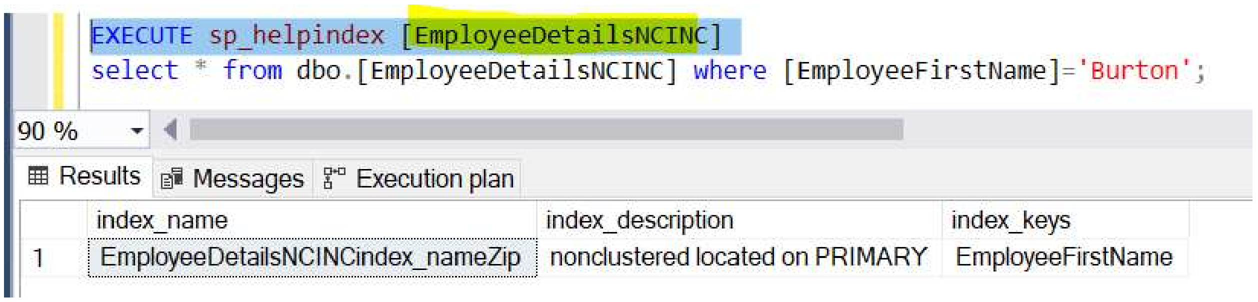
Results | Messages | Execution plan

	index_name	index_description	index_keys
1	EmployeeDetailsCIndex_nameZip	clustered located on PRIMARY	EmployeeFirstName
2	EmployeeDetailsNCIndex_nameZip	nonclustered located on PRIMARY	EmployeeLastName, EmployeeZip(-)

Only a Non-clustered index

Now we created the non-clustered index EmployeeDetailsNCINCindex_nameZip on the table.

```
CREATE TABLE [dbo].[EmployeeDetailsNCINC](
	[EmployeeId] [int] NOT NULL,
	[EmployeeFirstName] [varchar](50) NULL,
	[EmployeeLastName] [varchar](50) NULL,
	[EmployeeZip] [varchar](50) NULL

)

create NONCLUSTERED INDEX EmployeeDetailsNCINCindex_nameZip

ON dbo.EmployeeDetailsNCINC([EmployeeFirstName] ASC)
```



```
EXECUTE sp_helpindex [EmployeeDetailsNCINC]
select * from dbo.[EmployeeDetailsNCINC] where [EmployeeFirstName]='Burton';
```

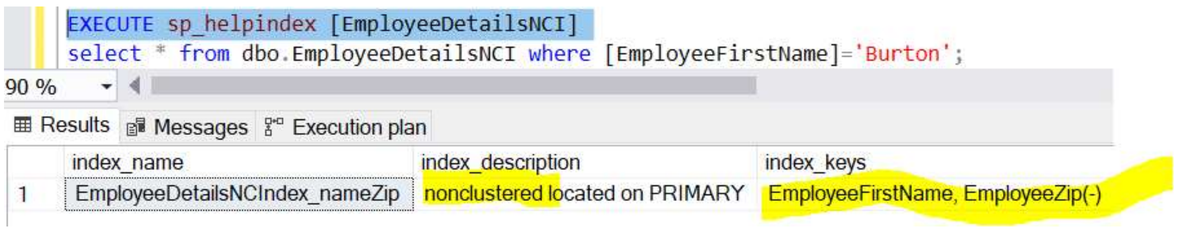
	index_name	index_description	index_keys
1	EmployeeDetailsNCINCindex_nameZip	nonclustered located on PRIMARY	EmployeeFirstName

Composite index (can only be non-clustered)

Composite index is used to create non clustered index on the multiple columns.

Here we have created composite index EmployeeDetailsNCIndex_nameZip on the table EmployeeDetailsNCI

```
CREATE TABLE [dbo].[EmployeeDetailsNCI](
        [EmployeeId] [int] NOT NULL,
        [EmployeeFirstName] [varchar](50) NULL,
        [EmployeeLastName] [varchar](50) NULL,
        [EmployeeZip] [varchar](50) NULL
)

create NONCLUSTERED INDEX EmployeeDetailsNCIndex_nameZip
ON dbo.EmployeeDetailsNCI([EmployeeFirstName] ASC,[EmployeeZip] DESC)
```

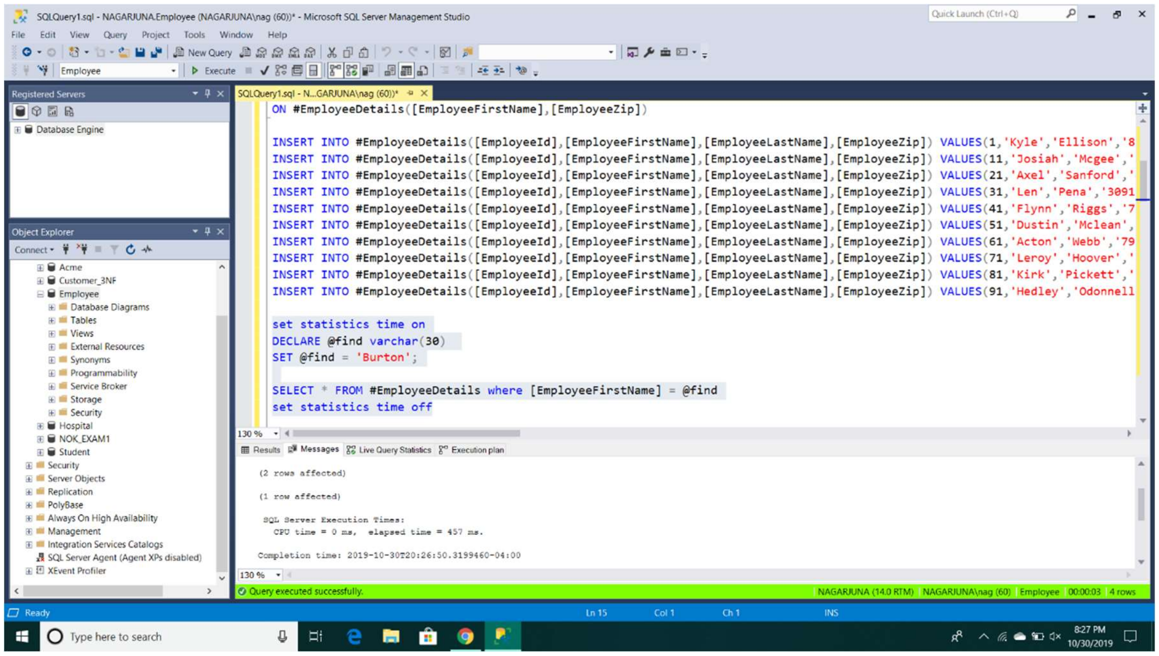


```
EXECUTE sp_helpindex [EmployeeDetailsNCI]
select * from dbo.EmployeeDetailsNCI where [EmployeeFirstName]='Burton';
```

	index_name	index_description	index_keys
1	EmployeeDetailsNCIndex_nameZip	nonclustered located on PRIMARY	EmployeeFirstName, EmployeeZip(-)

Table variable with clustered Index:

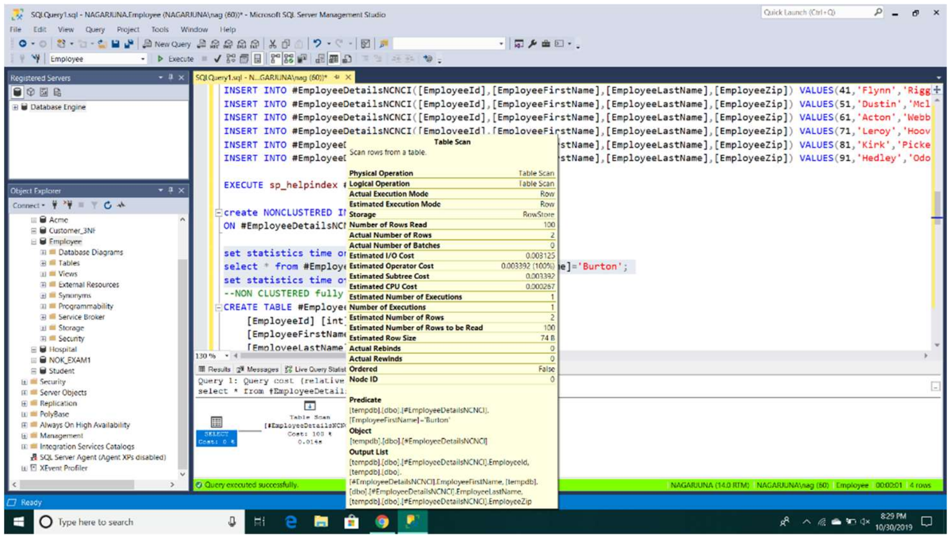
```
create CLUSTERED INDEX EmployeeDetailswithClusteredIndex
ON #EmployeeDetails([EmployeeFirstName],[EmployeeZip])
set statistics time on
DECLARE @find varchar(30)
SET @find = 'Burton';

SELECT * FROM #EmployeeDetails where [EmployeeFirstName] = @find
set statistics time off
```



```
ON #EmployeeDetails([EmployeeFirstName],[EmployeeZip])

INSERT INTO #EmployeeDetails([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(1,'Kyle','Ellison','8
INSERT INTO #EmployeeDetails([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(11,'Josiah','Mcgee','
INSERT INTO #EmployeeDetails([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(21,'Axel','Sanford','
INSERT INTO #EmployeeDetails([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(31,'Len','Pena','3091
INSERT INTO #EmployeeDetails([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(41,'Flynn','Riggs','7
INSERT INTO #EmployeeDetails([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(51,'Dustin','Mclean',
INSERT INTO #EmployeeDetails([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(61,'Acton','Webb','79
INSERT INTO #EmployeeDetails([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(71,'Leroy','Hoover','
INSERT INTO #EmployeeDetails([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(81,'Kirk','Pickett','
INSERT INTO #EmployeeDetails([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(91,'Hedley','Odonnell

set statistics time on
DECLARE @find varchar(30)
SET @find = 'Burton';

SELECT * FROM #EmployeeDetails where [EmployeeFirstName] = @find
set statistics time off
```

```
(2 rows affected)

(1 row affected)

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 457 ms.

Completion time: 2019-10-30T20:26:50.3199460-04:00
```

Here, time taken to execute the query is 457ms


SQLQuery1.sql - NAGARJUNA.Employee (NAGARJUNA\nag (60))* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
Registered Servers
Database Engine
Object Explorer
SQLQuery1.sql - N...GARJUNA\nag (60))*
Query executed successfully.
NAGARJUNA (14.0 RTM) | NAGARJUNA\nag (60) | Employee | 00:00:01 | 4 rows
Ready

```
INSERT INTO #EmployeeDetailsNCNCI([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(41,'Flynn','Rigg
INSERT INTO #EmployeeDetailsNCNCI([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(51,'Dustin','Mcl
INSERT INTO #EmployeeDetailsNCNCI([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(61,'Acton','Webb
INSERT INTO #EmployeeDetailsNCNCI([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(71,'Leroy','Hoov
INSERT INTO #EmployeeI                           stName],[EmployeeLastName],[EmployeeZip]) VALUES(81,'Kirk','Picke
INSERT INTO #EmployeeI                           stName],[EmployeeLastName],[EmployeeZip]) VALUES(91,'Hedley','Odo

EXECUTE sp_helpindex #

create NONCLUSTERED I
ON #EmployeeDetailsNCI

set statistics time o
select * from #Employe                            e]='Burton';
set statistics time o
--NON CLUSTERED fully
CREATE TABLE #Employe
    [EmployeeId] [int
    [EmployeeFirstName
    [EmployeeLastName

Query 1: Query cost (relative
select * from #EmployeeDetail

Table Scan
[#EmployeeDetailsNCN
Cost: 100 %
0.014s

SELECT
Cost: 0 %
```

Table Scan

Scan rows from a table.

Physical Operation	Table Scan
Logical Operation	Table Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Number of Rows Read	100
Actual Number of Rows	2
Actual Number of Batches	0
Estimated I/O Cost	0.003125
Estimated Operator Cost	0.003392 (100%)
Estimated Subtree Cost	0.003392
Estimated CPU Cost	0.000267
Estimated Number of Executions	1
Number of Executions	1
Estimated Number of Rows	2
Estimated Number of Rows to be Read	100
Estimated Row Size	74 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	False
Node ID	0

Predicate
[tempdb].[dbo].[#EmployeeDetailsNCNCI].[EmployeeFirstName]='Burton'
Object
[tempdb].[dbo].[#EmployeeDetailsNCNCI]
Output List
[tempdb].[dbo].[#EmployeeDetailsNCNCI].EmployeeId, [tempdb].[dbo].[#EmployeeDetailsNCNCI].EmployeeFirstName, [tempdb].[dbo].[#EmployeeDetailsNCNCI].EmployeeLastName, [tempdb].[dbo].[#EmployeeDetailsNCNCI].EmployeeZip

Number of rows read 100 as it did scan the table.

Non clustered and non-covering Index on temp table:

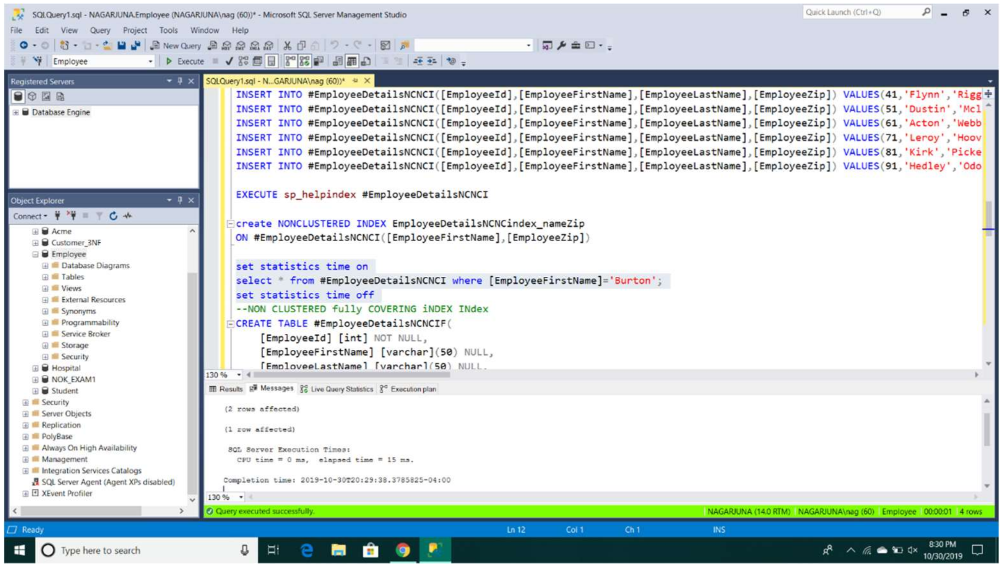
create NONCLUSTERED INDEX EmployeeDetailsNCNCindex_nameZip

ON #EmployeeDetailsNCNCI([EmployeeFirstName],[EmployeeZip])

set statistics time on

select * from #EmployeeDetailsNCNCI where [EmployeeFirstName]='Burton';

set statistics time off


SQLQuery1.sql - NAGARJUNA.Employee (NAGARJUNA\nag (60))* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
SQLQuery1.sql - N...GARJUNA\nag (60))*
Results
Messages
Live Query Statistics
Execution plan
Query executed successfully.
NAGARJUNA (14.0 RTM) | NAGARJUNA\nag (60) | Employee | 00:00:01 | 4 rows
Ln 12 Col 1 Ch 1 INS

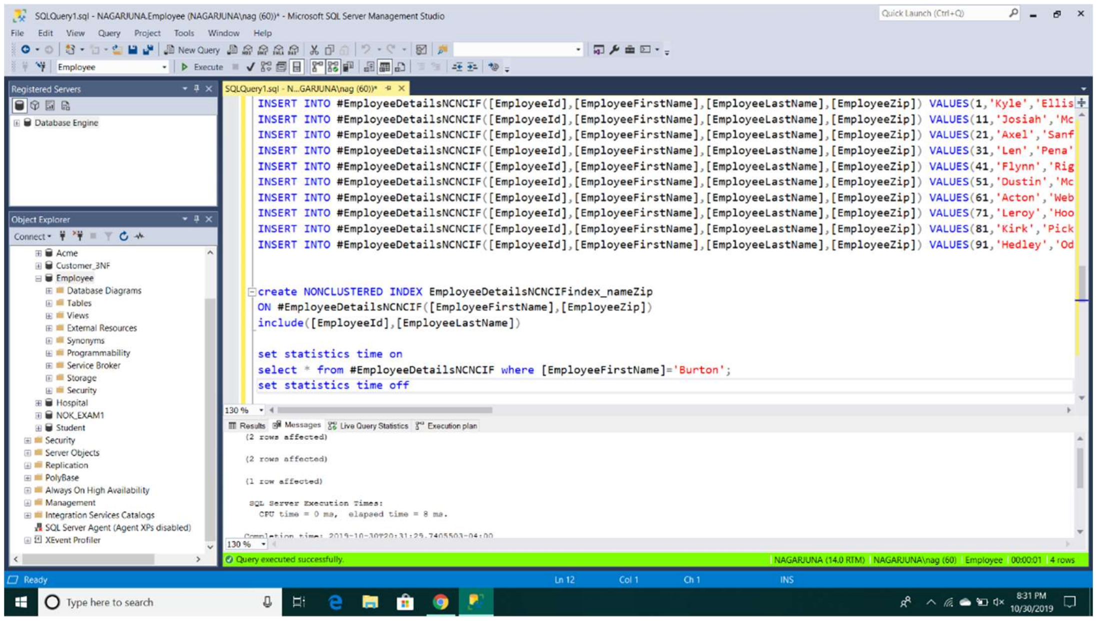
```
INSERT INTO #EmployeeDetailsNCNCI([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(41,'Flynn','Rigg
INSERT INTO #EmployeeDetailsNCNCI([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(51,'Dustin','Mcl
INSERT INTO #EmployeeDetailsNCNCI([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(61,'Acton','Webb
INSERT INTO #EmployeeDetailsNCNCI([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(71,'Leroy','Hoov
INSERT INTO #EmployeeDetailsNCNCI([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(81,'Kirk','Picke
INSERT INTO #EmployeeDetailsNCNCI([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(91,'Hedley','Odo

EXECUTE sp_helpindex #EmployeeDetailsNCNCI

create NONCLUSTERED INDEX EmployeeDetailsNCNCindex_nameZip
ON #EmployeeDetailsNCNCI([EmployeeFirstName],[EmployeeZip])

set statistics time on
select * from #EmployeeDetailsNCNCI where [EmployeeFirstName]='Burton';
set statistics time off
--NON CLUSTERED fully COVERING iNDEX INdex
CREATE TABLE #EmployeeDetailsNCNCIF(
    [EmployeeId] [int] NOT NULL,
    [EmployeeFirstName] [varchar](50) NULL,
    [EmployeeLastName] [varchar](50) NULL,
```

```
(2 rows affected)

(1 row affected)

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 15 ms.

Completion time: 2019-10-30T20:29:38.3785825-04:00
```

Here, it took 15ms to execute the query

Non-clustered fully covering Index:

```
create NONCLUSTERED INDEX EmployeeDetailsNCNCIFindex_nameZip
ON #EmployeeDetailsNCNCIF([EmployeeFirstName],[EmployeeZip])
include([EmployeeId],[EmployeeLastName])


set statistics time on
select * from #EmployeeDetailsNCNCIF where [EmployeeFirstName]='Burton';
set statistics time off
```

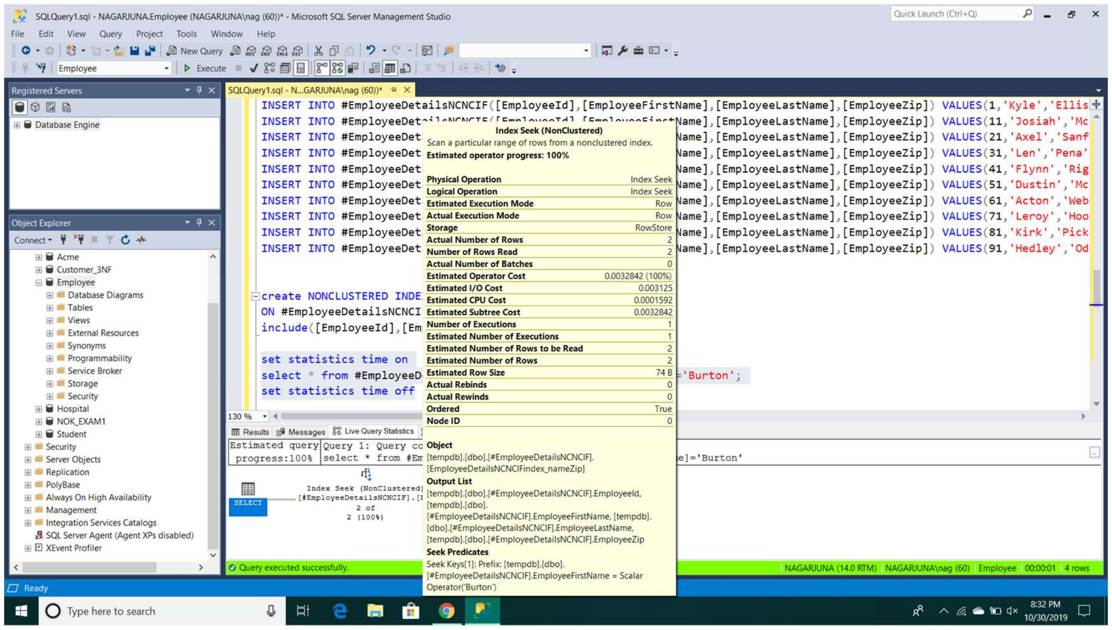


```
INSERT INTO #EmployeeDetailsNCNCIF([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(1,'Kyle','Ellis
INSERT INTO #EmployeeDetailsNCNCIF([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(11,'Josiah','Mc
INSERT INTO #EmployeeDetailsNCNCIF([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(21,'Axel','Sanf
INSERT INTO #EmployeeDetailsNCNCIF([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(31,'Len','Pena'
INSERT INTO #EmployeeDetailsNCNCIF([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(41,'Flynn','Rig
INSERT INTO #EmployeeDetailsNCNCIF([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(51,'Dustin','Mc
INSERT INTO #EmployeeDetailsNCNCIF([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(61,'Acton','Web
INSERT INTO #EmployeeDetailsNCNCIF([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(71,'Leroy','Hoo
INSERT INTO #EmployeeDetailsNCNCIF([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(81,'Kirk','Pick
INSERT INTO #EmployeeDetailsNCNCIF([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(91,'Hedley','Od


create NONCLUSTERED INDEX EmployeeDetailsNCNCIFindex_nameZip
ON #EmployeeDetailsNCNCIF([EmployeeFirstName],[EmployeeZip])
include([EmployeeId],[EmployeeLastName])

set statistics time on
select * from #EmployeeDetailsNCNCIF where [EmployeeFirstName]='Burton';
set statistics time off
```

```
(2 rows affected)

(2 rows affected)

(1 row affected)

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 8 ms.

Completion time: 2019-10-30T20:31:29.7405503-04:00
```

Query executed successfully. NAGARJUNA (14.0 RTM) NAGARJUNA\nag (60) Employee 00:00:01 4 rows

Here, the execution time is 8ms


SQLQuery1.sql - NAGARJUNA.Employee (NAGARJUNA\nag (60))* - Microsoft SQL Server Management Studio

```
INSERT INTO #EmployeeDetailsNCNCIF([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(1,'Kyle','Ellis
INSERT INTO #EmployeeDet Name],[EmployeeLastName],[EmployeeZip]) VALUES(11,'Josiah','Mc
INSERT INTO #EmployeeDet Name],[EmployeeLastName],[EmployeeZip]) VALUES(21,'Axel','Sanf
INSERT INTO #EmployeeDet Name],[EmployeeLastName],[EmployeeZip]) VALUES(31,'Len','Pena'
INSERT INTO #EmployeeDet Name],[EmployeeLastName],[EmployeeZip]) VALUES(41,'Flynn','Rig
INSERT INTO #EmployeeDet Name],[EmployeeLastName],[EmployeeZip]) VALUES(51,'Dustin','Mc
INSERT INTO #EmployeeDet Name],[EmployeeLastName],[EmployeeZip]) VALUES(61,'Acton','Web
INSERT INTO #EmployeeDet Name],[EmployeeLastName],[EmployeeZip]) VALUES(71,'Leroy','Hoo
INSERT INTO #EmployeeDet Name],[EmployeeLastName],[EmployeeZip]) VALUES(81,'Kirk','Pick
INSERT INTO #EmployeeDet Name],[EmployeeLastName],[EmployeeZip]) VALUES(91,'Hedley','Od

create NONCLUSTERED INDE
ON #EmployeeDetailsNCNCI
include([EmployeeId],[Em

set statistics time on
select * from #EmployeeD ='Burton';
set statistics time off
```

Index Seek (NonClustered)
Scan a particular range of rows from a nonclustered index.
Estimated operator progress: 100%

Physical Operation	Index Seek
Logical Operation	Index Seek
Estimated Execution Mode	Row
Actual Execution Mode	Row
Storage	RowStore
Actual Number of Rows	2
Number of Rows Read	2
Actual Number of Batches	0
Estimated Operator Cost	0.0032842 (100%)
Estimated I/O Cost	0.003125
Estimated CPU Cost	0.0001592
Estimated Subtree Cost	0.0032842
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows to be Read	2
Estimated Number of Rows	2
Estimated Row Size	74 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	0

Object
[tempdb].[dbo].[#EmployeeDetailsNCNCIF].[EmployeeDetailsNCNCIFindex_nameZip]
Output List
[tempdb].[dbo].[#EmployeeDetailsNCNCIF].EmployeeId, [tempdb].[dbo].[#EmployeeDetailsNCNCIF].EmployeeFirstName, [tempdb].[dbo].[#EmployeeDetailsNCNCIF].EmployeeLastName, [tempdb].[dbo].[#EmployeeDetailsNCNCIF].EmployeeZip
Seek Predicates
Seek Keys[1]: Prefix: [tempdb].[dbo].[#EmployeeDetailsNCNCIF].EmployeeFirstName = Scalar Operator('Burton')

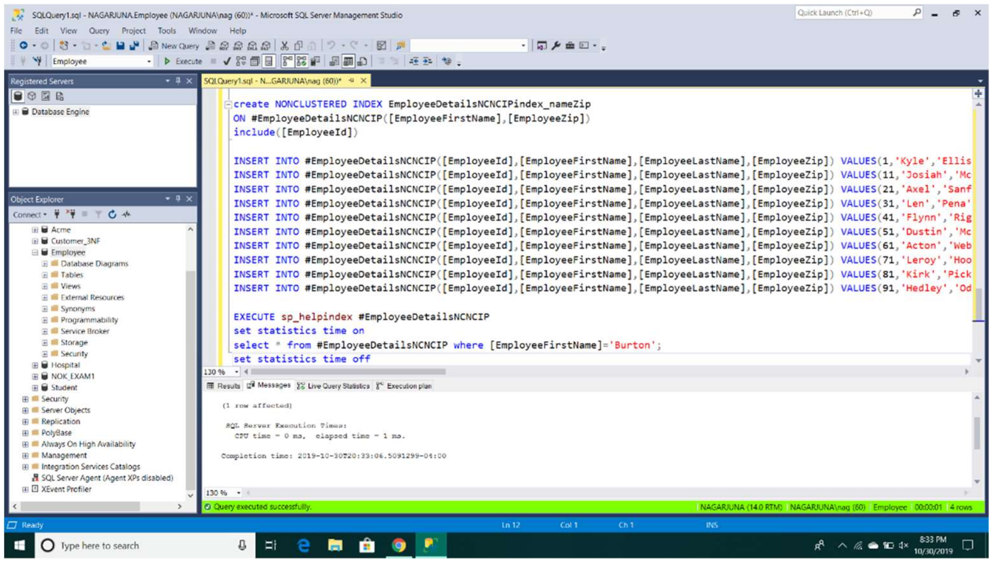
```
Estimated query progress:100%   Query 1: Query cc
                                select * from #Em   e]='Burton'
SELECT    Index Seek (NonClustered)
          [#EmployeeDetailsNCNCIF].[I
          2 of
          2 (100%)
```

Query executed successfully.   NAGARJUNA (14.0 RTM) | NAGARJUNA\nag (60) | Employee | 00:00:01 | 4 rows

Here, number of rows read are 2 as it did a seek operation

Non-clustered partial covering Index:

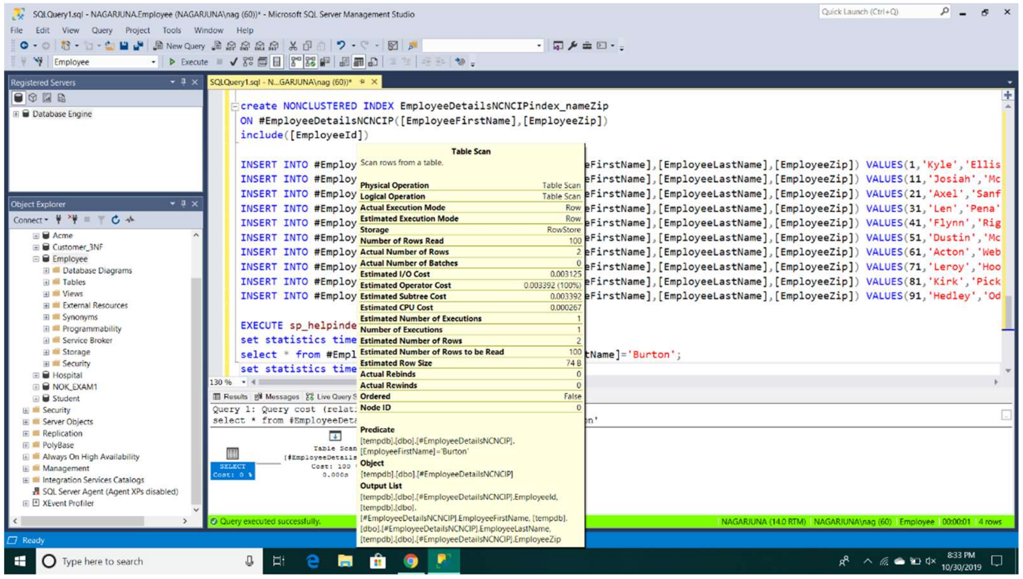
```
create NONCLUSTERED INDEX EmployeeDetailsNCNCIPindex_nameZip
ON #EmployeeDetailsNCNCIP([EmployeeFirstName],[EmployeeZip])
include([EmployeeId])


set statistics time on
select * from #EmployeeDetailsNCNCIP where [EmployeeFirstName]='Burton';
set statistics time off
```



```
create NONCLUSTERED INDEX EmployeeDetailsNCNCIPindex_nameZip
ON #EmployeeDetailsNCNCIP([EmployeeFirstName],[EmployeeZip])
include([EmployeeId])

INSERT INTO #EmployeeDetailsNCNCIP([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(1,'Kyle','Ellis
INSERT INTO #EmployeeDetailsNCNCIP([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(11,'Josiah','Mc
INSERT INTO #EmployeeDetailsNCNCIP([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(21,'Axel','Sanf
INSERT INTO #EmployeeDetailsNCNCIP([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(31,'Len','Pena'
INSERT INTO #EmployeeDetailsNCNCIP([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(41,'Flynn','Rig
INSERT INTO #EmployeeDetailsNCNCIP([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(51,'Dustin','Mc
INSERT INTO #EmployeeDetailsNCNCIP([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(61,'Acton','Web
INSERT INTO #EmployeeDetailsNCNCIP([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(71,'Leroy','Hoo
INSERT INTO #EmployeeDetailsNCNCIP([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(81,'Kirk','Pick
INSERT INTO #EmployeeDetailsNCNCIP([EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZip]) VALUES(91,'Hedley','Od

EXECUTE sp_helpindex #EmployeeDetailsNCNCIP
set statistics time on
select * from #EmployeeDetailsNCNCIP where [EmployeeFirstName]='Burton';
set statistics time off
```

```
(1 row affected)

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 1 ms.

Completion time: 2019-10-30T20:33:06.5091299-04:00
```

Here, the execution time is 1ms



```
Query 1: Query cost (relat
select * from #EmployeeDeta
```

Table Scan
Scan rows from a table.

Physical Operation	Table Scan
Logical Operation	Table Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Number of Rows Read	100
Actual Number of Rows	2
Actual Number of Batches	0
Estimated I/O Cost	0.003125
Estimated Operator Cost	0.003392 (100%)
Estimated Subtree Cost	0.003392
Estimated CPU Cost	0.000267
Estimated Number of Executions	1
Number of Executions	1
Estimated Number of Rows	2
Estimated Number of Rows to be Read	100
Estimated Row Size	74 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	False
Node ID	0

Predicate
[tempdb].[dbo].[#EmployeeDetailsNCNCIP].[EmployeeFirstName]='Burton'
Object
[tempdb].[dbo].[#EmployeeDetailsNCNCIP]
Output List
[tempdb].[dbo].[#EmployeeDetailsNCNCIP].EmployeeId, [tempdb].[dbo].[#EmployeeDetailsNCNCIP].EmployeeFirstName, [tempdb].[dbo].[#EmployeeDetailsNCNCIP].EmployeeLastName, [tempdb].[dbo].[#EmployeeDetailsNCNCIP].EmployeeZip

Here, it has read all 100 rows as it did table scan

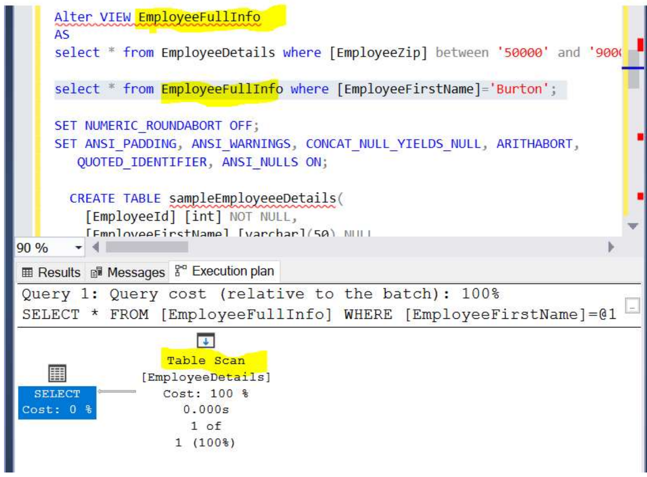
Create a view with and without an index

Here first we created the vnormal view(EmployeeFullInfo) to retrieve the details of the burton, we can see that the query optimizer used the table scan option.

Then we created the view (EmployeeFullInfo_Indexed) with a index EmployeeDetailsCindex_ID on the table.

Also here we noticed that the number of rows read is 100 in the normal view while the index view reads just 44 rows to get the details of the Burton.

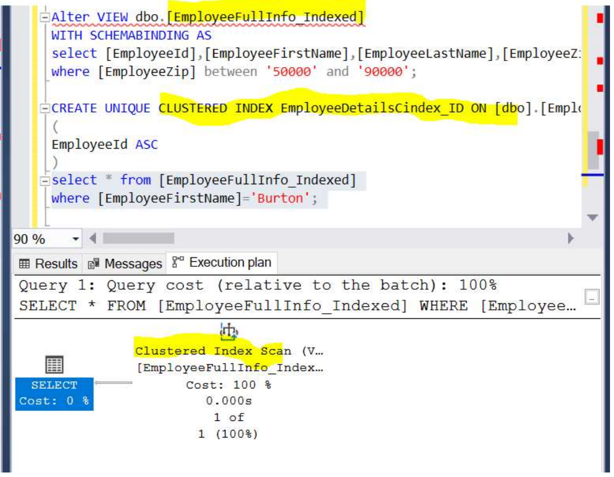
Also we have seen the estimated query cost is different, i.e. the view with a index has a low query cost compared to the normal view created on the table.



```
Alter VIEW EmployeeFullInfo
AS
select * from EmployeeDetails where [EmployeeZip] between '50000' and '900(

select * from EmployeeFullInfo where [EmployeeFirstName]='Burton';

SET NUMERIC_ROUNDABORT OFF;
SET ANSI_PADDING, ANSI_WARNINGS, CONCAT_NULL_YIELDS_NULL, ARITHABORT,
   QUOTED_IDENTIFIER, ANSI_NULLS ON;

  CREATE TABLE sampleEmployeeeDetails(
    [EmployeeId] [int] NOT NULL,
```

```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM [EmployeeFullInfo] WHERE [EmployeeFirstName]=@1
```

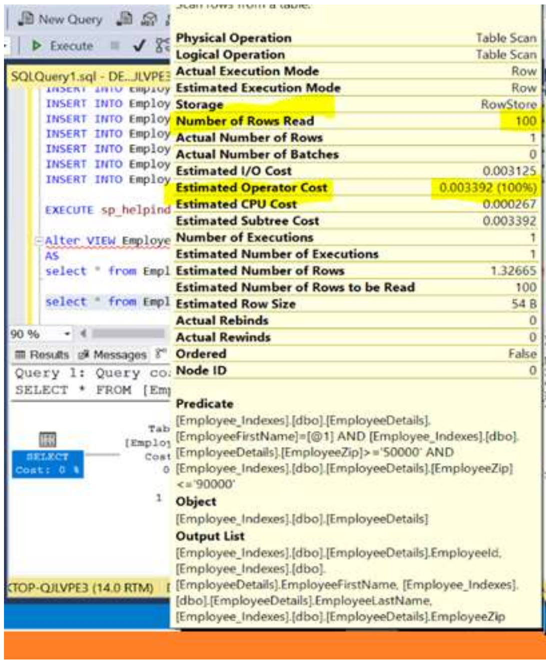
```
SELECT        Table Scan
Cost: 0 %     [EmployeeDetails]
              Cost: 100 %
              0.000s
              1 of
              1 (100%)
```



```
Alter VIEW dbo.[EmployeeFullInfo_Indexed]
WITH SCHEMABINDING AS
select [EmployeeId],[EmployeeFirstName],[EmployeeLastName],[EmployeeZ
where [EmployeeZip] between '50000' and '90000';

CREATE UNIQUE CLUSTERED INDEX EmployeeDetailsCindex_ID ON [dbo].[Emplo
(
EmployeeId ASC
)
select * from [EmployeeFullInfo_Indexed]
where [EmployeeFirstName]='Burton';
```

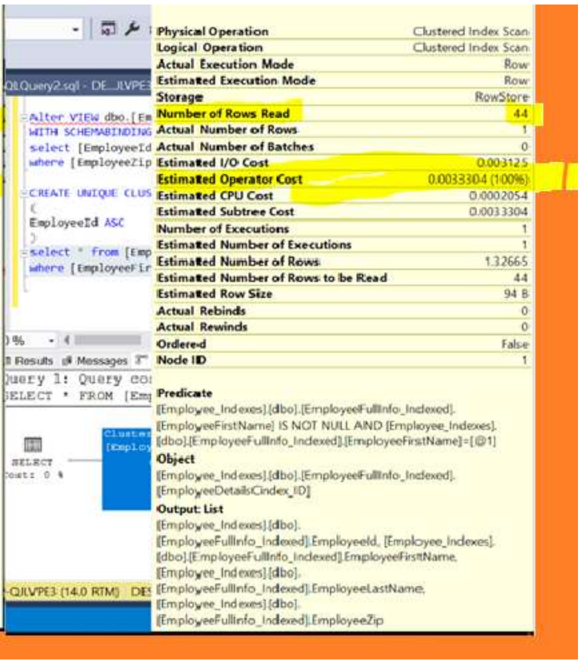
```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM [EmployeeFullInfo_Indexed] WHERE [Employee…
```

```
SELECT        Clustered Index Scan (V…
Cost: 0 %     [EmployeeFullInfo_Index…
              Cost: 100 %
              0.000s
              1 of
              1 (100%)
```



Physical Operation	Table Scan
Logical Operation	Table Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Number of Rows Read	100
Actual Number of Rows	1
Actual Number of Batches	0
Estimated I/O Cost	0.003125
Estimated Operator Cost	0.003392 (100%)
Estimated CPU Cost	0.000267
Estimated Subtree Cost	0.003392
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows	1.32665
Estimated Number of Rows to be Read	100
Estimated Row Size	54 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	False
Node ID	0

Predicate
[Employee_Indexes].[dbo].[EmployeeDetails].[EmployeeFirstName]=[@1] AND [Employee_Indexes].[dbo].[EmployeeDetails].[EmployeeZip]>='50000' AND [Employee_Indexes].[dbo].[EmployeeDetails].[EmployeeZip]<='90000'
Object
[Employee_Indexes].[dbo].[EmployeeDetails]
Output List
[Employee_Indexes].[dbo].[EmployeeDetails].EmployeeId, [Employee_Indexes].[dbo].[EmployeeDetails].EmployeeFirstName, [Employee_Indexes].[dbo].[EmployeeDetails].EmployeeLastName, [Employee_Indexes].[dbo].[EmployeeDetails].EmployeeZip



Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Number of Rows Read	44
Actual Number of Rows	1
Actual Number of Batches	0
Estimated I/O Cost	0.003125
Estimated Operator Cost	0.0033304 (100%)
Estimated CPU Cost	0.0002054
Estimated Subtree Cost	0.0033304
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows	1.32665
Estimated Number of Rows to be Read	44
Estimated Row Size	94 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	False
Node ID	1

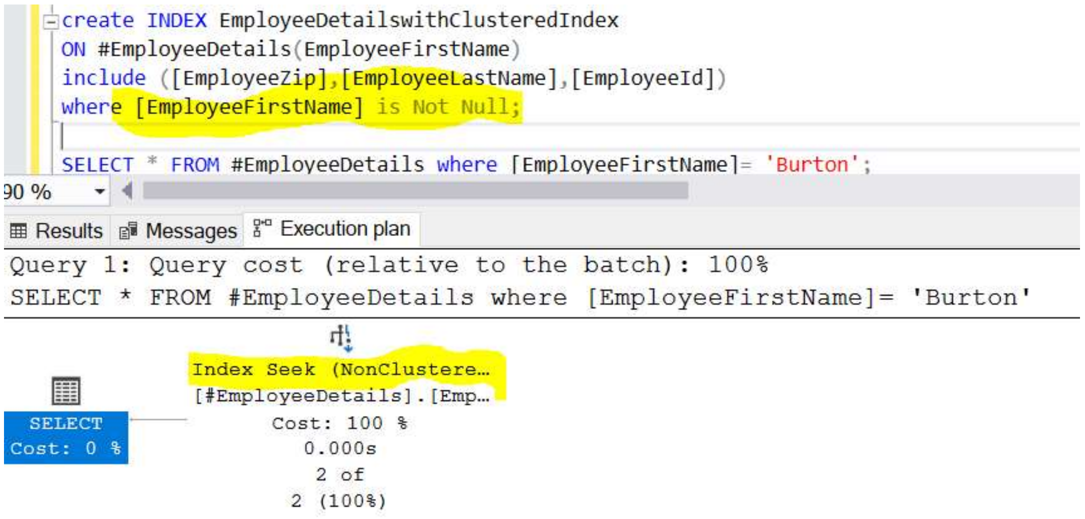
Predicate
[Employee_Indexes].[dbo].[EmployeeFullInfo_Indexed].[EmployeeFirstName] IS NOT NULL AND [Employee_Indexes].[dbo].[EmployeeFullInfo_Indexed].[EmployeeFirstName]=[@1]
Object
[Employee_Indexes].[dbo].[EmployeeFullInfo_Indexed].[EmployeeDetailsCindex_ID]
Output List
[Employee_Indexes].[dbo].[EmployeeFullInfo_Indexed].EmployeeId, [Employee_Indexes].[dbo].[EmployeeFullInfo_Indexed].EmployeeFirstName, [Employee_Indexes].[dbo].[EmployeeFullInfo_Indexed].EmployeeLastName, [Employee_Indexes].[dbo].[EmployeeFullInfo_Indexed].EmployeeZip

Use a filtered index for any of the above

Here we created the non-clustered index with the filter EmployeeFirstName is not null,

Here in the Employee Table we have some firstnames has null values,so it creates a index with the EmployeeFirstName without any nulls.

```
CREATE TABLE #EmployeeDetails
(
	[EmployeeId] [int] NOT NULL,
	[EmployeeFirstName] [varchar](50) NULL,
	[EmployeeLastName] [varchar](50) NULL,
	[EmployeeZip] [varchar](50) NULL
)
create INDEX EmployeeDetailswithClusteredIndex
ON #EmployeeDetails(EmployeeFirstName)
include ([EmployeeZip],[EmployeeLastName],[EmployeeId])

where [EmployeeFirstName] is Not Null;
```



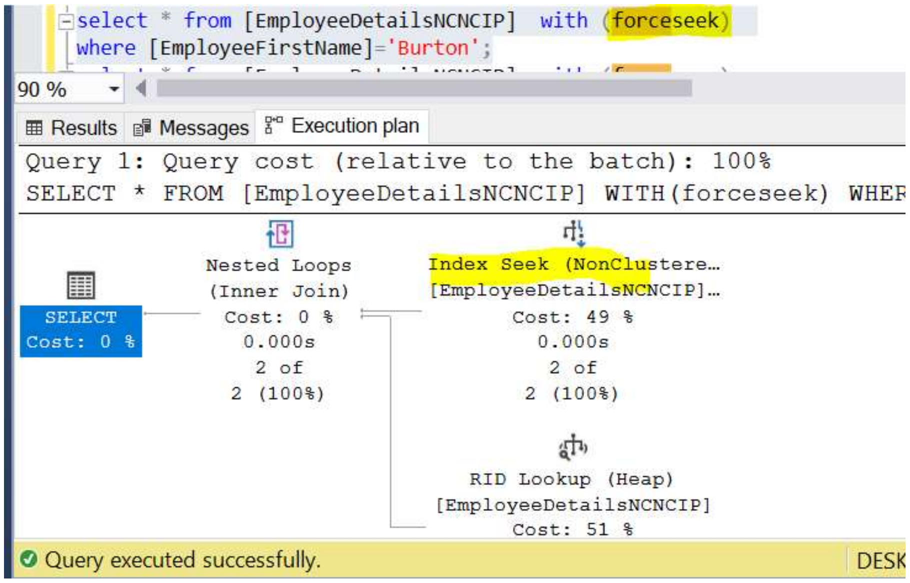
```
create INDEX EmployeeDetailswithClusteredIndex
ON #EmployeeDetails(EmployeeFirstName)
include ([EmployeeZip],[EmployeeLastName],[EmployeeId])
where [EmployeeFirstName] is Not Null;

SELECT * FROM #EmployeeDetails where [EmployeeFirstName]= 'Burton';
```

```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM #EmployeeDetails where [EmployeeFirstName]= 'Burton'
```

Create Execution plans using hints

Here the FORCESEEK hint constrains the query optimizer to use only an index seek for definining the access path to the data.

Also we can see in the below screenshot that the the query optimizer uses the index seek to fetch the details of the burton

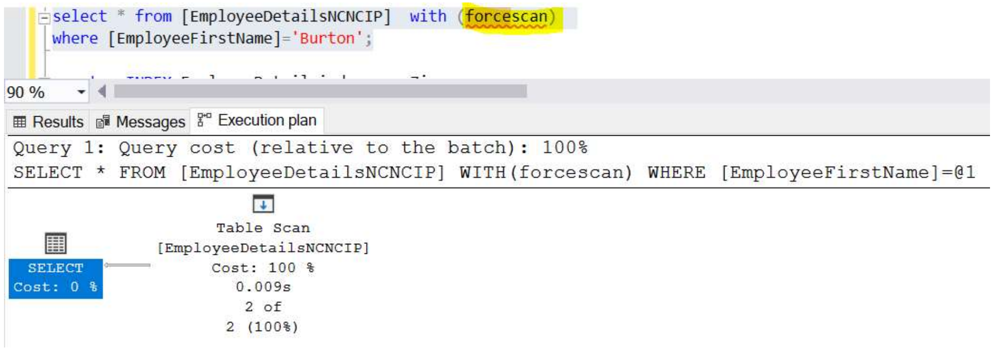


```
select * from [EmployeeDetailsNCNCIP]  with (forceseek)
where [EmployeeFirstName]='Burton';
```

```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM [EmployeeDetailsNCNCIP] WITH(forceseek) WHER
```

Query executed successfully.

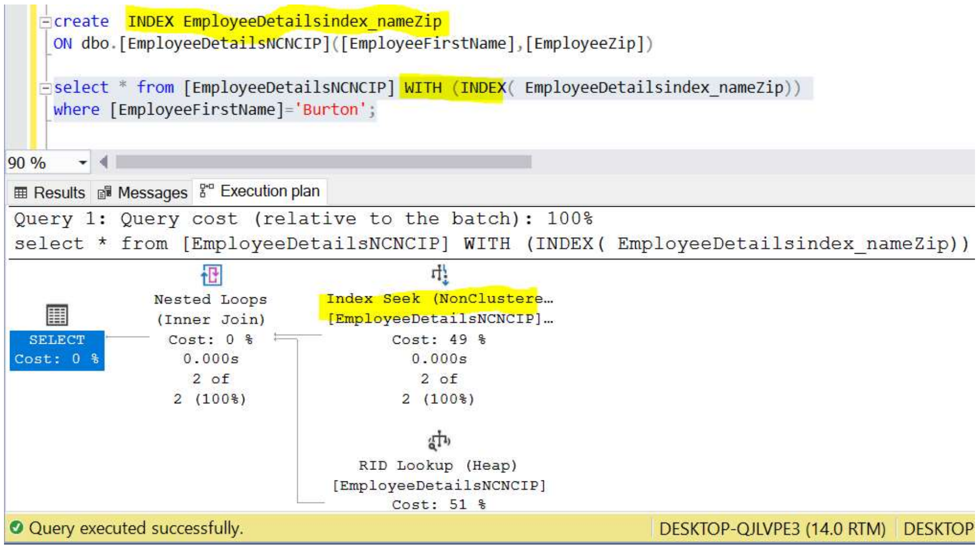
While the query optimizer uses the table scan to fetch the details of the Burton.



```
select * from [EmployeeDetailsNCNCIP]  with (forcescan)
where [EmployeeFirstName]='Burton';
```

```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM [EmployeeDetailsNCNCIP] WITH(forcescan) WHERE [EmployeeFirstName]=@1
```

Here we demonstrated the with index to retrieve the details of the Employee Burton.



```
create  INDEX EmployeeDetailsindex_nameZip
ON dbo.[EmployeeDetailsNCNCIP]([EmployeeFirstName],[EmployeeZip])

select * from [EmployeeDetailsNCNCIP] WITH (INDEX( EmployeeDetailsindex_nameZip))
where [EmployeeFirstName]='Burton';
```

```
Query 1: Query cost (relative to the batch): 100%
select * from [EmployeeDetailsNCNCIP] WITH (INDEX( EmployeeDetailsindex_nameZip))
```

Query executed successfully.

Index Choices	Execution Time(ms)
No index	1ms
Primary Key Index	1ms
Primary Clustered Index	11ms
Custom Clustered Index	1ms
Non-Clustered Composite Index	9ms
Clustered Primary Index and non-clustered In	0ms
Temp Table with Non-Covering Index	15ms
Temp table with a fully covering Index	8ms
Filtered Index	6ms

Since we have taken a small database for this analysis, the results are not effective, but from the results we found the combined clustered and non- clustered performs better comparatively.

Demonstrate use of: disable, rebuild, drop.You must run queries with execution plans showing the difference between index and disabled index performance.

Here first we created the table EmployeeDetailsNCNCIF to demonstrate the use of Disable, Rebuilt and Drop on the indexes created.

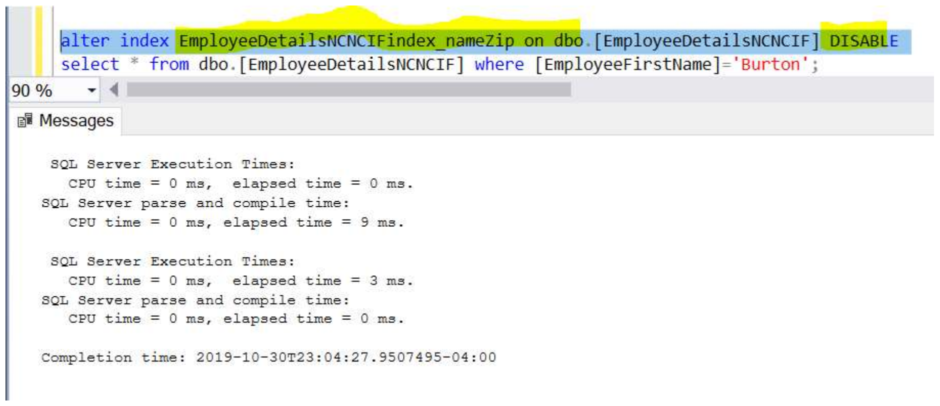
We created the non clustered index on the table EmployeeDetailsNCNCIFindex_nameZip

Syntax:

```
CREATE TABLE [dbo].[EmployeeDetailsNCNCIF](
        [EmployeeId] [int] NOT NULL,
        [EmployeeFirstName] [varchar](50) NULL,
        [EmployeeLastName] [varchar](50) NULL,
        [EmployeeZip] [varchar](50) NULL
)

create NONCLUSTERED INDEX EmployeeDetailsNCNCIFindex_nameZip
ON dbo.[EmployeeDetailsNCNCIF]([EmployeeFirstName],[EmployeeZip])

include([EmployeeId],[EmployeeLastName])
```

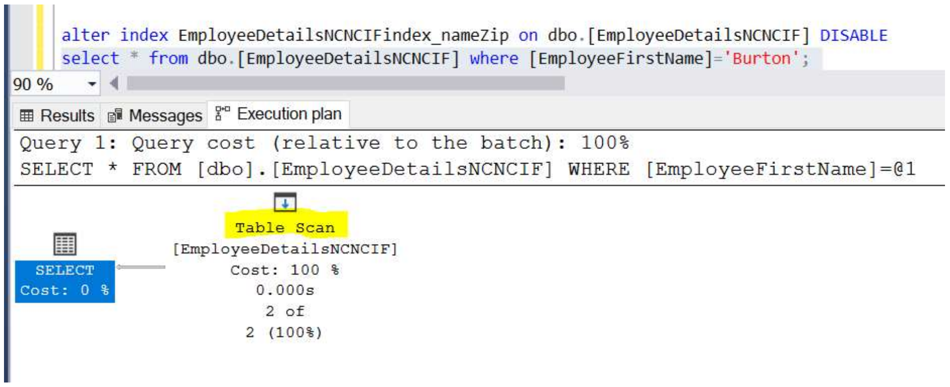
Disable:

- Here we have used the disable option on the existing index EmployeeDetailsNCNCIFindex_nameZip.
- We can see in the below screenshot that the employee last is retrieved using the table scan.
- We generally prefer to disable the Index if we are going to re-enable it again.



```
alter index EmployeeDetailsNCNCIFindex_nameZip on dbo.[EmployeeDetailsNCNCIF] DISABLE
select * from dbo.[EmployeeDetailsNCNCIF] where [EmployeeFirstName]='Burton';
```

```
 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 9 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 3 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

Completion time: 2019-10-30T23:04:27.9507495-04:00
```



```
alter index EmployeeDetailsNCNCIFindex_nameZip on dbo.[EmployeeDetailsNCNCIF] DISABLE
select * from dbo.[EmployeeDetailsNCNCIF] where [EmployeeFirstName]='Burton';
```

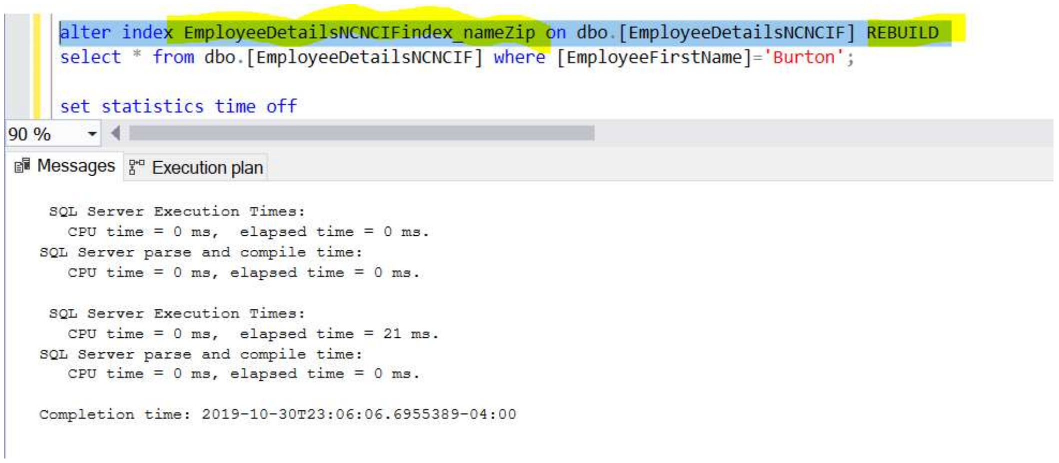
```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM [dbo].[EmployeeDetailsNCNCIF] WHERE [EmployeeFirstName]=@1
```

Rebuilt:

Here we have used the rebuilt is used to enable the disalebed index.

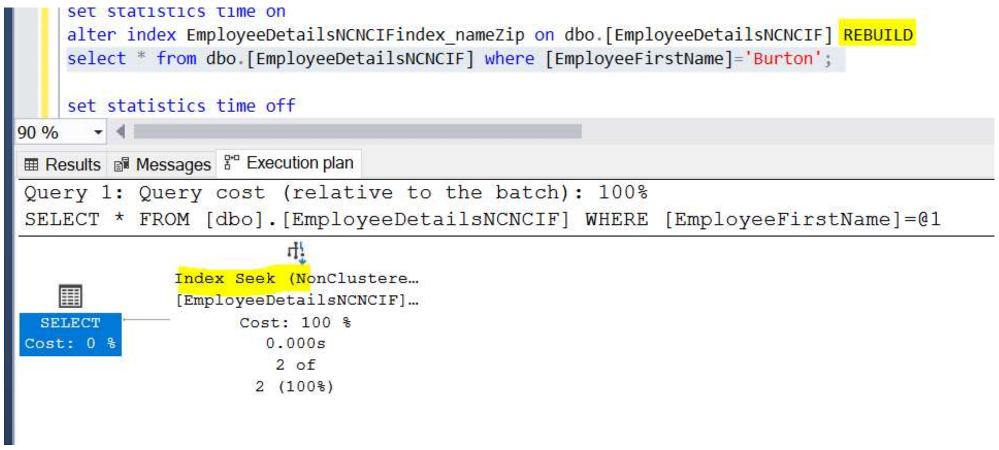
The rebuilt operator enables the EmployeeDetailsNCNCIFindex_nameZip index on the EmployeeDetailsNCNCIF table.

The query optimizer uses the index seek to retrieve the Burton employee details



```
alter index EmployeeDetailsNCNCIFindex_nameZip on dbo.[EmployeeDetailsNCNCIF] REBUILD
select * from dbo.[EmployeeDetailsNCNCIF] where [EmployeeFirstName]='Burton';

set statistics time off
```

```
 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 21 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

Completion time: 2019-10-30T23:06:06.6955389-04:00
```



```
set statistics time on
alter index EmployeeDetailsNCNCIFindex_nameZip on dbo.[EmployeeDetailsNCNCIF] REBUILD
select * from dbo.[EmployeeDetailsNCNCIF] where [EmployeeFirstName]='Burton';

set statistics time off
```

```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM [dbo].[EmployeeDetailsNCNCIF] WHERE [EmployeeFirstName]=@1
```

DROP:

Drop statement to remove existing indexes. Also we see that the rebuilt cant be used once the drop is used on the index.

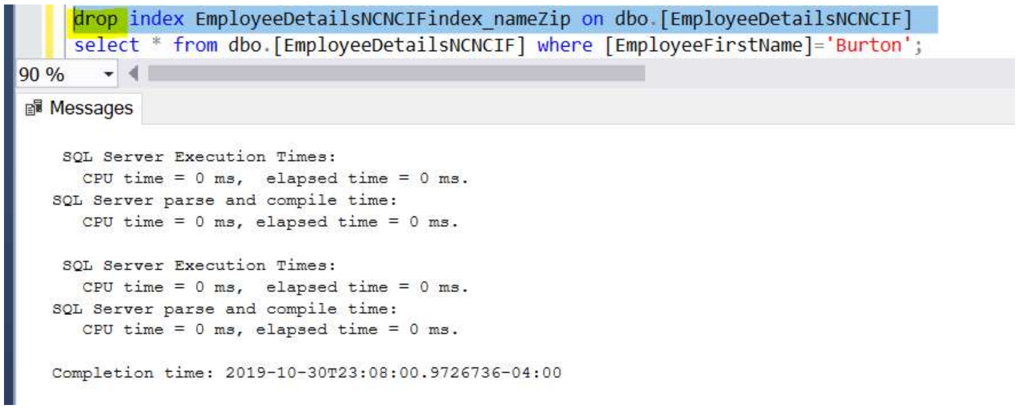
Also the query optimizer uses the table scan to retrieve the employee details of burton

We received the error as below when we try to rebuilt the index.

Msg 2727, Level 11, State 1, Line 246

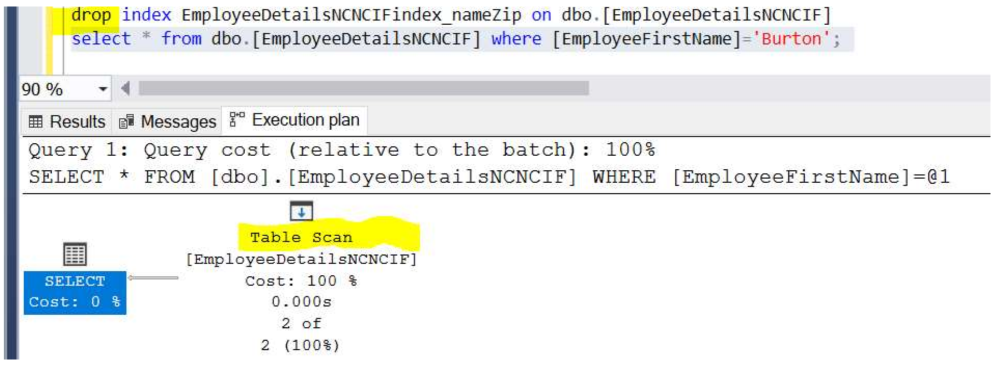
Cannot find index 'EmployeeDetailsNCNCIFindex_nameZip'.

We can see the same in the below screenshots.



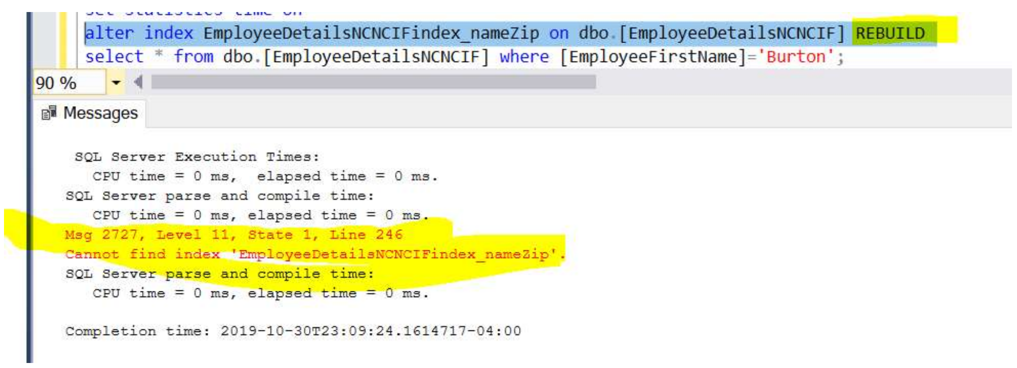
```
drop index EmployeeDetailsNCNCIFindex_nameZip on dbo.[EmployeeDetailsNCNCIF]
select * from dbo.[EmployeeDetailsNCNCIF] where [EmployeeFirstName]='Burton';
```

```
 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

Completion time: 2019-10-30T23:08:00.9726736-04:00
```



```
drop index EmployeeDetailsNCNCIFindex_nameZip on dbo.[EmployeeDetailsNCNCIF]
select * from dbo.[EmployeeDetailsNCNCIF] where [EmployeeFirstName]='Burton';
```

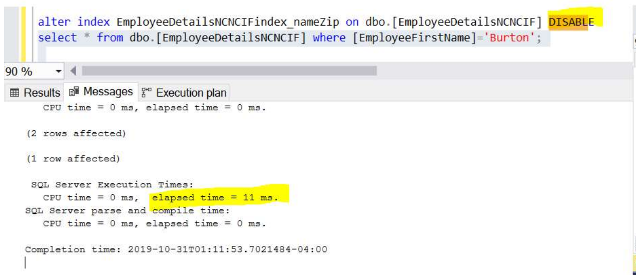
```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM [dbo].[EmployeeDetailsNCNCIF] WHERE [EmployeeFirstName]=@1
```



```
alter index EmployeeDetailsNCNCIFindex_nameZip on dbo.[EmployeeDetailsNCNCIF] REBUILD
select * from dbo.[EmployeeDetailsNCNCIF] where [EmployeeFirstName]='Burton';
```

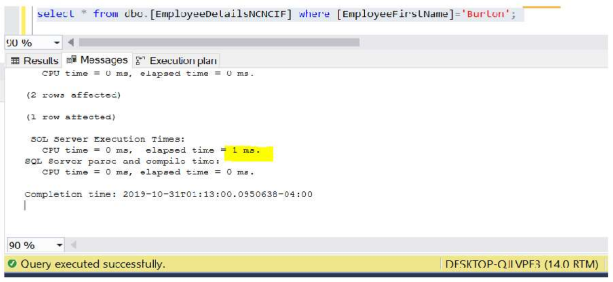
```
 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.
Msg 2727, Level 11, State 1, Line 246
Cannot find index 'EmployeeDetailsNCNCIFindex_nameZip'.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

Completion time: 2019-10-30T23:09:24.1614717-04:00
```

Execution time after disable and rebuilt of the index:



```
alter index EmployeeDetailsNCNCIFindex_nameZip on dbo.[EmployeeDetailsNCNCIF] DISABLE
select * from dbo.[EmployeeDetailsNCNCIF] where [EmployeeFirstName]='Burton';
```

```
   CPU time = 0 ms, elapsed time = 0 ms.

(2 rows affected)

(1 row affected)

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 11 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

Completion time: 2019-10-31T01:11:53.7021484-04:00
```



```
select * from dbo.[EmployeeDetailsNCNCIF] where [EmployeeFirstName]='Burton';
```

```
   CPU time = 0 ms, elapsed time = 0 ms.

(2 rows affected)

(1 row affected)

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 1 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

Completion time: 2019-10-31T01:13:00.0950638-04:00
```

Query executed successfully. DESKTOP-QIIVPF3 (14.0 RTM)

Index status	Disable	Rebuilt
Execution Time	11ms	1ms