Understanding and Managing SQL Server Fragmentation

Greg Robidoux
Edgewood Solutions
gregr@edgewoodsolutions.com

Bullett Manale
Idera
www.idera.com

Idera Solutions
for SQL Server

Performance & Availability

- **SQL diagnostic manager™**
  Monitor, diagnose & manage performance
- **SQL mobile manager™**
  Monitor, diagnose & manage performance from a PDA
- **SQL defrag manager™**
  Automate & optimize database defragmentation

Compliance & Security

- **SQL compliance manager™**
  Capture, audit & alert on user activity
- **SQLsecure™**
  Assess security risks & audit access rights

Change Management

- **SQL change manager™**
  Manage & monitor schema changes

Backup & Recovery

- **SQLsafe™**
  Accelerate, compress & encrypt database backups

Administration

- **SQL admin toolset™**
  24 essential tools for monitoring, administering, troubleshooting and reporting

idera™
MANAGING YOUR WINDOWS℠ WORLD™

Agenda - Fragmentation

- Overview
  - Understanding Storage
  - What fragmentation is and how it occurs
  - Detecting
  - Fixing
  - Managing
- Other Tools
- Questions / Wrap Up

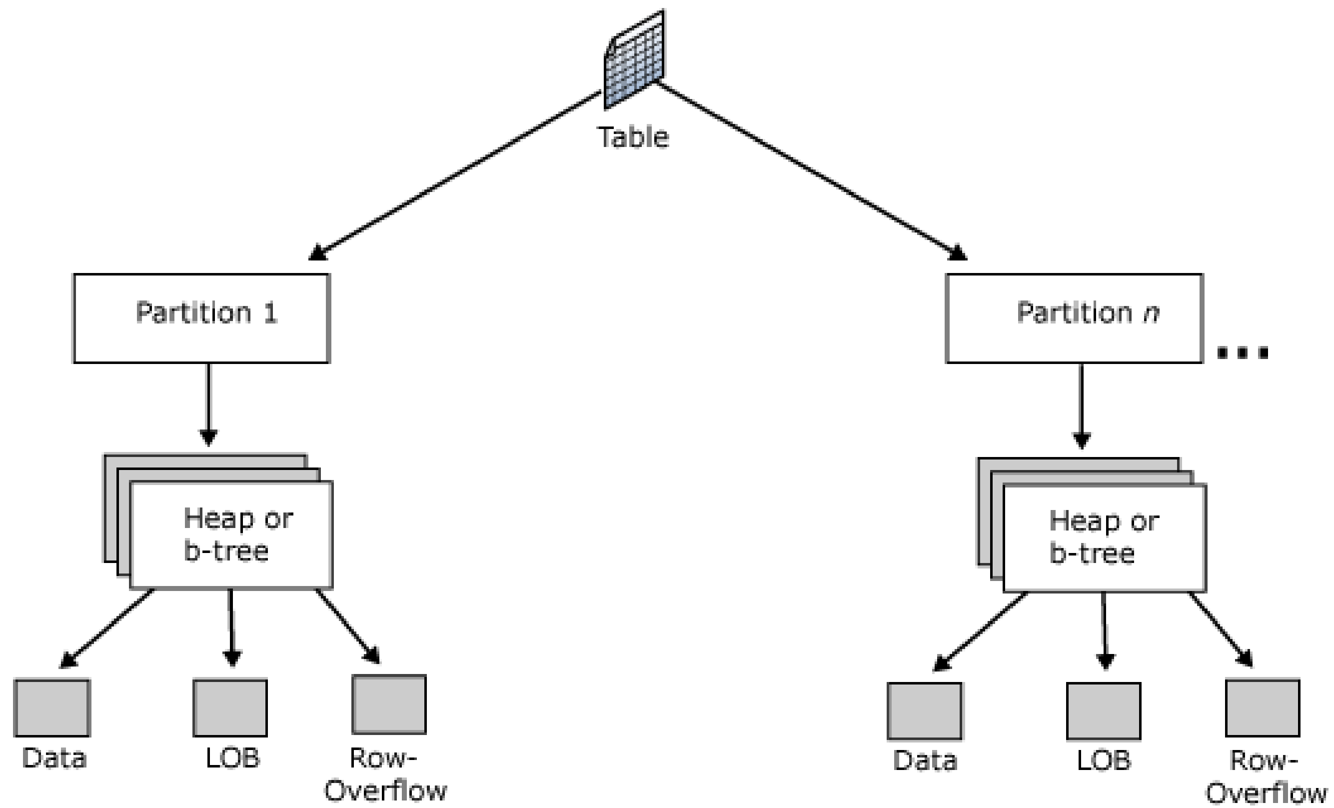
MSSQLTips.com

Storage

- Data storage
  - Pages - tables and indexes are stored as a collection of 8-KB pages
  - Extents – 8 contiguous pages = 64KB total space
- Cluster or Heap
  - Clustered – data is stored in order based on the clustered index (b-tree)
  - Heap – there is no particular order to how data is stored
- Partitions
  - Tables can have one or more partitions. The default is one per table.

MSSQLTips.com

Storage



Source: http://msdn.microsoft.com/en-us/library/ms130214.aspx

MSSQLTips.com

Storage

Allocation type	Used to manage
IN_ROW_DATA (Data)	• Data or index rows that contain all data, except large object (LOB) data. • Pages are of type Data or Index.
LOB_DATA (LOB)	• Large object data stored in one or more of these data types: text, ntext, image, xml, varchar(max), nvarchar(max), varbinary(max), or CLR user-defined types (CLR UDT). • Pages are of type Text/Image
ROW_OVERFLOW_DATA (Row-Overflow)	• Variable length data stored in varchar, nvarchar, varbinary, or sql_variant columns that exceed the 8,060 byte row size limit. • Pages are of type Data

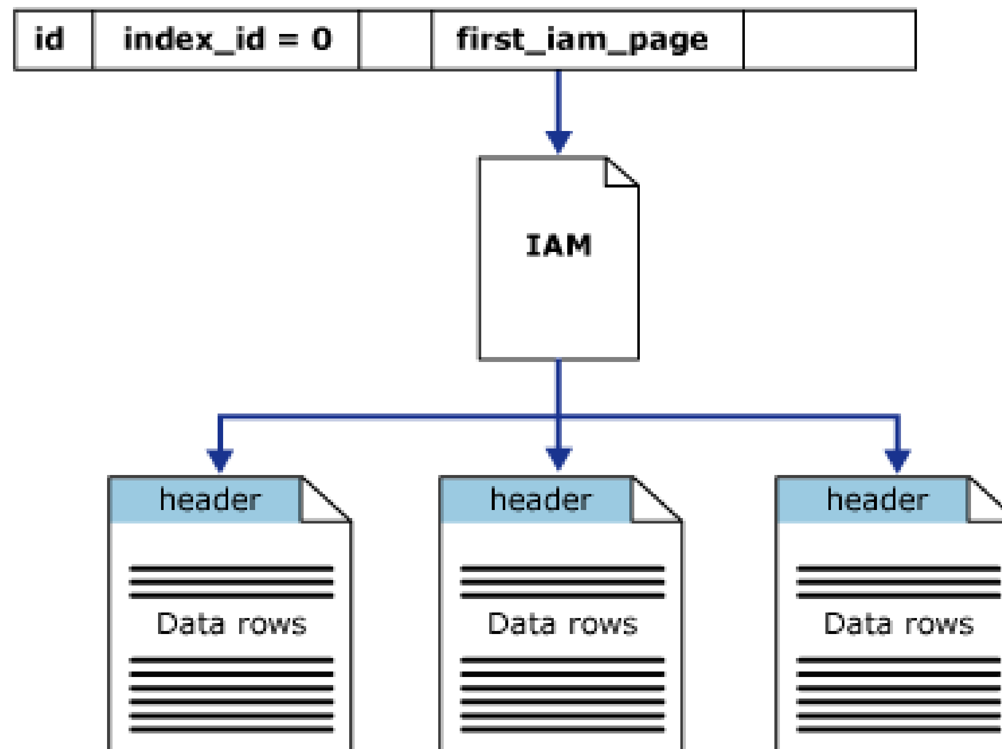
MSSQLTips.com

Storage - HEAP

- A heap is a table without a clustered index
- Uses Index Allocation Map (IAM) pages to find data

MSSQLTips.com

Storage - HEAP



Source: http://msdn.microsoft.com/en-us/library/ms130214.aspx

MSSQLTips.com

Storage - Indexes

- Indexes
  - Clustered
  - Non-clustered
  - XML
  - Full Text
- Clustered and Non-clustered
  - Use b-tree index structure

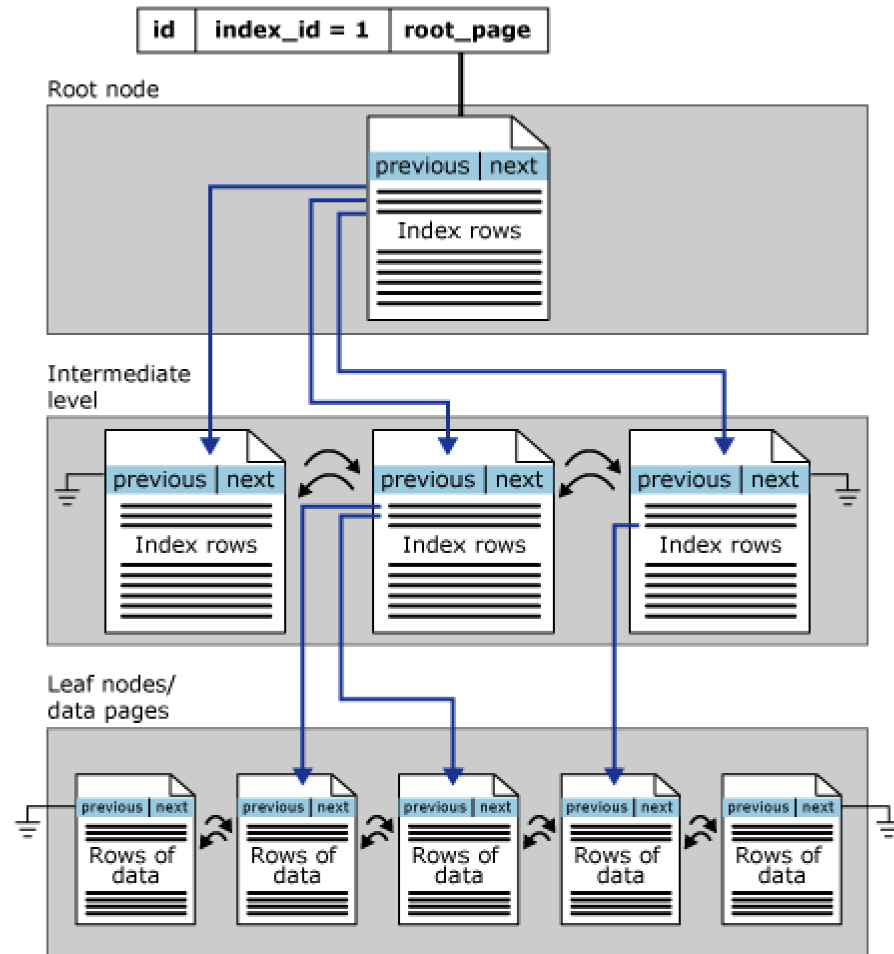
MSSQLTips.com

Storage – B-TREE

- B-TREE Structure
  - In SQL Server, indexes are organized as B-trees.
  - Each page in an index B-tree is called an index node.
  - The top node of the B-tree is called the root node.
  - The bottom level of nodes in the index is called the leaf nodes.
  - Any index levels between the root and the leaf nodes are collectively known as intermediate levels.
  - In a clustered index, the leaf nodes contain the data pages of the underlying table.
  - The root and leaf nodes contain index pages holding index rows.
  - Each index row contains a key value and a pointer to either an intermediate level page in the B-tree, or a data row in the leaf level of the index.
  - The pages in each level of the index are linked in a doubly-linked list. This means that there are pointers to the previous and next pages.

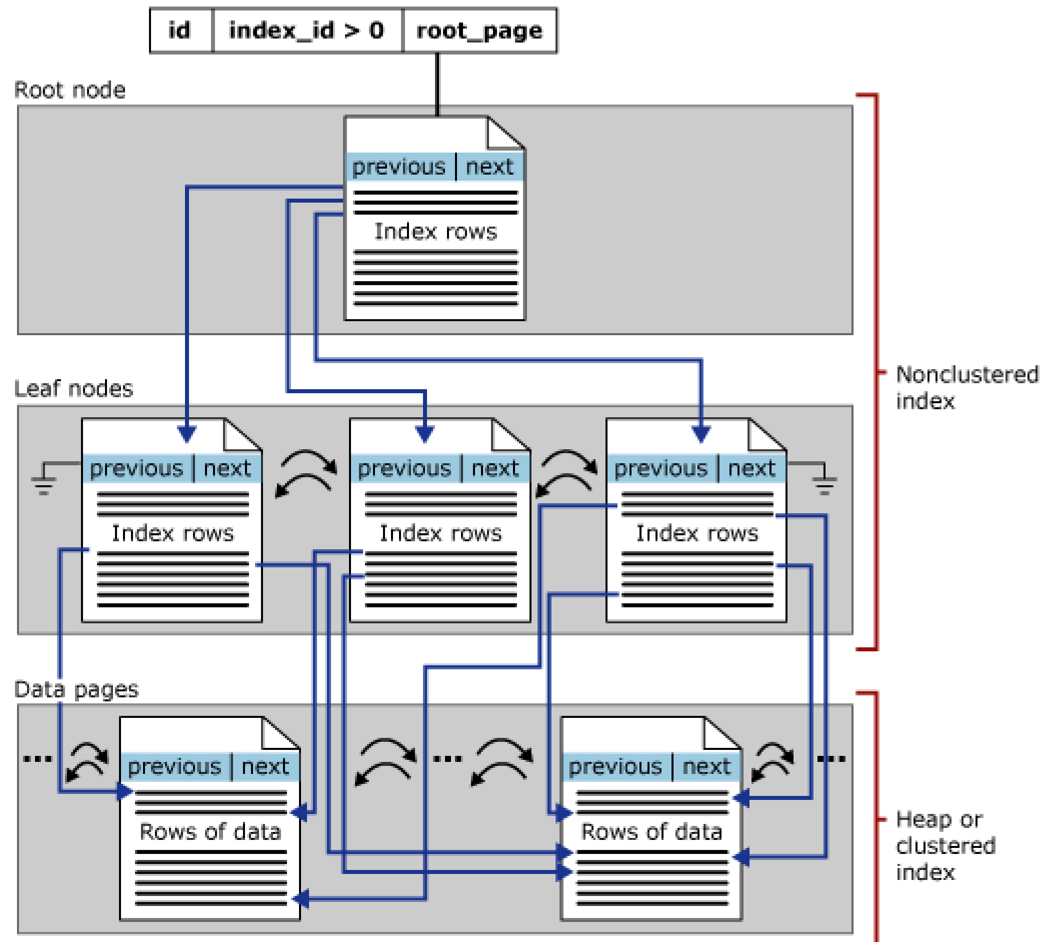
MSSQLTips.com

Storage - Clustered Index



Source: http://msdn.microsoft.com/en-us/library/ms130214.aspx

Storage - Non-Clustered Index



Source: http://msdn.microsoft.com/en-us/library/ms130214.aspx

MSSQLTips.com

Storage - Allocation

- sys.allocation_units - contains a row for each allocation unit in the database.

```
SELECT o.name AS table_name,p.index_id, i.name AS index_name ,
   au.type_desc AS allocation_type, au.data_pages, partition_number
FROM sys.allocation_units AS au
   JOIN sys.partitions AS p ON au.container_id = p.partition_id
   JOIN sys.objects AS o ON p.object_id = o.object_id
   JOIN sys.indexes AS i ON p.index_id = i.index_id
         AND i.object_id = p.object_id
```


MSSQLTips.com

Storage - Allocation

	table_name	index_id	index_name	allocation_type	data_pages	partition_number
156	SalesOrderDetail	1	PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID	IN_ROW_DATA	1233	1
157	SalesOrderDetail	2	AK_SalesOrderDetail_rowguid	IN_ROW_DATA	406	1
158	SalesOrderDetail	3	IX_SalesOrderDetail_ProductID	IN_ROW_DATA	226	1
159	CurrencyRate	1	PK_CurrencyRate_CurrencyRateID	IN_ROW_DATA	96	1
160	CurrencyRate	2	AK_CurrencyRate_CurrencyRateDate_FromCurrencyCode_ToCurrencyCode	IN_ROW_DATA	46	1
161	Customer	1	PK_Customer_CustomerID	IN_ROW_DATA	103	1
162	Customer	2	AK_Customer_rowguid	IN_ROW_DATA	55	1
163	Customer	3	AK_Customer_AccountNumber	IN_ROW_DATA	50	1
164	Customer	5	IX_Customer_TerritoryID	IN_ROW_DATA	34	1
165	SalesOrderHeader	1	PK_SalesOrderHeader_SalesOrderID	IN_ROW_DATA	699	1
166	SalesOrderHeader	2	AK_SalesOrderHeader_rowguid	IN_ROW_DATA	90	1
167	SalesOrderHeader	3	AK_SalesOrderHeader_SalesOrderNumber	IN_ROW_DATA	98	1
168	SalesOrderHeader	5	IX_SalesOrderHeader_CustomerID	IN_ROW_DATA	43	1
169	SalesOrderHeader	6	IX_SalesOrderHeader_SalesPersonID	IN_ROW_DATA	55	1
170	CustomerAddress	1	PK_CustomerAddress_CustomerID_AddressID	IN_ROW_DATA	108	1
171	CustomerAddress	2	AK_CustomerAddress_rowguid	IN_ROW_DATA	65	1

	table_name	index_id	index_name	allocation_type	data_pages	partition_number
174	Document	1	PK_Document_DocumentID	LOB_DATA	0	1
175	Document	1	PK_Document_DocumentID	IN_ROW_DATA	1	1
176	Document	1	PK_Document_DocumentID	ROW_OVERFLOW_DATA	0	1
177	Document	2	AK_Document_FileName_Revision	IN_ROW_DATA	1	1

MSSQLTips.com

What is fragmentation

- What it is
  - **Fragmentation** is when storage space is used inefficiently, reducing storage capacity and in most cases performance
  - **Internal fragmentation** occurs when storage is allocated without using it
  - **External fragmentation** is when storage becomes divided into many small pieces over time

MSSQLTips.com

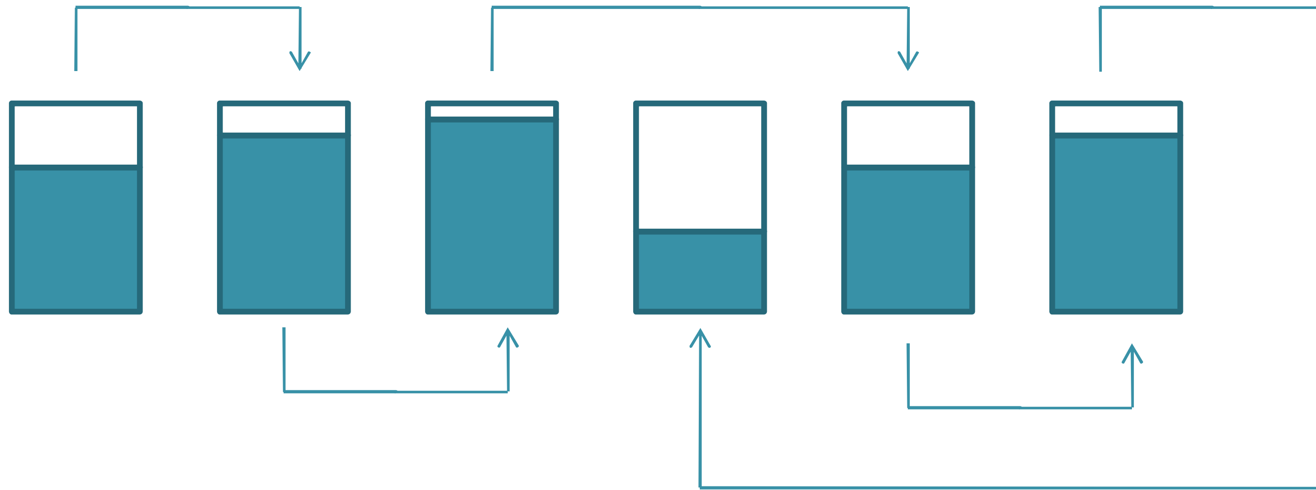
What is fragmentation

- **SQL Server Fragmentation**
  - **Logical Fragmentation**
    - This is the percentage of out-of-order pages in the leaf pages of an index.
    - An out-of-order page is one for which the next page indicated in an IAM is a page that is different from the page pointed to by the next page pointer in the leaf page.
  - **Extent Fragmentation**
    - This is the percentage of out-of-order extents in the leaf pages of a heap.
    - An out-of-order extent is one for which the extent that contains the current page for a heap is not physically the next extent after the extent that contains the previous page.

MSSQLTips.com

Logical Fragmentation

Page level fragmentation



MSSQLTips.com

Extent Fragmentation

Fragmentation – before

Fragmentation – after

When it happens

- How it happens
  - Data modification
    - Inserting – causes page splits
    - Deleting – leaves free space
    - Updating – changing index values
  - Shrinking
    - DBCC SHRINKDATABASE
    - DBCC SHRINKFILE (data files)
    - AutoShrink

MSSQLTips.com

Why is this bad

- Uses additional space
- Increased I/O for range scans
- Impacts performance
- Places additional demands on hardware

MSSQLTips.com

Detecting

- DBCC SHOWCONTIG
- sys.dm_db_index_physical_stats
  - **LIMITED** - fastest and scans the smallest number of pages. It scans all pages for a heap, but only the parent-level pages for an index, which are the pages above the leaf-level.
  - **SAMPLED** - returns statistics based on a one percent sample of all the pages in the index or heap. If the index or heap has fewer than 10,000 pages DETAILED mode is used.
  - **DETAILED** - scans all pages and returns all statistics. This also takes the longest

MSSQLTips.com

Detecting

```
SELECT *
FROM sys.dm_db_index_physical_stats (
  { database_id | NULL } ,
  { object_id | NULL } ,
  { index_id | NULL | 0 } ,
  { partition_number | NULL } ,
  { mode | NULL | DEFAULT } )
```


MSSQLTips.com

Detecting – all DBs all indexes

```
SELECT *
FROM sys.dm_db_index_physical_stats (
  NULL,
  NULL,
  NULL,
  NULL ,
  'LIMITED');
```

Detecting – one table all indexes

```
DECLARE @db_id SMALLINT;
DECLARE @object_id INT;
SET @db_id = DB_ID(N'Test');
SET @object_id = OBJECT_ID(N'Test.dbo.Address');

SELECT *
FROM sys.dm_db_index_physical_stats (
   @db_id,
   @object_id,
   NULL,
   NULL ,
   'LIMITED');
```


MSSQLTips.com

DMF - Output

1	DBName	TName	index_id	partition_number	index_type_desc	alloc_unit_type_desc	index_depth	index_level	avg_fragmentation_in_percent
2	AdventureWorks	Employee	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	0	28.57142857
3	AdventureWorks	Employee	1	1	CLUSTERED INDEX	IN_ROW_DATA	2	1	0
4	AdventureWorks	Employee	2	1	NONCLUSTERED INDEX	IN_ROW_DATA	2	0	66.66666667
5	AdventureWorks	Employee	2	1	NONCLUSTERED INDEX	IN_ROW_DATA	2	1	0
6	AdventureWorks	Employee	3	1	NONCLUSTERED INDEX	IN_ROW_DATA	2	0	50
7	AdventureWorks	Employee	3	1	NONCLUSTERED INDEX	IN_ROW_DATA	2	1	0
8	AdventureWorks	Employee	4	1	NONCLUSTERED INDEX	IN_ROW_DATA	1	0	0
9	AdventureWorks	Employee	5	1	NONCLUSTERED INDEX	IN_ROW_DATA	1	0	0

1	fragment_count	avg_fragment_size_in_pages	page_count	avg_page_space_used_in_percent	record_count	ghost_record_count	version_ghost_record_count
2	4	1.75	7	97.8679145	290	0	0
3	1	1	1	1.099579936	7	0	0
4	3	1	3	66.58430936	290	0	0
5	1	1	1	2.112676056	3	0	0
6	2	1	2	51.32196689	290	0	0
7	1	1	1	0.716580183	2	0	0
8	1	1	1	82.38201137	290	0	0
9	1	1	1	50.13590314	290	0	0

1	min_record_size_in_bytes	max_record_size_in_bytes	avg_record_size_in_bytes	forwarded_record_count
2	143	219	189.255	NULL
3	11	11	11	NULL
4	47	65	53.772	NULL
5	53	59	55.666	NULL
6	19	27	26.662	NULL
7	27	29	28	NULL
8	21	21	21	NULL
9	12	12	12	NULL

MSSQLTips.com

SHOWCONTIG - Output

```
DBCC SHOWCONTIG scanning 'Employee' table...
Table: 'Employee' (869578136); index ID: 1, database ID: 10
TABLE level scan performed.
- Pages Scanned................................: 7
- Extents Scanned..............................: 3
- Extent Switches..............................: 3
- Avg. Pages per Extent........................: 2.3
- Scan Density [Best Count:Actual Count].......: 25.00% [1:4]
- Logical Scan Fragmentation ..................: 28.57%
- Extent Scan Fragmentation ...................: 33.33%
- Avg. Bytes Free per Page.....................: 172.6
- Avg. Page Density (full).....................: 97.87%
DBCC execution completed. If DBCC printed error messages, contact your system administrator.
```


MSSQLTips.com

sys.dm_db_index_physical_stats

Column name	Description
database_id	Database ID of the table or view.
object_id	Object ID of the table or view that the index is on.
index_id	Index ID of an index. 0 = HEAP
partition_number	Partition number.
index_type_desc	Description of the index type: HEAP, Clustered, Non-clustered, etc…
alloc_unit_type_desc	Description of the allocation unit type: In-Row, LOB, Overflow
index_depth	Number of index levels.
index_level	Current level of the index.
avg_fragmentation_in_percent	Logical fragmentation for indexes, or extent fragmentation for heaps in the IN_ROW_DATA allocation unit.
fragment_count	Number of fragments in the leaf level of an IN_ROW_DATA allocation unit.
avg_fragment_size_in_pages	Average number of pages in one fragment in the leaf level of an IN_ROW_DATA allocation unit.
page_count	Total number of index or data pages.
avg_page_space_used_in_percent	Average percentage of available data storage space used in all pages.
record_count	Total number of records.
ghost_record_count	Number of deleted records (ghost records) ready for removal by the ghost cleanup task in the allocation unit.
version_ghost_record_count	Number of ghost records retained by an outstanding snapshot isolation transaction in an allocation unit.
min_record_size_in_bytes	Minimum record size in bytes.
max_record_size_in_bytes	Maximum record size in bytes.
avg_record_size_in_bytes	Average record size in bytes.
forwarded_record_count	Number of forwarded records in a heap.

MSSQLTips.com

Fixing

- Clustered and Non-clustered indexes
  - ALTER INDEX … REBUILD
    - Rebuild a new index, built side by side
    - Can run as a parallel operation
    - Offline operation except for Enterprise Edition
  - ALTER INDEX … REORGANIZE
    - First moves page data to the left side of the index to get the pages as full as possible and then removes any unneeded pages, then reorders the pages
    - Does not correct extent fragmentation
    - Can not run as a parallel operation
    - Online operation
- HEAP
  - Create a clustered index on the table and then drop the clustered index
  - Or create new table and move data to the new table

MSSQLTips.com

Rebuild vs Reorganize

Functionality	Rebuild	Reorganize
Online/Offline	Offline / (Online Enterprise)	Online
Faster when logical fragmentation is:	High	Low
Parallel processing	Yes	No
Compacts pages	Yes	Yes
Can be stopped and restarted without losing work completed to that point	No	Yes
Able to untangle interleaved indexes	May reduce interleaving	No
Additional free space is required in the data file for defragmenting	Yes	No
Faster on larger indexes	Yes	No
Rebuilds statistics	Yes	No
Log space usage	High in full recovery mode (logs entire contents of the index), low in bulk logged or simple recovery mode (only logs allocation of space)	Varies based on the amount of work performed
May skip pages on busy systems	No	Yes

Source: http://technet.microsoft.com/en-us/library/cc966523.aspx

MSSQLTips.com

Rebuild or Reorganize

Guideline when to rebuild versus reorganize based on the avg_fragmentation_in_percent value

avg_fragmentation_in_percent value	Corrective statement
< = 30%	ALTER INDEX REORGANIZE
> 30%	ALTER INDEX REBUILD

MSSQLTips.com

ALTER INDEX … REBUILD

```
ALTER INDEX { index_name | ALL }
   ON <object>
   REBUILD
WITH
   PAD_INDEX = { ON | OFF }
   FILLFACTOR = fillfactor  {1-100}
   SORT_IN_TEMPDB = { ON | OFF }
   IGNORE_DUP_KEY = { ON | OFF }
   STATISTICS_NORECOMPUTE = { ON | OFF }
   ONLINE = { ON | OFF }
   ALLOW_ROW_LOCKS = { ON | OFF }
   ALLOW_PAGE_LOCKS = { ON | OFF }
   MAXDOP = max_degree_of_parallelism  {  0 = all or specify value }
```

Note: The underlined value is the default

ALTER INDEX … REBUILD

- FILLFACTOR – how full to make the leaf level pages of the index. This is a percentage from 1-100, the default is 0 which is the same as 100.
- PAD_INDEX – specifies whether you want to leave free space in the intermediate pages. The fillfactor value is used for this, either the saved value or the value you specify in the command.
- SORT_IN_TEMPDB – this specifies whether to use the TempDB database to do a sort or to use the user database. If there is enough memory to sort the index this will all be done in memory instead.
- IGNORE_DUP_KEY – this tells SQL whether to continue or fail the index build if there is a duplicate key.
- STATISTICS_NORECOMPUTE – this tells SQL whether to re-compute the statistics for the index
- ONLINE – this allows the index to be built online, so the there is no locking of the table or index. This is only available for the Enterprise Edition.
- ALLOW_ROW_LOCKS – tells SQL whether to use row locking when building the index
- ALLOW_PAGE_LOCKS - tells SQL whether to use page level locking when building the index
- MAXDOP – specifies how many processors to use for the index build. Only available in the Enterprise Edition.

Source: http://msdn.microsoft.com/en-us/library/ms130214.aspx

MSSQLTips.com

ALTER INDEX … REBUILD

```
ALTER INDEX PK_Employee_EmployeeID
  ON HumanResources.Employee
  REBUILD;
```


MSSQLTips.com

ALTER INDEX … REBUILD

```
ALTER INDEX ALL
   ON Production.Product
   REBUILD
WITH
   (FILLFACTOR = 80,
   SORT_IN_TEMPDB = ON,
   STATISTICS_NORECOMPUTE = ON);
```


MSSQLTips.com

ALTER INDEX … REORGANIZE

```
ALTER INDEX { index_name | ALL }
  ON <object>
  REORGANIZE
WITH
  LOB_COMPACTION  = { ON | OFF }
```

Note: The underlined value is the default

ALTER INDEX … REORGANIZE

- LOB_COMPACTION - Specifies that all pages that contain large object (LOB) data are compacted. LOB data types are image, text, ntext, varchar(max), nvarchar(max), varbinary(max), and xml.

MSSQLTips.com

ALTER INDEX … REORGANIZE

```
ALTER INDEX PK_ProductPhoto_ProductPhotoID
   ON Production.ProductPhoto
   REORGANIZE ;
```


MSSQLTips.com

Partitioned Indexes - Rebuild

```
ALTER INDEX { index_name | ALL }
  ON <object>
  REBUILD
  PARTITION = partition_number
WITH
  SORT_IN_TEMPDB = { ON | OFF }
  MAXDOP = max_degree_of_parallelism
```


MSSQLTips.com

Partitioned Indexes - Reorganize

ALTER INDEX { *index_name* | ALL }
  ON <object>
  REORGANIZE
  PARTITION = *partition_number*
WITH
  LOB_COMPACTION = { ON | OFF }

Partitioned Indexes

```
ALTER INDEX IX_TransactionHistory
  ON Production.TransactionHistory
  REBUILD
  Partition = 5;

ALTER INDEX IX_TransactionHistory
  ON Production.TransactionHistory
  REORGANIZE
  Partition = 5;
```


MSSQLTips.com

DROP EXISTING

- You can use the DROP_EXISTING clause to rebuild the index, add or drop columns, modify options, modify column sort order, or change the partition scheme or filegroup.
- If the index enforces a PRIMARY KEY or UNIQUE constraint and the index definition is not altered in any way, the index is dropped and re-created preserving the existing constraint.
- DROP_EXISTING enhances performance when you re-create a clustered index, with either the same or different set of keys, on a table that also has nonclustered indexes.
- The nonclustered indexes are rebuilt once, and then only if the index definition has changed.
- The DROP_EXISTING clause does not rebuild the nonclustered indexes when the index definition has the same index name, key and partition columns, uniqueness attribute, and sort order as the original index.

MSSQLTips.com

DROP EXISTING

```
CREATE CLUSTERED INDEX
  IX_WorkOrder_ProductID
  ON Production.WorkOrder(ProductID)
WITH
  (DROP_EXISTING = ON);
```


MSSQLTips.com

HEAP

```
CREATE CLUSTERED INDEX IX_WorkOrder
   ON Production.WorkOrder(ProductID)

DROP Production.WorkOrder. IX_WorkOrder
```

MSSQLTips.com

Managing

- Collecting data
- Selective rebuilds / reorgs
- Removing unused indexes
- Recovery Models and impact
- Transaction Log Usage
  - DBCC SQLPERF(logspace)
- Online rebuilds
- MAXDOP
- Changing index settings
  - PAD INDEX
  - FILLFACTOR
- Maintenance Plans

MSSQLTips.com

Impact by Recovery Model

	Database Recovery Model		
Index operation	Full	Bulk-logged	Simple
ALTER INDEX REORGANIZE	Fully logged	Fully logged	Fully logged
ALTER INDEX REBUILD	Fully logged	Minimally logged	Minimally logged
CREATE INDEX	Fully logged	Minimally logged	Minimally logged
DROP INDEX	Index page deallocation is fully logged; new heap rebuild, if applicable, is fully logged.	Index page deallocation is fully logged; new heap rebuild, if applicable, is minimally logged.	Index page deallocation is fully logged; new heap rebuild, if applicable, is minimally logged.

Source: http://msdn.microsoft.com/en-us/library/ms130214.aspx

MSSQLTips.com

Monitoring Using Other Tools

MSSQLTips.com

Questions and Wrap-up

- Thanks to our sponsor: Idera
- Next webcast in the series:
  - Database Mirroring Concepts
  - May 13th, 2009, 4pm EDT

idera™
MANAGING YOUR WINDOWS® WORLD™