PASS SQL saturday

#183 | ALBUQUERQUE 2013

Table Indexing for the .NET Developer

Denny Cherry
mrdenny@dcac.co
twitter.com/mrdenny

About Me

- Denny Cherry & Associates Consulting
- People Talking Tech
- Author or Coauthor of 5 books
- 8+ SQL Mag articles
- Dozens of other articles
- Microsoft MVP
- Microsoft Certified Master
- VMware vExpert
- Microsoft Certified Trainer

Today's Goals

- Introduce the different kinds of indexes
- Common Misconceptions about indexes
- Downsides to indexes
- Introduce advanced index tuning techniques
- Q & A

Today's Goals

- **Introduce the different kinds of indexes**
- Common Misconceptions about indexes
- Downsides to indexes
- Introduce advanced index tuning techniques
- Q & A

Different Kinds of Indexes

- Five Kinds of Indexes
  - Clustered
  - Non-clustered
  - Full Text
  - XML
  - ColumnStore Indexes
- There's new stuff in SQL Server 2012
  - Semantic Search

Clustered Indexes

- 1 Clustered Index per table
- Contain Full Copy of row data within in the index
- Up to 16 indexed columns can be part of the index
  - (15 if the table contains any XML indexes)
- Primary Key will by default be the Clustered Index
- Must be created on the same filegroup as the table
- Clustered Indexes should be as narrow as possible
- While not required, they are highly recommended

Non-clustered Index

- Up to 999 per table Starting with SQL Server 2008
  - 255 in SQL Server 2005 and below
- Up to 16 indexed columns in the index
- Non-indexed columns can be included via INCLUDE statement
- Non-Clustered indexes always contain the clustered index columns (when table has a clustered index)
- When table is a heap, the Row ID is stored in every non-clustered index.
- Can be created on any filegroup within the database
- Can be filtered indexes to include fewer rows in the index.

Differences between unique and non-unique clustered indexes

- Non-Unique clustered indexes have an extra column called the uniqueifier which ensures that values within the index are unique.
- Uniqueifier is only used for rows which are not unique.

EmpId	Uniqufier
1	
2	
3	
4	0
4	1
5	
6	
7	0
7	1
8	

Full Text Indexes

- Not accessed via normal SELECT statements
- Require use of a predicate:
  - CONTAINS
  - CONTAINSTABLE
  - FREETEXT
  - FREETEXTTABLE
- Can be used to search binary values (doc, docx, xls, pdf) stored within the database.
- Natural Language Search
- Can index XML documents, but only indexes the values, not the tags.

Full Text Indexes (SQL 2005 and below)

- Created and managed outside of the database via Microsoft Search Service
- Backed up with the database (starting in SQL 2005)
- Searches entire index and returns all matches, which you then filter against your normal table to return correct set of rows.

Full Text Indexes (SQL 2008 and up)

- Now stored within the database
- Command is still parsed via MS Search service, but looking is done natively
- Full text search now only searches the required subset of rows
- When creating your indexes use an identity field as the key to improve query performance.

XML Indexes

- Allows you to index specific nodes of the XML document
- 249 XML Indexes pre table
- Requires a Clustered Index on the table
- Each xml column can have a single primary XML index and multiple secondary XML indexes
- XML Indexes can only be created on a single XML Column
- No online rebuilds
- Not available for XML variables. Only used on tables.

Primary XML Index

- When created creates a hidden node table
  - Contains base table primary key and 12 columns of info about every node within the XML value
- Effectively the clustered index on the node table
  - Base Table Clustered Index Value
  - Node id from the node table
- Increases storage 200-500%

Secondary XML Indexes

- Non-Clustered Indexes on the hidden node table
- Three kinds of secondary indexes
  - PATH index on the node id (path) and the value
  - VALUE index is on the value and the node id (path)
  - PROPERTY index is on the base table's clustered index, node id (path) and the value

Today's Goals

- Introduce the different kinds of indexes
- **Common Misconceptions about indexes**
- Downsides to indexes
- Introduce advanced index tuning techniques
- Q & A

Common Misconceptions about indexes

- Indexes don't require maintenance
- If I create one index for each column in my where clause I'll be fine
- The table is sorted based on the order of the Clustered Index
- Clustered Indexes are required

Today's Goals

- Introduce the different kinds of indexes
- Common Misconceptions about indexes
- **Downsides to indexes**
- Introduce advanced index tuning techniques
- Q & A

Downsides to indexes

- Indexes take up space
  - On large complex databases the indexes can take up more space than the table
  - Data is duplicated in each index which contains the column
- Indexes slow down insert, update, delete (especially full text indexes) statements
- Using the wrong index can be slower than using no index
- Encrypted data can't be effectively indexed

Today's Goals

- Introduce the different kinds of indexes
- Common Misconceptions about indexes
- Downsides to indexes
- **Introduce advanced index tuning techniques**
- Q & A

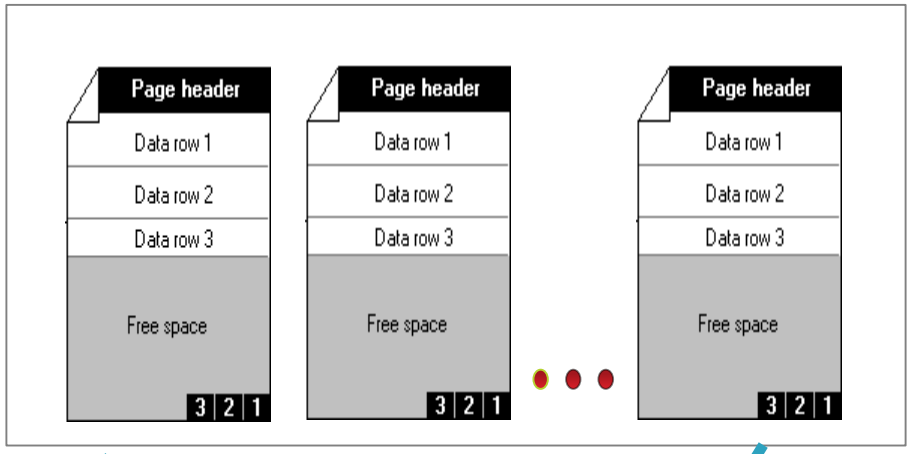
Advanced Index Tuning Techniques

- Fillfactor
  - Tells the SQL Server how much free space to leave in the leaf level pages.
- Padding
  - Tells the SQL Server to use the Fillfactor setting to leave free space in the intermediate-level pages.
- Online Rebuilds
- Data Compression

What is a ColumnStore Index?

- Totally new and different approach to indexing
- Data is stored via columns not rows
- Each column is stored separately, then compressed using VertiPak compression engine
- SQL Server's first B-Tree less index

How does ColumnStore do that?



Existing Pages

Needed Columns

ColumnStore: Use Case

- Data continues to grow, but performance requirements stay the same
- Many data warehouses approach PB ranges
- Data needs to be filtered, aggregated, and grouped despite the size of the dataset

Limitations

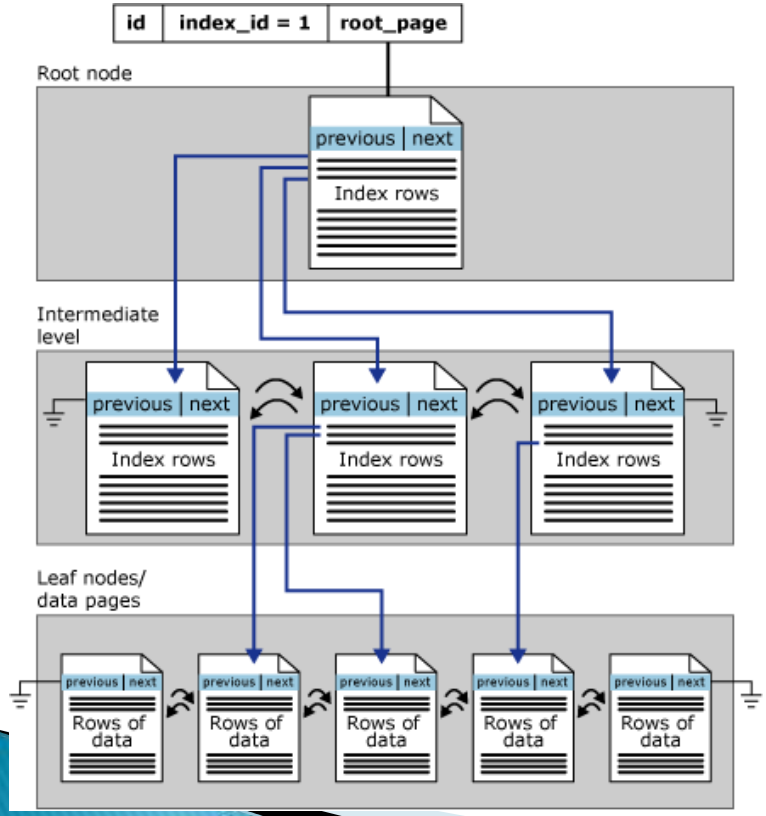
- Unsupported Data Types include
  - Uniqueidentifier
  - Blob
  - Numeric (19,2) or higher
- Read Only
- OUTER JOINs using ColumnStore don't perform well

Using the Advanced Index Tuning Techniques

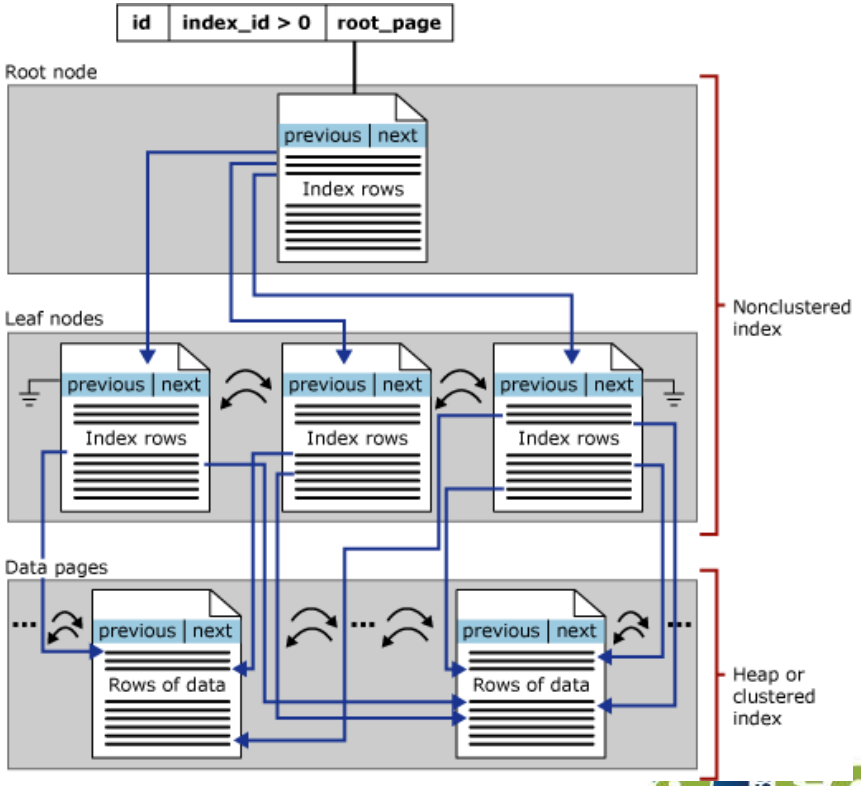
```
CREATE INDEX MyIndex ON dbo.MyTable
ON (Col1, Col5, Col3)
INCLUDE (Col4, Col2)
WHERE Col6 = 'Value3'
WITH (FILLFACTOR=70, PAD_INDEX=ON,
  ONLINE=ON, DATA_COMPRESSION = ROW |
  PAGE);
```

Physical Index B-Tree Layout

Clustered (BOL 2005 / 2008)



Non-Clustered (BOL 2005 / 2008)



How large are my indexes?

- SELECT *
- FROM sys.dm_db_index_physical_stats (db_id(), object_id('table_name'), null, null, 'detailed')
  - Database Id
  - Object Id
  - Index Id
  - Partition Number
  - Mode (NULL | Limited, Sampled, Detailed)

What Indexes are being used?

```
DECLARE @dbid INT
    , @dbName VARCHAR(100);

SELECT @dbid = DB_ID()
    , @dbName = DB_NAME();

WITH partitionCTE (OBJECT_ID, index_id, row_count, partition_count)
AS
(
    SELECT [OBJECT_ID]
        , index_id
        , SUM([ROWS]) AS 'row_count'
        , COUNT(partition_id) AS 'partition_count'
    FROM sys.partitions
    GROUP BY [OBJECT_ID]
        , index_id
)

SELECT OBJECT_NAME(i.[OBJECT_ID]) AS objectName
        , i.name
        , CASE
            WHEN i.is_unique = 1
                THEN 'UNIQUE '
            ELSE ''
          END + i.type_desc AS 'indexType'
        , ddius.user_seeks
        , ddius.user_scans
        , ddius.user_lookups
        , ddius.user_updates
        , cte.row_count
        , CASE WHEN partition_count > 1 THEN 'yes'
            ELSE 'no' END AS 'partitioned?'
        , CASE
            WHEN i.type = 2 And i.is_unique = 0
                THEN 'Drop Index ' + i.name
                    + ' On ' + @dbName
                    + '.dbo.' + OBJECT_NAME(ddius.[OBJECT_ID]) + ';'
            WHEN i.type = 2 And i.is_unique = 1
                THEN 'Alter Table ' + @dbName
                    + '.dbo.' + OBJECT_NAME(ddius.[OBJECT_ID])
                    + ' Drop Constraint ' + i.name
            ELSE ''
          END AS 'SQL_DropStatement'
FROM sys.indexes AS i
```

More Reading…

- http://mrdenny.com/res/table-indexing-net

Q & A

Denny Cherry

mrdenny@dcac.co
http://www.dcac.co
http://www.twitter.com/mrdenny