NonClustered Index Structure

sqlservercentral.com/articles/Indexing/136537

The index internals has always been an interesting topic for SQL Developers. This article will clarify some of the interesting internal facts about non-clustered and clustered indexes based on these two statements:

1. In a unique non-clustered index, the clustered index key is added to the leaf level of the non-clustered B-Tree structure
2. In a non-unique non-clustered index, the clustered index key is added to the leaf and non-leaf levels of non-clustered B-Tree structure

The clustered index is added to the leaf level of non-clustered index irrespective of whether the non-clustered index is unique or non-unique, I will explain this concept by creating an SQL table and visualizing the index structures.

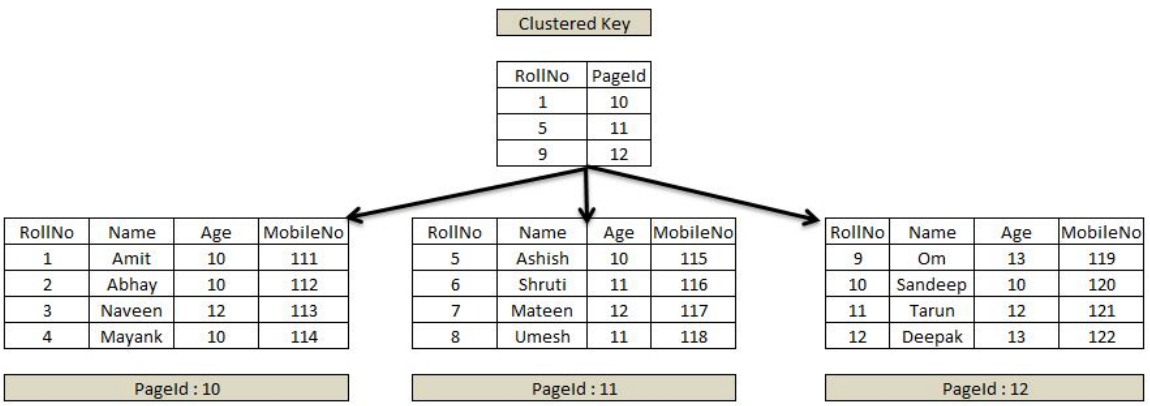
Consider a table, *dbo.Class*:

```
CREATE TABLE dbo.Class(
  RollNo INT,
  Name VARCHAR(50),
  Age INT,
  MobileNo VARCHAR(10)
);
```

Once we have created the table, we will create clustered and non-clustered indexes and will examine different scenarios. First, create a clustered index on this table with the column, RollNo, as the clustered index key. Let's assume I have just two levels: root and leaf.

```
create unique clustered index cix_class on dbo.Class (RollNo)
```

The clustered index sorts and stores the data in a B-Tree structure with index keys at the root and intermediate levels. The table contents are stored in the leaf level of B-Tree. The B-Tree structure of the clustered index looks like Figure 1 below.



Zoom in | Open in new window

Figure 1

Now we create a unique non-clustered index on table dbo.class with MobileNo as the index key.

```
create unique nonclustered index nix_demo_mobile on dbo.Class(MobileNo)
```

A unique non-clustered index is created with the non-clustered index key being added to the root, intermediate and leaf level pages. The leaf level page of the non-clustered index also includes clustered index key viz. RollNo as shown in Figure 2.

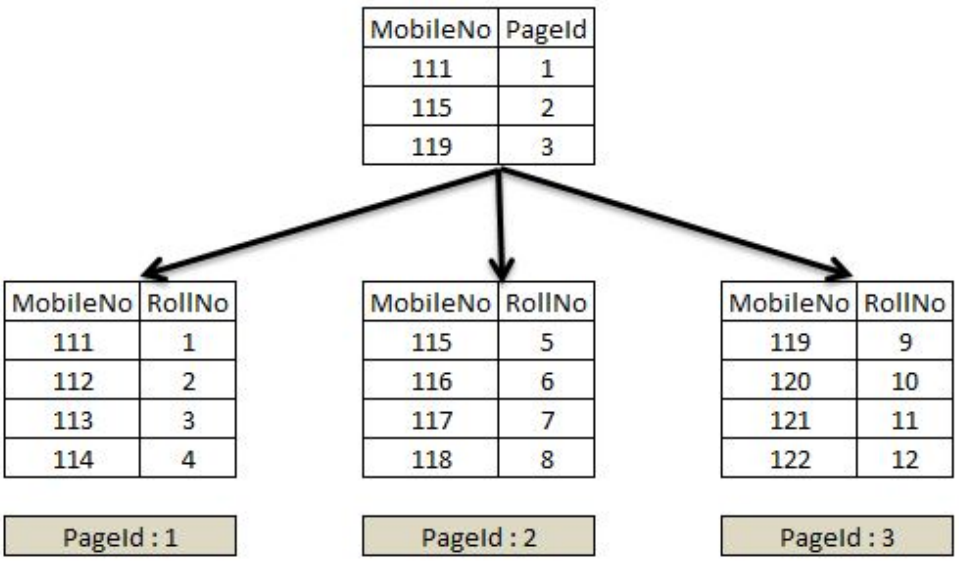


Figure 2

Let's examine some facts for the unique non-clustered index:

The clustered index key is not included *in the root level of the non-clustered index*

The MobileNo column is unique, and a query with a predicate on this column can easily seek through this non-clustered B-Tree structure to reach out the exact leaf level page. Hence, there is no need to include clustered key in the root level of the non-clustered index.

Consider the query:

```
select *
 from dbo.class
 where MobileNo = '111'
```

The MobileNo column is unique and query engine can seek through the non-clustered index by navigating from root to locate the leaf level page where MobileNo value is stored for '111', as shown by arrows in Figure 3.

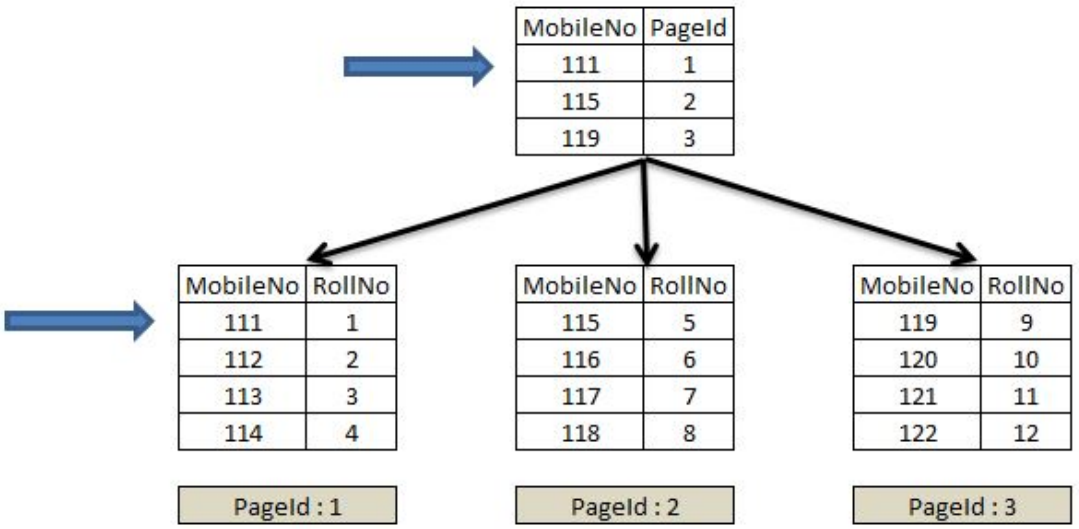


Figure 3

The clustered index key is included in the leaf level of the non-clustered index

Considering the same query.

```
select *
 from dbo.class
 where MobileNo = '111'
```

SQL can seek the leaf level page in the non-clustered index, which contains data for the column (MobileNo= 111) as shown in **Figure 4**.

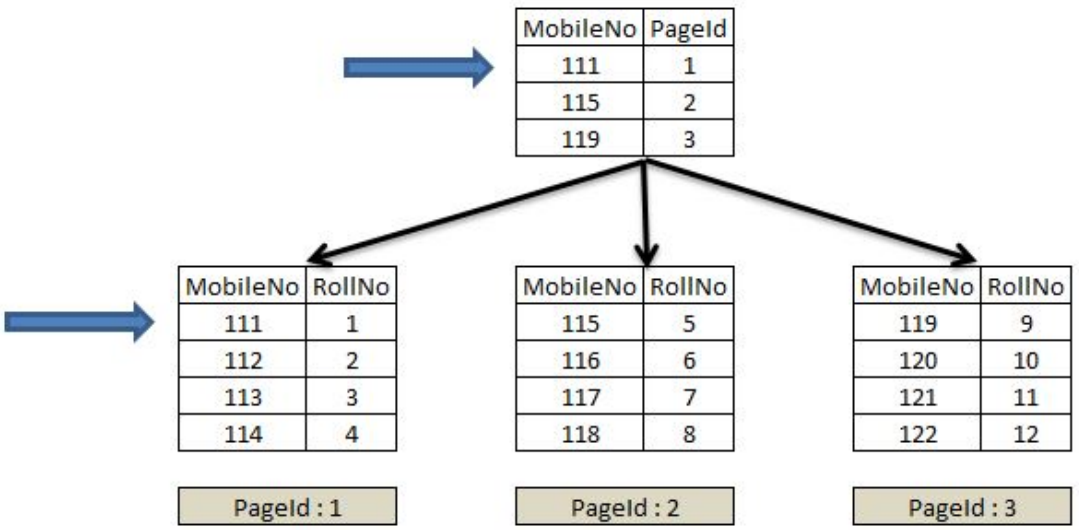


Figure 4

But what about the data of other columns? If we observe the execution plan for the SELECT query we can see the key lookup, as shown in Figure 5.

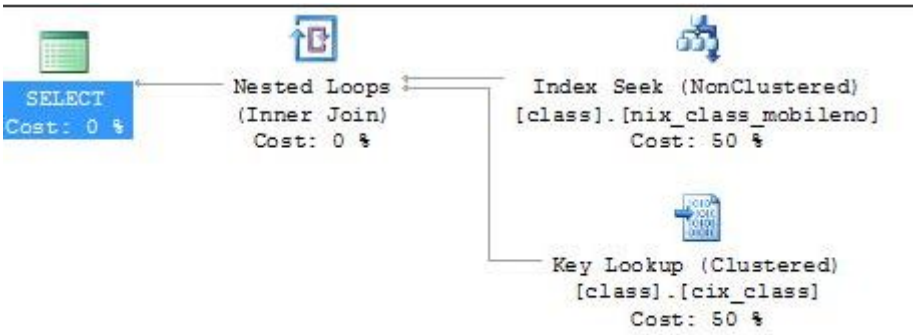


Figure 5

So what is this key lookup? This is the lookup that SQL Server has performed on the clustered index with the help of clustered index key (Rollno) in the non-clustered index leaf level page. As shown in Figure 6, the query engine locates the leaf level page with the MobileNo value of '111' and finds the corresponding RollNo value. Since RollNo is the clustered index key, the query engine uses this value and seeks through the clustered index B-Tree to fetch the values for other columns of the table.

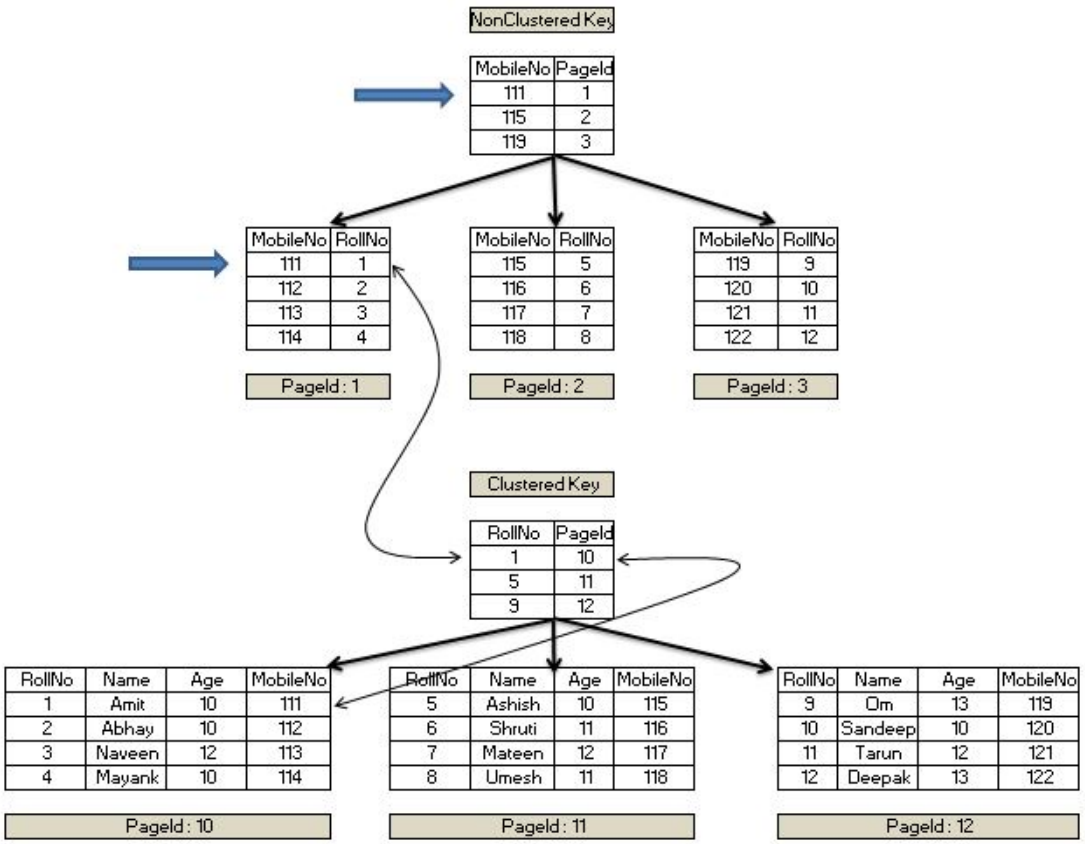


Figure 6

Non-unique non-clustered index

Now create a non-unique non-clustered index on the table, dbo.class, with Age as the index key.

```
create nonclustered index nix_NU_demo_mobile on dbo.Class(Age)
```

A non-unique non-clustered index is created with the non-clustered index key being added to the root, intermediate and leaf level pages.The interesting fact is that the clustered index key is also added to leaf and non-leaf level pages of the non-unique non-clustered index as shown in Figure 7.

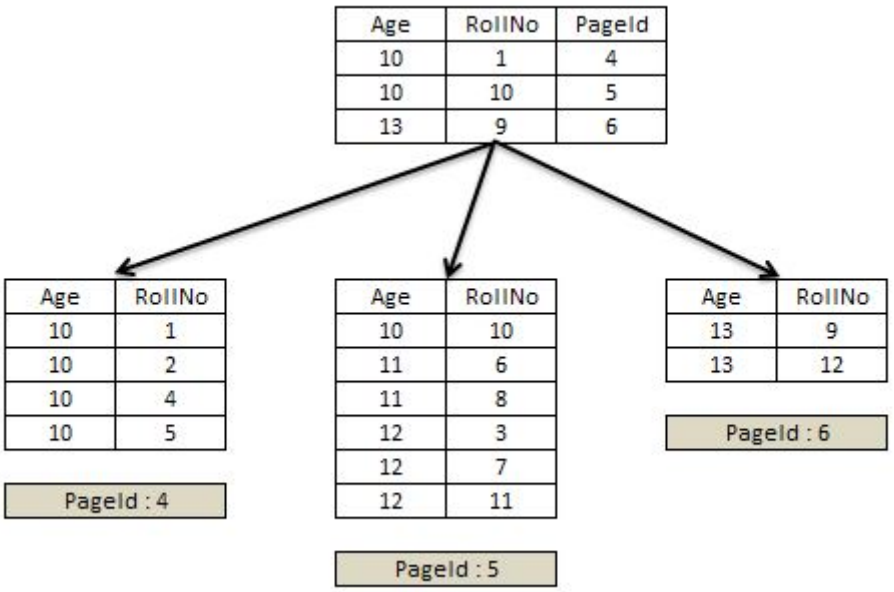


Figure 7

Let's examine some facts for this non-unique non-clustered index.

The clustered index key is added to leaf and non-leaf level pages of non-unique non-clustered index.

It is clear after reading the relationship of the clustered index and unique non-clustered index above, that the clustered index key is added to just the leaf level page of a unique non-clustered index and not the non-leaf level pages.

Considering the same structure for the non-unique non-clustered index, where the clustered index key, RollNo, is added to just leaf level page of the non-clustered index as shown in Figure 7. Let's see how SQL engine will handle below update query.

```
update dbo.class
 set age = 12
 where age = 10
 AND rollno = 10
```

This query will update the leaf level pages of the non-clustered index as we are modifying the non-clustered index key (i.e. Age). With the help of first predicate, Age=10, SQL Server has to scan the leaf level pages to know whether the second predicate, Rollno=10 matches, as shown in Figure 8. In this case, it will read two pages, but with a large table, the number of IO operations might be huge (refer to Figure 8).

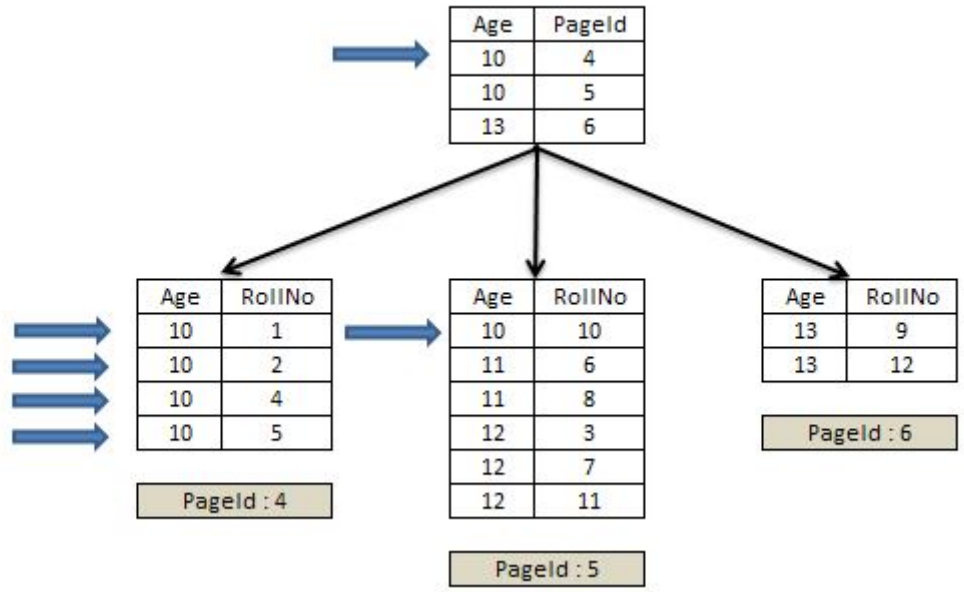


Figure 8

Hence, to perform an efficient seek over the non-unique non-clustered index, SQL Server adds the clustered index key to the root pages and intermediate levels of the non-unique non-clustered index as shown in Figure 9.

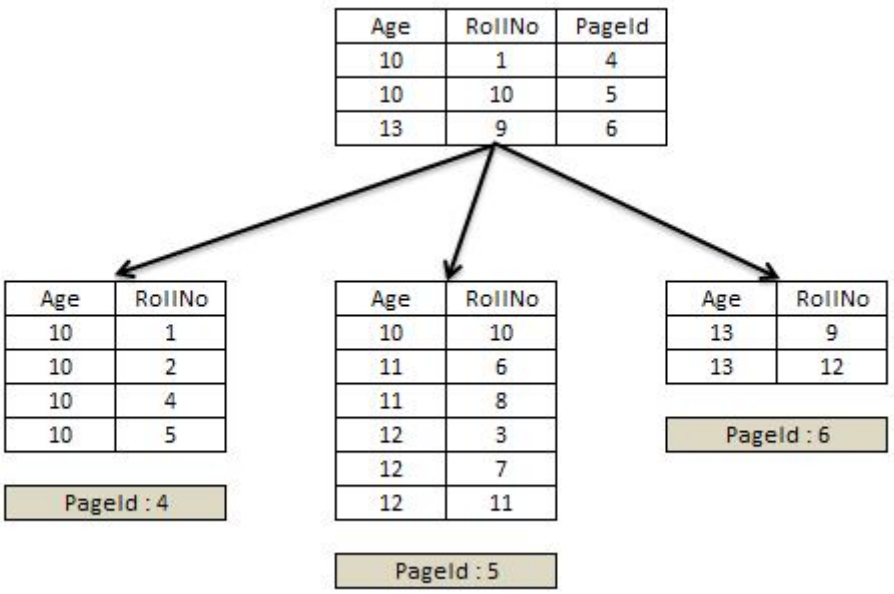


Figure 9

Consider the same update query:

```
update dbo.class
 set age = 12
 where age = 10
 AND rollno = 10
```

Since the clustered index key is added to the root level of the non-clustered index, SQL Server can easily seek the exact leaf level page by traversing through root page where it matches Age = 10 and RollNo = 10, as shown in Figure 10.

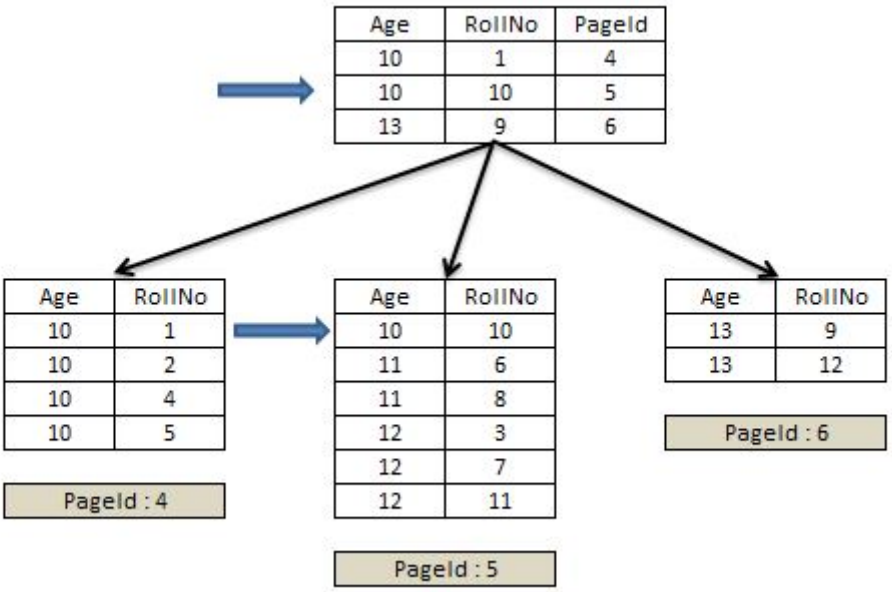


Figure 10

In Figure 10, we see that SQL Server was able to seek the exact leaf level page from the root page due the inclusion of clustered key in the root page. This helped SQL Server to uniquely identify the leaf level page with the data Age=10 and RollNo = 10.

The clustered index key is added to the leaf and non-leaf levels in non-unique non-clustered indexes for maintaining uniqueness to identify lower level B-Tree pages for reduced IO's.

Analyzing Index Internals in SQL Server Management Studio

We will see the clustered and non-clustered index details of the Sales.SalesOrderDetail table in the Adventureworks2012 database. We will see the index details for the table Sales.SalesOrderDetail with the help of SQL queries listed in the below steps:

Note: For the purpose of this demo, I will use AdventureWorks2012 database (http://msftdbprodsamples.codeplex.com/releases/view/93587)

Execute the query below to see the indexes present on table Sales.SalesOrderDetail

```
SELECT OBJECT_NAME(object_id),*
 FROM [sys].[indexes]
 WHERE OBJECT_NAME(object_id) = 'SalesOrderDetail';
```

In the result shown in Figure 11, we see the below indexes on the Sales.SalesOrderDetail table

- Clustered index
- Unique non-clustered index
- Non-unique non-clustered index

```
SELECT OBJECT_NAME(object_id), *
FROM [sys].[indexes]
WHERE OBJECT_NAME(object_id) = 'SalesOrderDetail';
```

100 %

Results | Messages | Execution plan

	(No column name)	object_id	name	index_id	type	type_desc	is_unique
1	SalesOrderDetail	1154103152	PK_SalesOrderDetail_SalesOrderID_SalesOrderDet...	1	1	CLUSTERED	1
2	SalesOrderDetail	1154103152	AK_SalesOrderDetail_rowguid	2	2	NONCLUSTERED	1
3	SalesOrderDetail	1154103152	IX_SalesOrderDetail_ProductID	3	2	NONCLUSTERED	0

Zoom in | Open in new window

Figure 11

[Note: is_unique column is highlighted to help the readers to identify unique and non-unique indexes]

We will explore the contents of each index below.

1. Clustered index

The DBCC IND command will help us to locate a Page on the index(refer to Query 2)

```
/*Query 2*/ DBCC IND('AdventureWorks2012','Sales.SalesOrderDetail',1)
```

> *Here is the description of parameters passed to DBCC IND command:*
>
> *'AdventureWorks2012' - Database name*
>
> *'Sales.SalesOrderDetail' - Table Name*
>
> *1- clustered index ID (from figure.11)*

```
dbcc ind('AdventureWorks2012','Sales.SalesOrderDetail',1) -- 1 is the Index ID of Clustered Index
```

100 %

Results | Messages

	PageFID	PagePID	IAMFID	IAMPID	ObjectID	IndexID	PartitionNumber	PartitionID	iam_chain_type	PageType	IndexLevel	NextPageFID	NextPagePID
1	1	810	NULL	NULL	1154103152	1	1	72057594046971904	In-row data	10	NULL	0	0
2	1	1431	1	810	1154103152	1	1	72057594046971904	In-row data	2	2	0	0
3	1	10608	1	810	1154103152	1	1	72057594046971904	In-row data	1	0	1	10609
4	1	10609	1	810	1154103152	1	1	72057594046971904	In-row data	1	0	1	10610
5	1	10610	1	810	1154103152	1	1	72057594046971904	In-row data	1	0	1	10611
6	1	10611	1	810	1154103152	1	1	72057594046971904	In-row data	1	0	1	10612
7	1	10612	1	810	1154103152	1	1	72057594046971904	In-row data	1	0	1	10613
8	1	10613	1	810	1154103152	1	1	72057594046971904	In-row data	1	0	1	10614
9	1	10614	1	810	1154103152	1	1	72057594046971904	In-row data	1	0	1	10615

Zoom in | Open in new window

Figure 12

On executing Query 2, we are presented with all the pages of the index. However for the scope of this article, we will select a root page (viz. the page with the IndexLevel = 2). By using DBCC Page command for root level page (viz. PagePid = 1431), we can see the page contents as shown in Figure 13.

```
/*Query 3*/DBCC PAGE('AdventureWorks2012',1,1431,3) with tableresults
```

> *Here is the description of parameters passed to DBCC Page command:*
>
> *'AdventureWorks2012' - Database name*
>
> *1 - Page File ID (as shown in fig.12)*
>
> *1431 - PagePID (the root level page as shown in fig.12)*
>
> *3 - Print option, using the print option 3 gets page header plus detailed per-row interpretation.*

'

```
dbcc page('AdventureWorks2012',1,1431,3) with tableresults
```

100 %

Results | Messages

6	BUFFER:	BUF @0x07772DE8	bcputicks	0
7	BUFFER:	BUF @0x07772DE8	bsampleCount	0
8	BUFFER:	BUF @0x07772DE8	bUse1	22049

	FileId	PageId	Row	Level	ChildFileId	ChildPageId	SalesOrderID (key)	SalesOrderDetailID (key)	KeyHashValue	Row Size
1	1	1431	0	2	1	10824	NULL	NULL	NULL	15
2	1	1431	1	2	1	10936	50178	30887	NULL	15
3	1	1431	2	2	1	10880	53166	46035	NULL	15
4	1	1431	3	2	1	10768	57058	61157	NULL	15
5	1	1431	4	2	1	10800	61245	76918	NULL	15
6	1	1431	5	2	1	10856	65760	91650	NULL	15
7	1	1431	6	2	1	11448	70016	106168	NULL	15

Zoom in | *Open in new window*

Figure 13

From Figure 13, we observe that SalesOrderID and SalesOrderDetailId (highlighted) are the clustered index keys for Sales.SalesOrderDetail table.

2. Unique non-clustered index

We will use DBCC IND command to locate root and leaf level Page on this index (refer to Query 4).

```
/*Query 4*/ DBCC IND('AdventureWorks2012','Sales.SalesOrderDetail',2)
```

> *Here is the description of parameters passed to DBCC IND command:*
>
> *'AdventureWorks2012' - Database name*
>
> *'Sales.SalesOrderDetail' - Table Name*
>
> *2- unique non-clustered index ID (from figure 11)*

```
dbcc ind('AdventureWorks2012','Sales.SalesOrderDetail',2) -- 2 is the Index ID of Unique NonClustered Index
```

Results | Messages

PageFID	PagePID	IAMFID	IAMPID	ObjectID	IndexID	PartitionNumber	PartitionID	iam_chain_type	PageType	IndexLevel	NextPageFID
1	5200	NULL	NULL	1154103152	2	1	72057594052345856	In-row data	10	NULL	0
1	5199	1	5200	1154103152	2	1	72057594052345856	In-row data	2	2	0
1	528	1	5200	1154103152	2	1	72057594052345856	In-row data	2	0	1
1	529	1	5200	1154103152	2	1	72057594052345856	In-row data	2	0	1
1	530	1	5200	1154103152	2	1	72057594052345856	In-row data	2	0	1
1	531	1	5200	1154103152	2	1	72057594052345856	In-row data	2	0	1
1	532	1	5200	1154103152	2	1	72057594052345856	In-row data	2	0	1
1	533	1	5200	1154103152	2	1	72057594052345856	In-row data	2	0	1

Zoom in | *Open in new window*

Figure 14

On executing Query 4, we get a list of pages on the index as shown in Figure 14. Our interest is in analyzing Root and leaf level pages. Let's consider a root level page viz. PagePID = 5199 (Index Level = 2) and leaf level page viz. PagePID = 528 (Index Level = 0). Using DBCC Page command for leaf level page (viz. PagePid = 528), we can see the page contents as shown in Figure 15.

```
/*Query 5*/DBCC PAGE('AdventureWorks2012',1,528,3) with tableresults
```

> *Here is the description of parameters passed to DBCC Page command:*
>
> *'AdventureWorks2012' - Database name*
>
> *1 - Page File ID (as shown in figure.14)*
>
> *528 - PagePID (the leaf level page as shown in figure.14)*
>
> *3 - Print option, using the print option 3 gets page header plus detailed per-row interpretation.*

```
dbcc page('AdventureWorks2012',1,528,3) with tableresults -- Leaf Level (Index level 0)
```

Results | Messages

6	BUFFER:	BUF @0x078568D0	bcputicks	92
7	BUFFER:	BUF @0x078568D0	bsampleCount	1
8	BUFFER:	BUF @0x078568D0	bUse1	22267

	FileId	PageId	Row	Level	rowguid (key)	SalesOrderID	SalesOrderDetailID	KeyHashValue	Row Size
1	1	528	0	0	988ABA69-223D-4401-970B-17264C20AEE6	66263	92910	(c89e40f154dc)	28
2	1	528	1	0	0FD4080B-49DF-45BF-9A8B-172676101E15	46647	11610	(1887aa51ec4b)	28
3	1	528	2	0	1875BC7F-1FEE-4F97-B2A0-1726A3F82796	52572	44615	(965b812503cf)	28
4	1	528	3	0	83E56C7A-52DB-43E6-B766-17272717FEFB	69606	105131	(ae48fa6d0e6c)	28

Zoom in | *Open in new window*

Figure 15

From Figure 15, we observe that the SalesOrderId and SalesOrderDetailID (Clustered Index keys) are added to leaf level along with rowguid viz. (unique non-clustered index key). Using DBCC Page command for root level page (viz. PagePid = 5199), we can see the page contents as shown in Figure 16.

```
/*Query 6*/DBCC PAGE('AdventureWorks2012',1,5199,3) with tableresults
```

> *Here is the description of parameters passed to DBCC Page command:*
>
> *'AdventureWorks2012' - Database name*
>
> *1 - Page File ID (as shown in fig.14)*
>
> *5199 - PagePID (the root level page as shown in fig.14)*
>
> *3 - Print option, using the print option 3 gets page header plus detailed per-row interpretation.*

```
dbcc page('AdventureWorks2012',1,5199,3) with tableresults -- Root Level (Index Level 2)
```

Results | Messages

BUFFER:	BUF @0x07849240	bhash	0x00000000	
BUFFER:	BUF @0x07849240	bpageno	(1:5199)	
BUFFER:	BUF @0x07849240	bdbid	11	
BUFFER:	BUF @0x07849240	breferences	0	
BUFFER:	BUF @0x07849240	bcputicks	0	
BUFFER:	BUF @0x07849240	bsampleCount	0	
BUFFER:	BUF @0x07849240	bUse1	22368	

	FileId	PageId	Row	Level	ChildFileId	ChildPageId	rowguid (key)	KeyHashValue	Row Size
1	1	5199	0	2	1	20096	NULL	NULL	23
2	1	5199	1	2	1	20192	0886F961-F7B5-4CA0-8B96-B607769258B9	NULL	23

Zoom in | Open in new window

Figure 16

From Figure 16, we see that the root level page has a rowguid (highlighted), and there is no clustered index key at the root level of the unique non-clustered index.

3. Non-unique non-clustered index

We will use DBCC IND command to locate root and leaf level Page on this index (refer to Query 7)

```
/*Query 7*/ DBCC IND('AdventureWorks2012','Sales.SalesOrderDetail',3)
```

> *Here is the description of parameters passed to DBCC IND command:*
>
> *'AdventureWorks2012' - Database name*
>
> *'Sales.SalesOrderDetail' - Table Name*
>
> *3- non-unique non-clustered index ID (from fig.11)*

```
dbcc ind('AdventureWorks2012','Sales.SalesOrderDetail',3) -- 3 is the Index ID of Non Unique NonClustered Index
```

Results | Messages

	PageFID	PagePID	IAMFID	IAMPID	ObjectID	IndexID	PartitionNumber	PartitionID	iam_chain_type	PageType	IndexLevel	NextPageFID	NextPa
193	1	6136	1	5202	1154103152	3	1	72057594052411392	In-row data	2	1	0	0
194	1	6160	1	5202	1154103152	3	1	72057594052411392	In-row data	2	0	1	6161
195	1	6161	1	5202	1154103152	3	1	72057594052411392	In-row data	2	0	1	6162
196	1	6162	1	5202	1154103152	3	1	72057594052411392	In-row data	2	0	1	6163
197	1	6163	1	5202	1154103152	3	1	72057594052411392	In-row data	2	0	1	6164
198	1	6164	1	5202	1154103152	3	1	72057594052411392	In-row data	2	0	1	6165
199	1	6165	1	5202	1154103152	3	1	72057594052411392	In-row data	2	0	1	6166
200	1	6166	1	5202	1154103152	3	1	72057594052411392	In-row data	2	0	1	6167
201	1	6167	1	5202	1154103152	3	1	72057594052411392	In-row data	2	0	1	6168

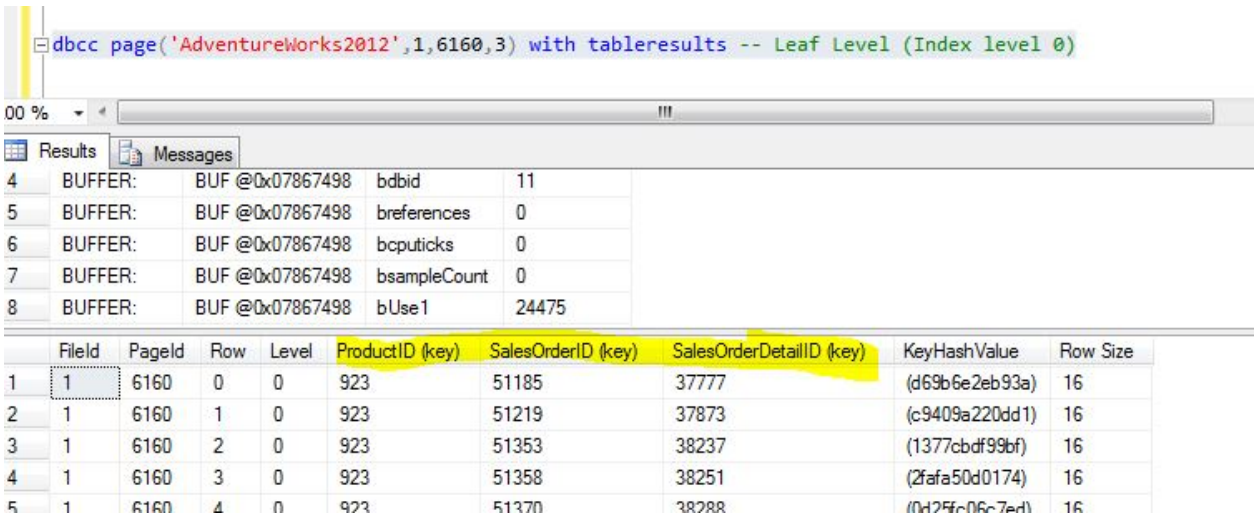
Zoom in | Open in new window

Figure 17

On executing the Query 7, we get a list of pages on the index as shown in Figure 17. Our interest is in analyzing the root and leaf level pages. Let's consider a root level page, viz. PagePID = 6136 (Index Level = 1), and leaf level page, viz. PagePID = 6160 (Index Level = 0). Using DBCC Page command for the leaf level page (viz. PagePid = 6160), we can see the page contents as shown in Figure 18.

```
/*Query 8*/DBCC PAGE('AdventureWorks2012',1,6160,3) with tableresults
```

> *Here is the description of parameters passed to DBCC Page command:*
>
> *'AdventureWorks2012' - Database name*
>
> *1 - Page File ID (as shown in figure.17)*
>
> *6160 - PagePID (the leaf level page as shown in figure.17)*
>
> *3 - Print option, using the print option 3 gets page header plus detailed per-row interpretation.*



Zoom in | Open in new window

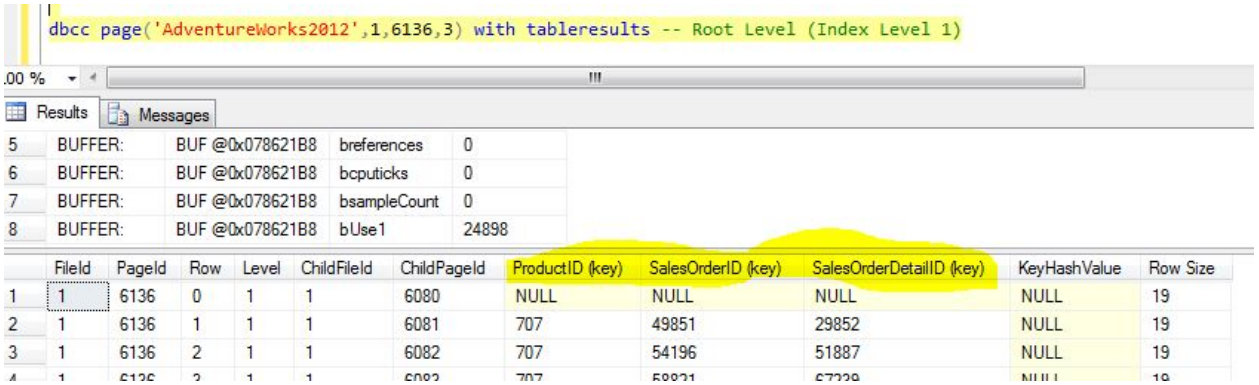
Figure 18

From Figure 18, we observe that the SalesOrderId and SalesOrderDetailID (Clustered Index keys) are added to the leaf level along with ProductId (non-unique non-clustered index key), as highlighted in Figure 18.

Using DBCC Page command for root level page (viz. PagePid = 6136), we can see the page contents as shown in Figure 19.

```
/*Query 8*/DBCC PAGE('AdventureWorks2012',1,6136,3) with tableresults
```

> *Here is the description of parameters passed to DBCC Page command:*
>
> *'AdventureWorks2012' - Database name*
>
> *1 - Page File ID (as shown in figure.17)*
>
> *6136 - PagePID (the leaf level page as shown in figure.17)*
>
> *3 - Print option, using the print option 3 gets page header plus detailed per-row interpretation.*



Zoom in | *Open in new window*

Figure.19

From Figure 19 we see that the clustered index keys, SalesOrderId and SalesOrderDetailID, along with non-unique non-clustered index key (ProductID) are added to root level of the non-unique non-clustered index.(highlighted).

In this blog, we saw some of the interesting and unexplored facts of non-clustered indexes. These facts may not be directly useful for your day to day work but understanding the concept will help you take better decisions on SQL indexing.

Hope this blog will be useful for you in understanding the internals of non-clustered index keys.

Reference

https://www.youtube.com/watch?v=y_bl9dArtmA