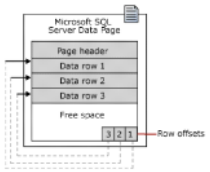
Pages vs Extents

Friday, February 8, 2019 6:44 PM

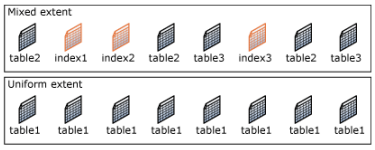
Pages

In SQL Server, the page size is 8-KB. This means SQL Server databases have 128 pages per megabyte. Each page begins with a 96-byte header that is used to store system information about the page. This information includes the page number, page type, the amount of free space on the page, and the allocation unit ID of the object that owns the page.

From <https://docs.microsoft.com/en-us/sql/relational-databases/pages-and-extents-architecture-guide?view=sql-server-2017>



- **Uniform** extents are owned by a single object; all eight pages in the extent can only be used by the owning object.
- **Mixed** extents are shared by up to eight objects. Each of the eight pages in the extent can be owned by a different object.



00-DatabasePageAll...-OFFICE\dellp (55))

```
Use Northwind;
Go

-- Examine data page allocations in the following tables using the
-- new DMF sys.dm_db_database_page_allocations
SELECT *
FROM sys.dm_db_database_page_allocations(db_id('Northwind'),
object_id('Employees'), 1, null, 'DETAILED')
WHERE page_type_desc = 'DATA_PAGE';

SELECT *
FROM sys.dm_db_database_page_allocations(db_id('Northwind'),
object_id('Customers'), 1, null, 'DETAILED')
WHERE page_type_desc = 'DATA_PAGE'
Order By extent_page_id;

SELECT *
FROM sys.dm_db_database_page_allocations(db_id('Northwind'),
object_id('Order_Details'), 1, null, 'DETAILED')
WHERE page_type_desc = 'DATA_PAGE'
Order By extent_page_id;
```

161 %

Results | Messages

	allocation_unit_type	allocation_unit_type_desc	data_clone_id	clone_state	clone_state_desc	extent_file_id	extent_page_id	allocated_page_iam_file_id	allocated_page_iam_page_id	allocated_page_file_id	allocated_page_page_id	is_allocated	is_iam_page	is_mixed_page_allocation	page_free_space_percent	page_type	page_type_desc	page_level	next_page_file_id	next_page_page_id	previous_page_file_id	previous_page_page_id	is_page_comp
1	368 1	IN_ROW_DATA	NULL	NULL	SECONDARY	1	448	1	456	1	448	1	0	0	NULL	1	DATA_PAGE	0	NULL	NULL	NULL	NULL	0

	allocation_unit_type	allocation_unit_type_desc	data_clone_id	clone_state	clone_state_desc	extent_file_id	extent_page_id	allocated_page_iam_file_id	allocated_page_iam_page_id	allocated_page_file_id	allocated_page_page_id	is_allocated	is_iam_page	is_mixed_page_allocation	page_free_space_percent	page_type	page_type_desc	page_level	next_page_file_id	next_page_page_id	previous_page_file_id	previous_page_page_id	is_page_comp
1	5120 1	IN_ROW_DATA	NULL	NULL	SECONDARY	1	400	1	378	1	400	1	0	0	NULL	1	DATA_PAGE	0	1	402	NULL	NULL	0
2	5120 1	IN_ROW_DATA	NULL	NULL	SECONDARY	1	400	1	378	1	402	1	0	0	NULL	1	DATA_PAGE	0	1	403	1	400	0
3	5120 1	IN_ROW_DATA	NULL	NULL	SECONDARY	1	400	1	378	1	403	1	0	0	NULL	1	DATA_PAGE	0	NULL	NULL	1	402	0

	i	allocation_unit_type	allocation_unit_type_desc	data_clone_id	clone_state	clone_state_desc	extent_file_id	extent_page_id	allocated_page_iam_file_id	allocated_page_iam_page_id	allocated_page_file_id	allocated_page_page_id	is_allocated	is_iam_page	is_mixed_page_allocation	page_free_space_percent	page_type	page_type_desc	page_level	next_page_file_id	next_page_page_id	previous_page_file_id	previous_page_page_id	is_page_c
97	80	1	IN_ROW_DATA	NULL	NULL	SECONDARY	1	216	1	117	1	223	1	0	0	NULL	1	DATA_PAGE	0	1	299	1	222	0
98	80	1	IN_ROW_DATA	NULL	NULL	SECONDARY	1	248	1	117	1	248	1	0	0	NULL	1	DATA_PAGE	0	1	249	1	299	0
99	80	1	IN_ROW_DATA	NULL	NULL	SECONDARY	1	248	1	117	1	249	1	0	0	NULL	1	DATA_PAGE	0	1	250	1	248	0
100	80	1	IN_ROW_DATA	NULL	NULL	SECONDARY	1	248	1	117	1	250	1	0	0	NULL	1	DATA_PAGE	0	1	251	1	249	0
101	80	1	IN_ROW_DATA	NULL	NULL	SECONDARY	1	248	1	117	1	251	1	0	0	NULL	1	DATA_PAGE	0	1	252	1	250	0
102	80	1	IN_ROW_DATA	NULL	NULL	SECONDARY	1	248	1	117	1	252	1	0	0	NULL	1	DATA_PAGE	0	1	253	1	251	0
103	80	1	IN_ROW_DATA	NULL	NULL	SECONDARY	1	248	1	117	1	253	1	0	0	NULL	1	DATA_PAGE	0	1	300	1	252	0
104	80	1	IN_ROW_DATA	NULL	NULL	SECONDARY	1	248	1	117	1	254	1	0	0	NULL	1	DATA_PAGE	0	1	255	1	300	0
105	80	1	IN_ROW_DATA	NULL	NULL	SECONDARY	1	248	1	117	1	255	1	0	0	NULL	1	DATA_PAGE	0	1	272	1	254	0
106	80	1	IN_ROW_DATA	NULL	NULL	SECONDARY	1	272	1	117	1	272	1	0	0	NULL	1	DATA_PAGE	0	1	273	1	255	0
107	80	1	IN_ROW_DATA	NULL	NULL	SECONDARY	1	272	1	117	1	273	1	0	0	NULL	1	DATA_PAGE	0	1	274	1	272	0
108	80	1	IN_ROW_DATA	NULL	NULL	SECONDARY	1	272	1	117	1	274	1	0	0	NULL	1	DATA_PAGE	0	1	275	1	273	0
109	80	1	IN_ROW_DATA	NULL	NULL	SECONDARY	1	272	1	117	1	275	1	0	0	NULL	1	DATA_PAGE	0	1	276	1	274	0

Query executed successfully. (local) (13.0 SP1) LABGURU-OFFICE\dellp (55) Northwind 00:00:00 208 rows

Human Index #1

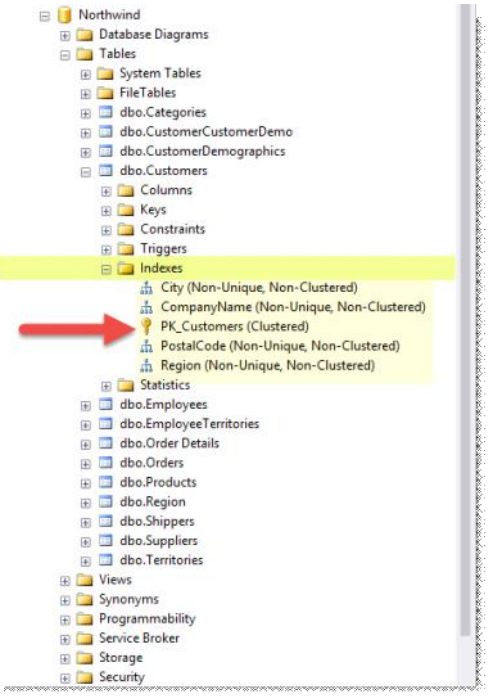
Friday, February 8, 2019 7:43 PM

Contents

Contents

Data Index #2

Tuesday, May 23, 2017 11:10 AM



Impacts on Searches

- Selectivity
  - A measure of how many rows are returned compared to the total number of rows
  - High selectivity means a small number of rows when related to the total number of rows
- Density
  - A measure of the lack of uniqueness of data in the table
  - High density indicates a large number of duplicates
- Index Depth
  - Number of levels within the index
  - Common misconception that indexes are deep

Figure 1.1 Index Leaf Nodes and Corresponding Table Data

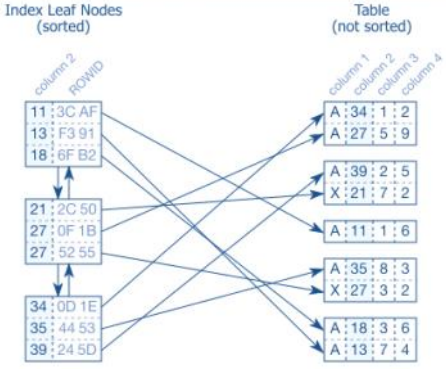
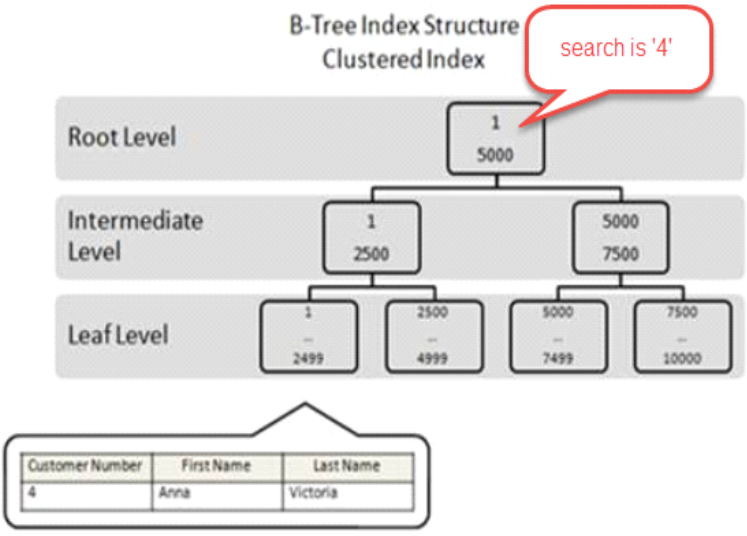


Figure 1.1 illustrates the index leaf nodes and their connection to the table data. Each index entry consists of the indexed columns (the key, column 2) and refers to the corresponding table row (via ROWID or RID).



HEAP Versus Clustered

Tuesday, May 23, 2017 6:09 PM

An Heap is good for small Data or Data that is NOT Queried Often

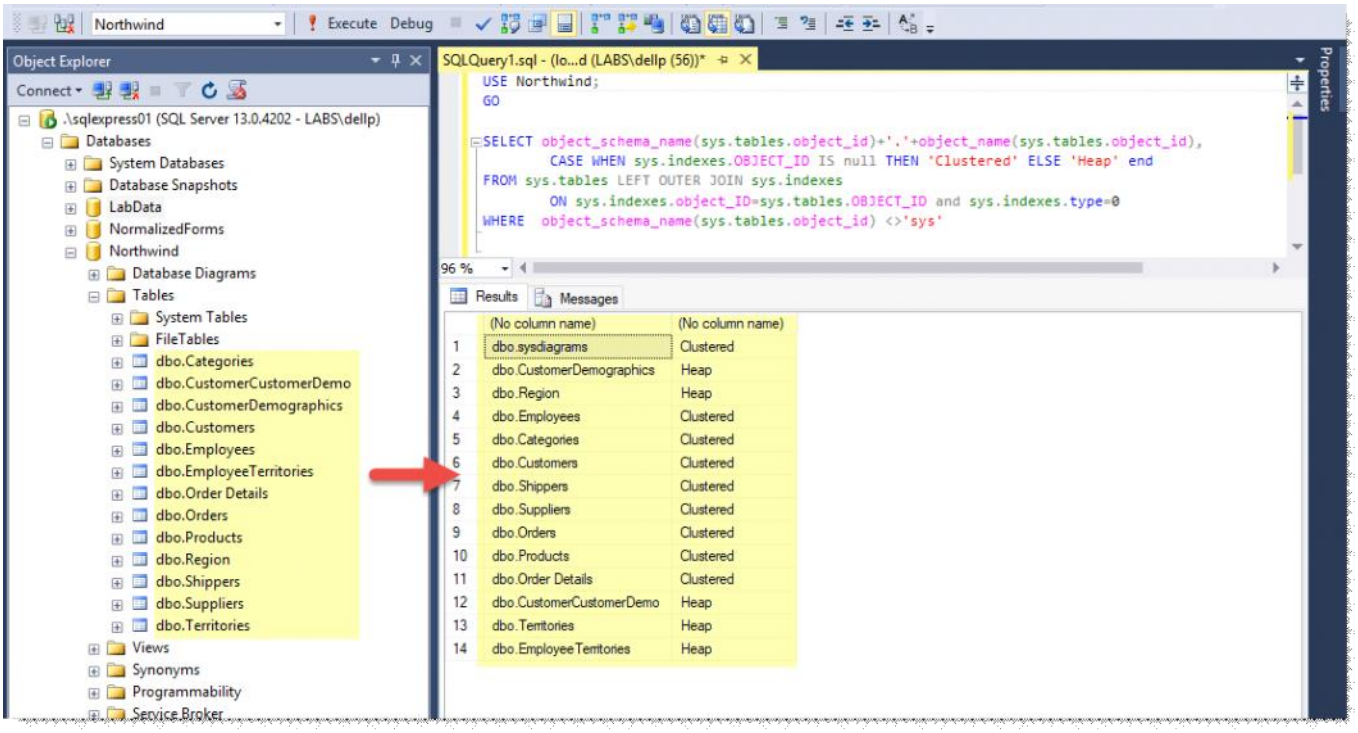
HEAP

- Data is not stored in any particular order
- Specific data can not be retrieved quickly, unless there are also non-clustered indexes
- Data pages are not linked, so sequential access needs to refer back to the index allocation map (IAM) pages
- Since there is no clustered index, additional time is not needed to maintain the index
- Since there is no clustered index, there is not the need for additional space to store the clustered index tree
- These tables have a index_id value of 0 in the sys.indexes catalog view

Clustered Table

- Data is stored in order based on the clustered index key
- Data can be retrieved quickly based on the clustered index key, if the query uses the indexed columns
- Data pages are linked for faster sequential access
- Additional time is needed to maintain clustered index based on INSERTS, UPDATES and DELETES
- Additional space is needed to store clustered index tree
- These tables have a index_id value of 1 in the sys.indexes catalog view

Send HeapVsClustered.sql



Heap Forward-Pointers

Tuesday, May 23, 2017 6:51 PM

Files: PersonHeapForwardPointers.sql
Student File: spForwardPointers.sql

Create Heap Table

RUN EACH STEP SEPARATLY

spForwardPointers.s...ls (LABS\dellp (51)) | PersonHeapForwardP...(LABS\dellp (55))*

```sql
Use Schools;
Go

Create Procedure spForwardPointers
    @Table VarChar(15)
AS
-----------------------------------------------------------------
--Check Forward Pointers
-----------------------------------------------------------------
    Select Database_id,
    Index_id,
    Object_name([object_id]) as TableName,
    Case when SI.name is null then 'HEAP' else SI.name End as IndexName,
    Index_type_desc,
    Avg_fragmentation_in_percent,
          forwarded_record_count
    From sys.dm_db_index_physical_stats(db_id(),object_id(@Table),null,null,'det
    Inner join sys.sysindexes AS SI on SDDIPS.[object_id] = SI.id
    AND SDDIPS.index_id = SI.indid
    Where index_level = 0
Go
```

PersonHeapForwardP...(LABS\dellp (55))* | HeapVsClustered.sq...d (LABS\dellp (53))

```sql
-----------------------------------------------------------
--RUN EACH STEP SEPARATELY
-----------------------------------------------------------
--Step One
-----------------------------------------------------------
CREATE TABLE PersonHeap
(
    PersonID uniqueidentifier DEFAULT NEWID(),
    FirstName VarChar(25) NOT NULL,
    LastName VarChar(25) NOT NULL,
    Address VarChar(50) NULL,
    City VarChar(25) NULL,
    State VarChar(2) NOT NULL,
    ZipCode VarChar(25) NULL,
)
GO
Exec spForwardPointers 'PersonHeap';

--Step Two
-----------------------------------------------------------
INSERT INTO PersonHeap VALUES
(NewID(), 'Mickey', 'Mouse', '123 Main Street', 'Sacramento', 'CA', '95830');
GO 1000
Select * From PersonHeap

--Step Three
-----------------------------------------------------------
UPDATE PersonHeap
SET Address = '12345678901234567890123456789012345678901234', City= '12345678901234567890123456789012345
WHERE LastName = 'Mouse';
GO
Select * From PersonHeap
Exec spForwardPointers 'PersonHeap';


--Step Four / Fix Forward Pointers
-----------------------------------------------------------
Alter table PersonHeap rebuild
Exec spForwardPointers 'PersonHeap';

Drop Table PersonHeap
```

96 % | Results | Messages

	Database_id	Index_id	TableName	IndexName	Index_type_desc	Avg_fragmentation_in_percent	forwarded_record_count
1	10	0	PersonHeap	HEAP	HEAP	0	0

96 % | Results | Messages

	Database_id	Index_id	TableName	IndexName	Index_type_desc	Avg_fragmentation_in_percent	forwarded_record_count
1	10	0	PersonHeap	HEAP	HEAP	0	265

Query executed successfully. | (local)\sqlexpress01 (13.0 ... | LABS\dellp (55) | Schools | 00:00:06 | 2001 rows

96 % | Results | Messages

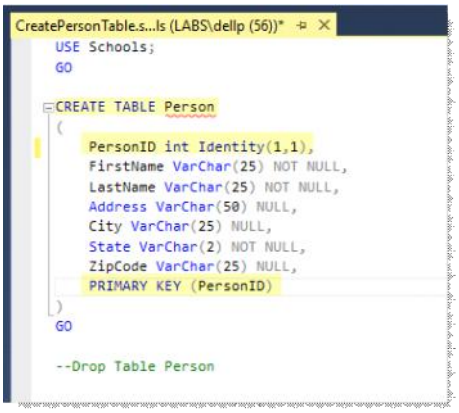
	Database_id	Index_id	TableName	IndexName	Index_type_desc	Avg_fragmentation_in_percent	forwarded_record_count
1	10	0	PersonHeap	HEAP	HEAP	0	0

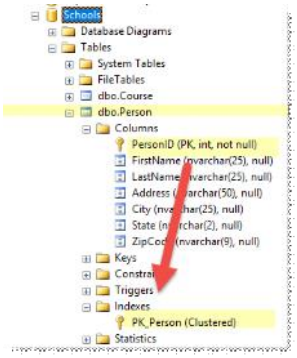
Simple Index

Tuesday, May 23, 2017 10:49 AM

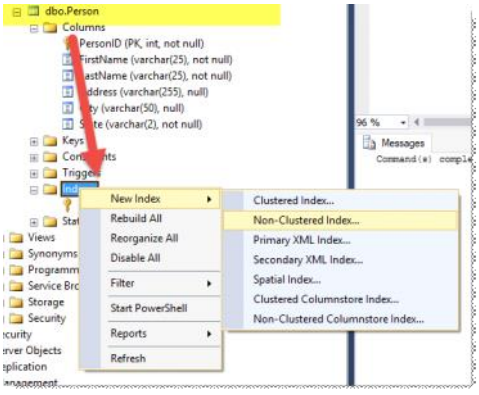
CreatePerson.Sql

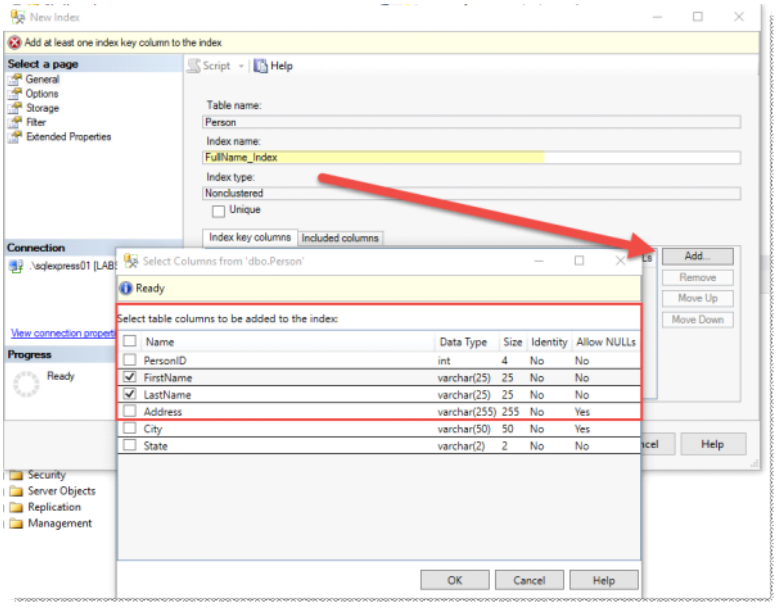
- [ ] Create Table Person With Primary Key (PersonID Identity)
- [ ] Check Index on PersonID

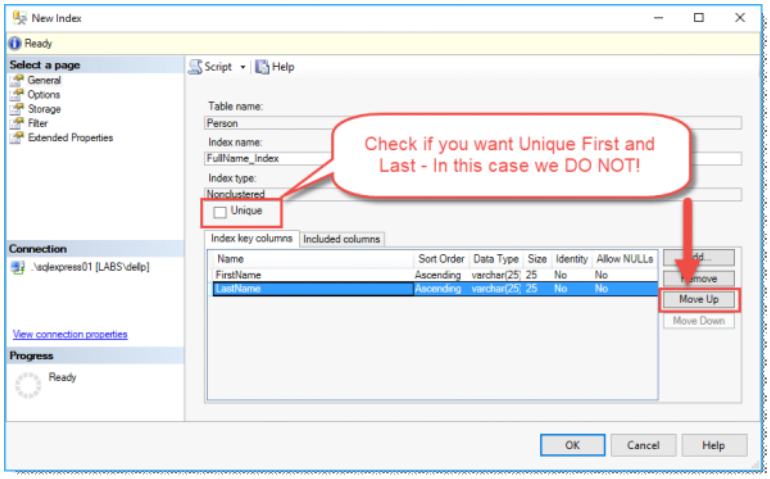


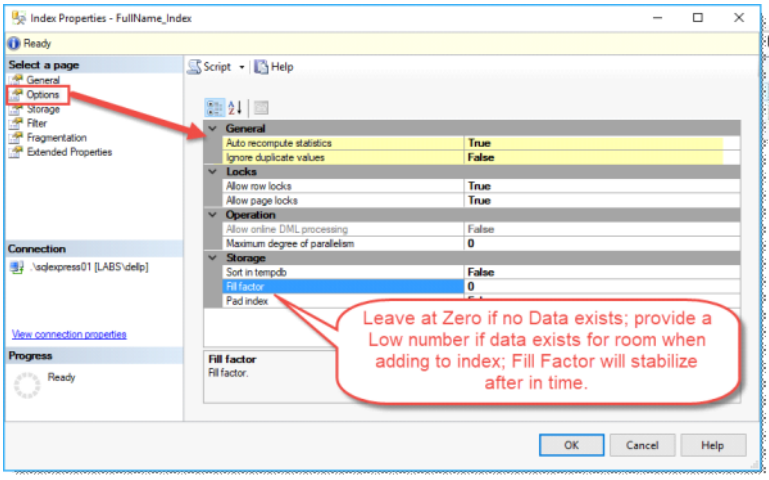


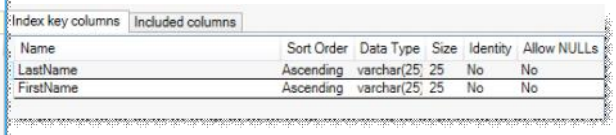
- [ ] Create Name Index
- [ ] Check Fragmentation (Properties)



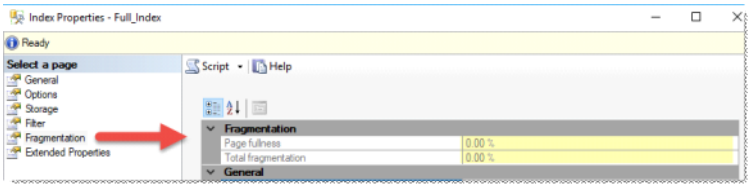








Create Index, then come back and check current Fragmentation



Disabling an Index

Tuesday, May 23, 2017 7:45 PM

Disabling an Index

ALTER INDEX FullName_Index ON Person DISABLE;
GO

Re-Enable DISABLED Index

ALTER INDEX FullName_Index ON Person REBUILD;
GO

Dropping an Index

DROP INDEX FullName_Index ON Person;
GO

Find Indexes (Table)

Tuesday, May 23, 2017 8:04 PM

FindIndexes.sql

SQLQuery7.sql - (lo...d (LABS\dellp (61))* | CreatePersonTable.s...ls (LABS\dellp (56))

```sql
USE Northwind;
GO

EXEC sp_helpindex 'Customers';
```

96 %

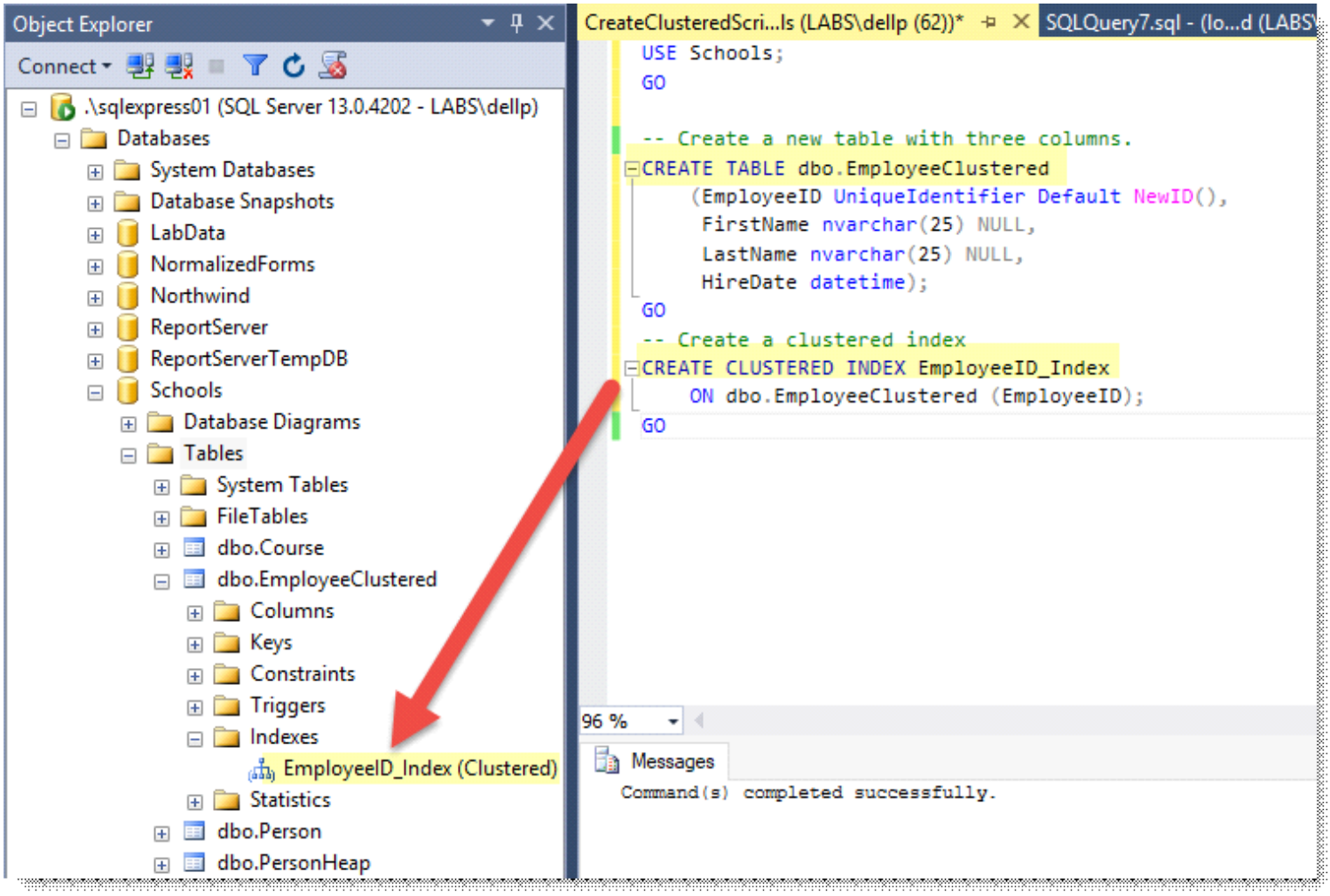
Results | Messages

	index_name	index_description	index_keys
1	City	nonclustered located on PRIMARY	City
2	CompanyName	nonclustered located on PRIMARY	CompanyName
3	PK_Customers	clustered, unique, primary key located on PRIMARY	CustomerID
4	PostalCode	nonclustered located on PRIMARY	PostalCode
5	Region	nonclustered located on PRIMARY	Region

Script Clustered Index

Tuesday, May 23, 2017 8:11 PM

One Clustered Index / Table



Script Non-Clustered Index

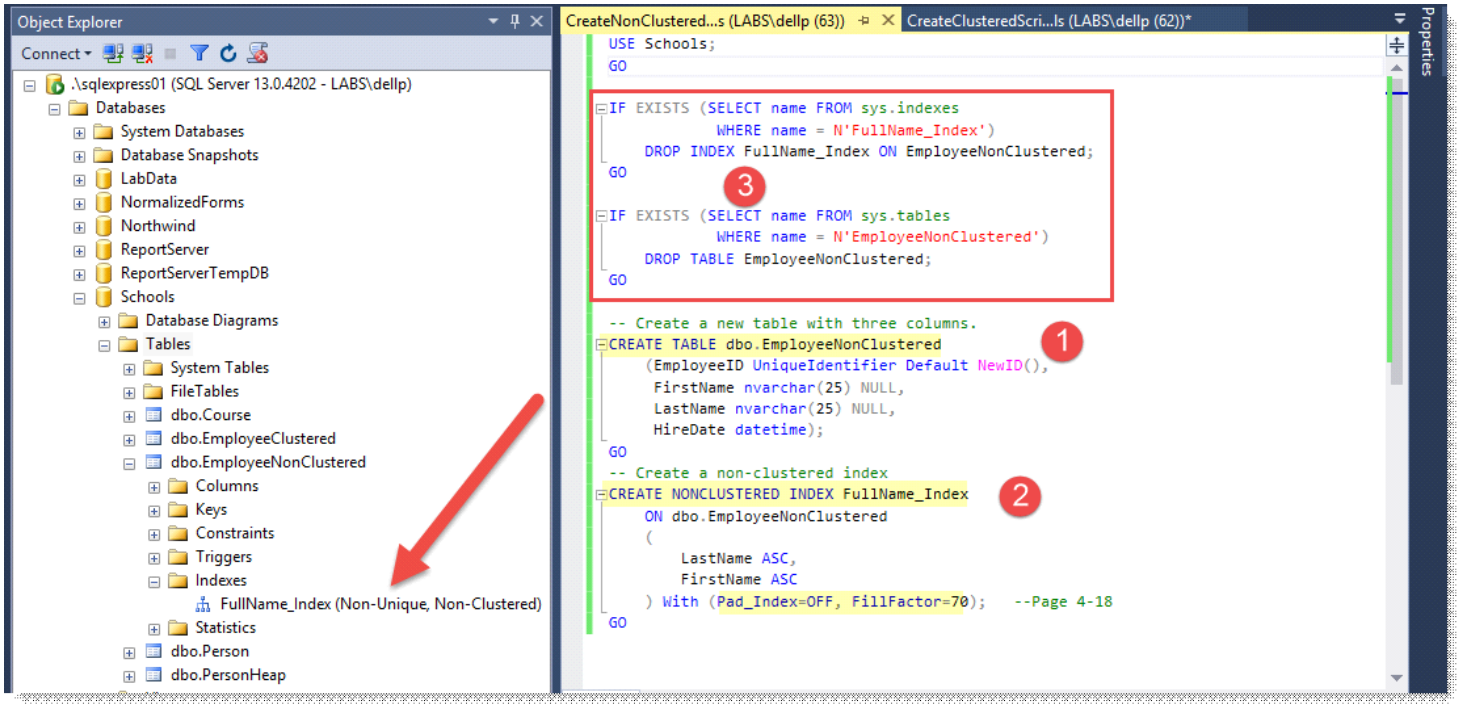
Tuesday, May 23, 2017 8:14 PM

Put If statements in after Table and Index are created…

The FILLFACTOR option for an index determines the percentage of free space that is reserved on each leaf level page of an index when it is created or rebuilt
From <https://dbamohsin.wordpress.com/tag/pad_index/>

PAD_INDEX=On; This option specifies index padding. When turned ON, it uses the percentage specified by FILLFACTOR is applied to the intermediate-level and root level pages of an index.

From <https://dbamohsin.wordpress.com/tag/pad_index/>



Fragmentation

Tuesday, May 23, 2017 12:56 PM

https://ola.hallengren.com/sql-server-index-and-statistics-maintenance.html

- [ ] Create Person - *CreatePersonTable.sql*

CreateIndexFullNam...s (LABS\dellp (67)) | CreatePersonTable.s...ls (LABS\dellp (56))

```sql
USE Schools;
GO

CREATE TABLE Person
(
    PersonID uniqueidentifier DEFAULT NEWID(),
    FirstName VarChar(25) NOT NULL,
    LastName VarChar(25) NOT NULL,
    Address VarChar(50) NULL,
    City VarChar(25) NULL,
    State VarChar(2) NOT NULL,
    ZipCode VarChar(25) NULL,
    PRIMARY KEY (PersonID)
)
GO
```

- [ ] Create FullName_Index

CreateIndexFullName.sql

CreateIndexFullNam...s (LABS\dellp (67))* | CreatePersonTable.s...ls (LABS\dellp (5

```sql
USE [Schools]
GO

CREATE NONCLUSTERED INDEX [FullName_Index] ON [dbo].[Person]
(
    [LastName] ASC,
    [FirstName] ASC
)
GO
```

- [ ] Add Rows
- [ ] Check Fragmentation

File: Insert 1000Rows

CreatePersonTable.s...ls (LABS\dellp (53)) | Insert 1000 Rows.sql...ls (LABS\dellp (61))*

```sql
USE Schools
Go

INSERT INTO Person VALUES
(NewID(), 'Mickey', 'Mouse', '123 Main Street', 'Sacramento', 'CA', '95830')
GO 1000

--Drop Index FullName_Index;
--Drop Table Person;
```

SQLQuery7.sql - (lo...ls (LABS\dellp (54)) | Insert 1000 Rows.sql...ls (LABS\dellp (

```sql
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT TOP (1000) [PersonID]
      ,[FirstName]
      ,[LastName]
      ,[Address]
      ,[City]
      ,[State]
  FROM [Schools].[dbo].[Person]
```

96 %

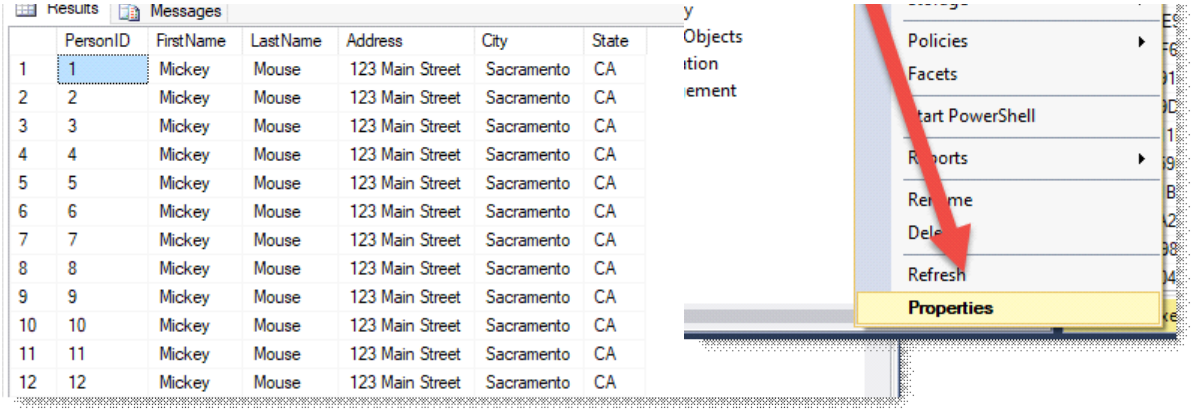
Results | Messages

	PersonID	FirstName	LastName	Address	City	State
1	1	Mickey	Mouse	123 Main Street	Sacramento	CA
2	2	Mickey	Mouse	123 Main Street	Sacramento	CA

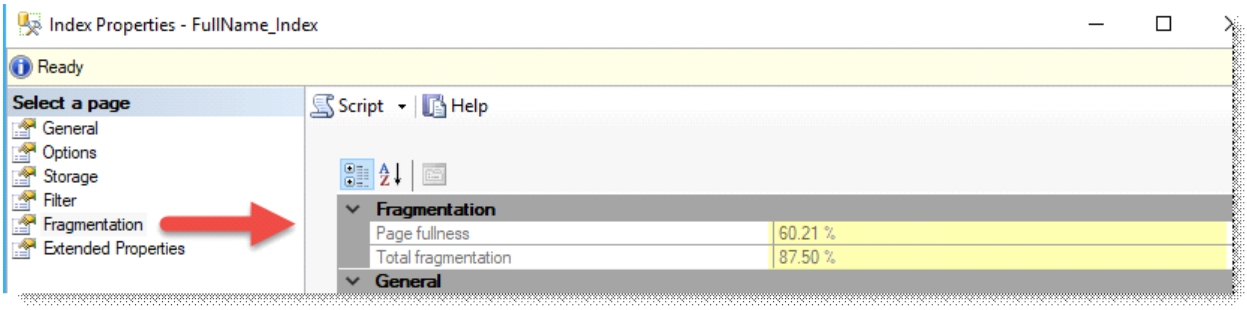
Indexes
PK__Person__AA2FFB85D5E3E2D4 (
FullName_Index (Non-Unique, Non
Statistic
Views
Synonyms
Programmability
Service Broker
Storage
Security
y
Objects
ition
ement

New Index
Script Index as
Rebuild
Reorganize
Disable
Storage
Policies
Facets

Results

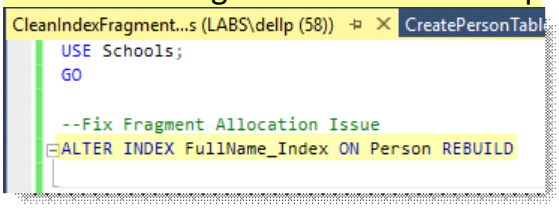


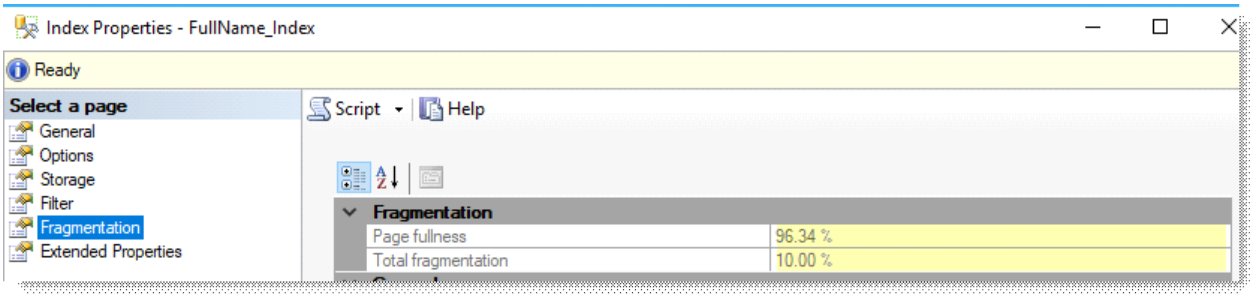
CHECK FRAGMENTATION



CLEAN FRAGMENTATION - REBUILD

CleanIndexFragmentationRebuild.sql





OR - TO CLEAN FRAGMENTATION - DROP/RE-CREATE

ClearIndexFragmentationDropCreate.sql

CleanIndexFragment...s (LABS\dellp (67)) | CleanIndexFragment...s (LABS\dellp (58))

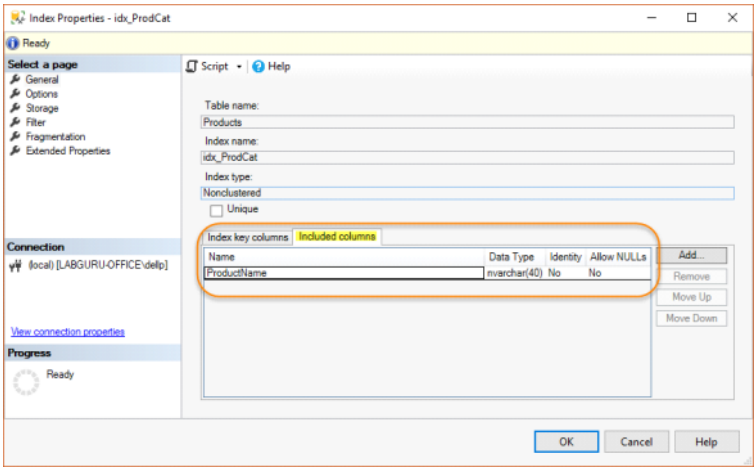
```sql
USE [Schools]
GO

DROP INDEX FullName_Index ON Person;
/****** Object:  Index [FullName_Index]    Script Date: 5/23/2017 7:30:45 PM ******/
CREATE NONCLUSTERED INDEX [FullName_Index] ON [dbo].[Person]
(
    [LastName] ASC,
    [FirstName] ASC
)ON [PRIMARY]
GO
```

Include And Filtered Index

Sunday, December 3, 2017 3:14 PM

Include Index

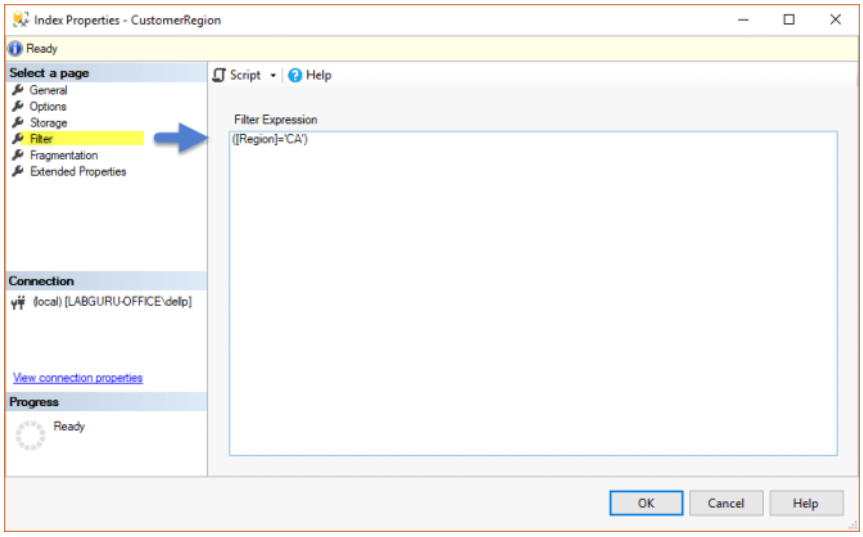
```sql
CREATE NONCLUSTERED INDEX idx_ProdCat
  ON Products(ProductID, CategoryID)
  INCLUDE (ProductName); --Covered w/o sorting

--      index       index     Retieve but not sort
SELECT ProductId, CategoryID, ProductName
FROM Products
WHERE CategoryID = 5;
```



Filtered Index

00b-FilteredIndex.s...ls (LABS\dellp (54)) | 00a-IndexInclude.sq...s (LABS\dellp

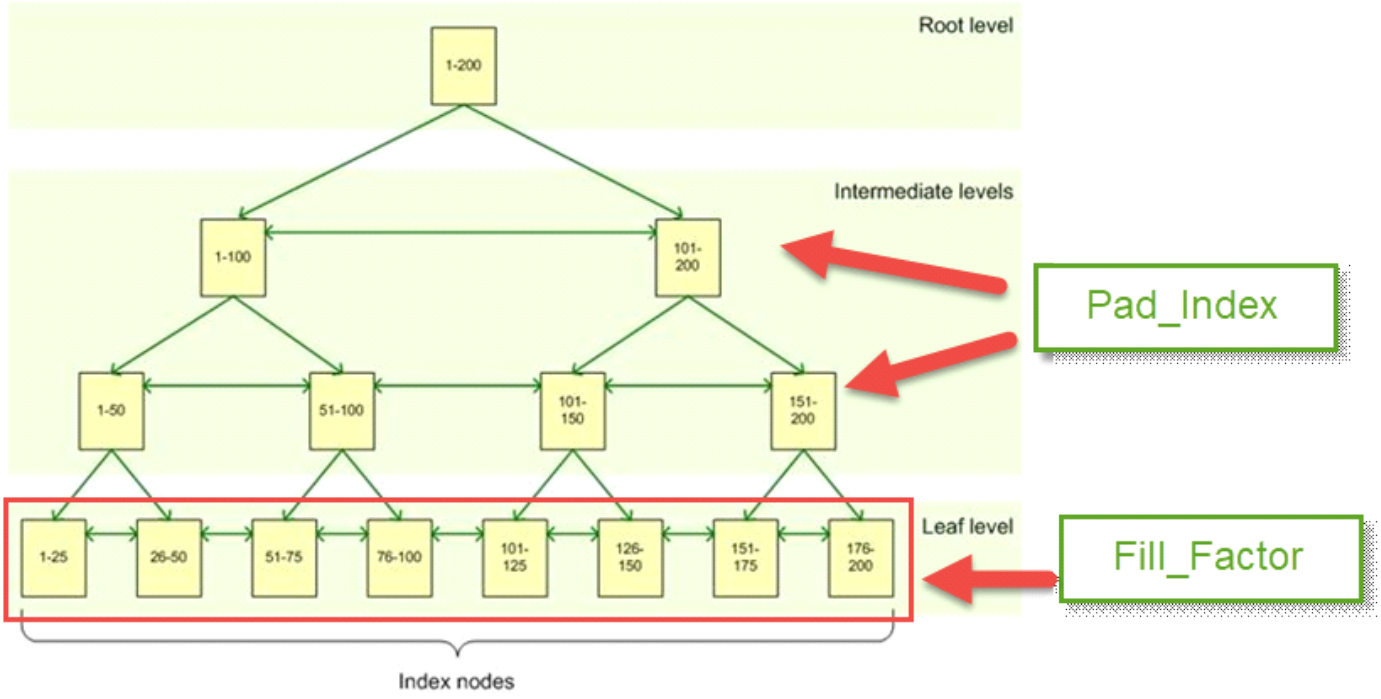
```sql
Use Northwind;
Go

SELECT Address, City, Region, PostalCode
FROM  Customers
WHERE Region = 'CA';

CREATE INDEX CustomerRegion
ON Customers (Region)
Where Region='CA';

--Drop Index CustomerRegion On Customers
```



Index Leaf Structure

Saturday, December 2, 2017 10:08 PM

https://dbamohsin.wordpress.com/tag/pad_index/



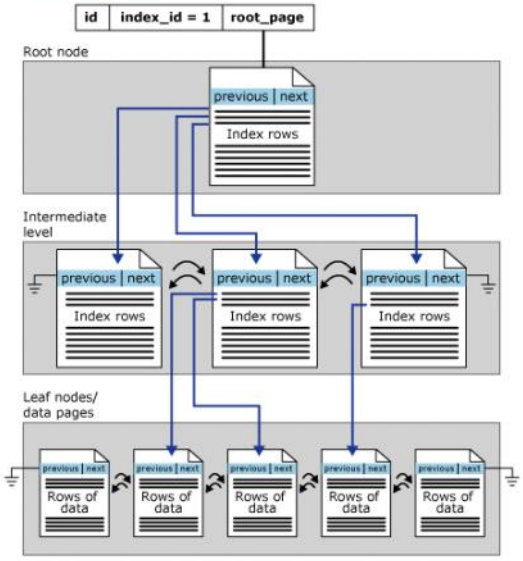
Fill-Factor & Pad_Index

Friday, February 8, 2019 9:56 PM

- FILLFACTOR applies to the bottom layer
  This is the leaf node/data layer in the picture below
- PAD_INDEX ON means "Apply FILLFACTOR to all layers"
  This is the root node and intermediate level in the picture below

This means that PAD_INDEX is only useful if FILLFACTOR is set. FILLFACTOR determines how much free space in an data page (roughly)

A picture from MSDN:



share improve this answer

answered Jul 28 '11 at 10:02

gbn
291k • 45 • 414 • 512

On this page msdn.microsoft.com/en-us/library/ms186869.aspx it says when pad_index is on, "The percentage of free space that is specified by FILLFACTOR is applied to the intermediate-level pages of the index". Will it also apply to the root level? Maybe it's just a books online oversight. – Ogrish Man Dec 4 '16 at 13:32