7 INDEXING TIPS TO IMPROVE SQL SERVER PERFORMANCE

BY PINAL DAVE

SQL Server Performance Tuning Expert at SQLAuthority.com

INTRODUCTION

SQL Server performance is always one of the most challenging subjects. Hard drives are getting cheaper and cheaper, and data is growing exponentially. With this new pattern, we all have a significant challenge. In the database world, though, we now have new problems affecting performance. Here are a few questions that keep coming up in various systems in the industry:

- A query which was running quickly now takes too long to run
- Our report is now running very slow
- During data import everything gets slow
- Every day during a specific period we are facing many deadlocks
- Queries are continuously responding with time-outs
- SELECT queries are slower when INSERT, UPDATE and DELETE are happening
- ...and many more...

Throughout the years I have seen and solved many similar issues. I have seen many bad practices implemented in SQL Server at the server and database level. There are hundreds of tips that one can practice to keep a database at optimal performance. The purpose of this document is to highlight a few best practices that can give maximum benefits to the SQL Server system. Out of thousands of best practices, I have selected the seven best practices related to indexes.

TIP ONE
DROP UNUSED INDEXES

Indexes are commonly created to gain additional performance from the system. It is very common for a new DBA to inherit index systems. When a new DBA gets a system which has been there for a long time, there are always lots of indexes already created.

New DBAs often do not have the understanding or documentation of why all of those indexes were created. The new DBA cannot drop indexes created by earlier admins and create a few more to accommodate new requests. The uncertain history of the indexes leads to the issue that there are way more indexes than needed.

Lots of unused indexes are an extra burden on SQL Server. Every time any field is updated when referenced in the index, the index also has to be updated. Updating the index is an additional load on the SQL Server engine. If you have an index maintenance script, it will even be wasting some resources on rebuilding/reorganizing indexes.

The best practice for unused indexes is to drop them. When dropping indexes, one has to be very careful that they do not drop any index which is useful to queries by mistake. It is recommended to evaluate this unused index script on the server after running it continuously for a while without restarting the services or server.

Unused index scripts are based on Dynamic Management Views (DMV) and will return lots of results. Select the top 2-3 indexes at a time and drop them on the development server. Keep your server on the watch for a while and, if you find it appropriate, drop them on the production server too.

You can download the script from here:
http://bit.ly/UnusedIndex

TIP TWO
CREATE MISSING INDEXES

Indexes are created to improve performance. As important as it is to drop unused indexes, it is also essential to create missing indexes. It is quite reasonable for developers to keep on writing optimal queries and changing the queries to suit new requests of the end users. The requirements of the end users always keep on improving, and developers follow up by changing the query.

The ever-changing queries have a circular effect on the indexes. The indexes created for specific queries become out-dated when the underlying queries change. As discussed in the earlier topic, we should drop the unused index. However, we must not forget to check the most relevant indexes against the recently running queries.

Missing index scripts provide the details of the indexes which are most beneficial to the queries. When any query plan is generated, it is always looking for the most optimal index. When the most optimal index is not found, SQL Server Engine uses another index which is the next best choice. It is a good idea to create an index which is going to be the most effective one.

The missing index script is based on DMV and will return lots of results. Select the top 2-3 indexes at a time and create them on the development server. Keep your server on the watch for a while and, if you find it appropriate, create them on production server too.

You can download the script from here:
http://bit.ly/MissingIndex

TIP THREE
REMOVING DUPLICATE INDEXES

Just like unused and missing indexes – duplicate indexes are another fundamental concept one must consider. As most developers inherit the database from previous developers, the understanding of the database schema and indexes is not usually at its best. Quite commonly, when a new index is needed developers do not look at the definition of the existing indexes; they just create the new index as per their requirement. Eventually, there will be multiple indexes with the same definition in the system.

There is absolutely no point to having two indexes with the same structure in any database system. This duplicate index not only takes up space on the hard drive but also reduces the performance. All the INSERT, UPDATE and DELETE queries will have to now update two similar sets of the data on every single occurrence. As there are duplicate indexes, only one of the indexes is used when any query is executing, making the duplicate index redundant.

The best practice is to drop the duplicate index and keep any database free from additional overhead. Again, please be sure to verify that the index is indeed a duplicate before dropping it.

You can download the script from here:
http://bit.ly/DuplicateIndex

TIP FOUR
SIGNIFICANCE OF CLUSTERED INDEXES

More than best practices, it is essential to understand the significance of the clustered index. The prevailing opinion and general best practice suggest that any table should have a clustered index. If your table is tiny and the database is of insignificant size, this property can be ignored. However, I suggest understanding the best practice related to clustered indexes.

Another common practice is to create clustered indexes on the columns which are often searched in the database. An additional advantage is that when the first set of the data is retrieved the next collection of the information which is commonly queried is placed adjacent. Clustered indexes also help improve performance when the unique values are retrieved.

You can read more about it here:
http://bit.ly/PKandIndex

Here are a few best practices for clustered indexes:

- Columns which have large numbers of unique, distinct data may be good candidates for clustered indexes.
- In an OLTP workload the standard practice is to create clustered indexes on the primary key as data is often looked up using the same keys.
- In SQL Server when a primary key is created, it automatically creates the clustered index if it does not already exist. There are some rare cases when primary keys are on different columns than clustered indexes. It is indeed a good practice to have clustered indexes on unique values (e.g., primary keys) as it will avoid adding a unique identifier on the clustered index.

Keep the width of the clustered index as narrow as possible.

TIP FIVE
COLUMN STORE INDEXES FOR DATA WAREHOUSING

There are two kinds of storage in the database: Row Store and Column Store. Row store does exactly as the name suggests – it stores rows of data on a page – and column store stores all the data in a column on the same page. These columns are much easier to search – instead of a script examining all the data in an entire row whether the data is relevant or not, column store queries need only to search a much smaller number of the columns.

Using an index storing only column data increases both search speed and hard drive use. Additionally, the column store indexes are heavily compressed, which translates to even more significant memory efficiency and faster searches.

Though this sounds very exciting, it does not mean that every single index should be converted from row store to column store index. One has to understand the proper places where to use row store or column store indexes.

A column store index stores each column in a separate set of disk pages, rather than storing multiple rows per page as data traditionally has been stored.

Here are two scenarios where column store indexes should be considered:

1) Use a clustered column store index to store fact tables or large dimensions table in data warehousing applications

2) Use a nonclustered column store index to perform real-time analysis of transaction applications

You can read more about it here:
http://bit.ly/ColumnStoreIndex

TIP SIX
SQL SERVER 2016/2017 CARDINALITY ESTIMATION

A common question that gets asked in many SQL Server forums is whether it is worth upgrading to SQL Server 2016 or SQL Server 2017. The answer is straightforward, and it is the one word – **YES!**

SQL Server 2016 and 2017 have introduced many new features which help SQL Server to run faster out of the box. When a user upgrades their database to the latest version of SQL Server, they can take advantage of the most recent algorithms and improvements in the SQL Server.

The most prominent improvement in SQL Server 2016/2017 is the updated cardinality estimation algorithm. The optimized cardinality algorithm guides SQL Server Optimizer Engine to come up with better execution plans. A better execution plan uses SQL Server indexes efficiently leading to better overall performance.

It is essential to remember to use the latest cardinality estimation algorithm; the database should be using the newest compatibility as well. The compatibility level setting for SQL Server 2016 should be set to 130, and for SQL Server 2017 it should be set to 140. If you are using SQL Server 2016 or 2017, but your database compatibility is set to the previous version of SQL Server, you may not be able to use the power of the latest improvements in algorithms in SQL Server engine.

You can read more about it here:
http://bit.ly/CompatLevel
and
http://bit.ly/Cardinality

TIP SEVEN
ADDITIONAL INDEXING BEST PRACTICES

As mentioned earlier there are many best practices related to indexes. I am listing a few here, in no particular order:

- The primary key is usually a good candidate for a clustered index.
- Keep the width of the index as narrow as possible.
- GUID is not a candidate for a clustered index (except in extreme cases) as it may lead to higher fragmentation and reduced performance.
- Any column which has high distinctive values (i.e. column is increasing, unique, identity column) is a good candidate for a clustered index.
- When creating multiple column indexes, it is usually a good idea to have the most selective columns as the first column in your index. (Verify this by a thorough test; this can vary case by case. When there are no other external conditional influences, this is considered as best practice.)
- Do not add indexes on every single column in the table; indexes should be carefully analyzed before creating them.
- Fill Factor 0 or 100 is the default value for the server. The index can be adjusted to the appropriate value. Higher fill factors for less frequently changed data and lower fill factors for more regularly changed data is recommended.

FINAL NOTE

Try all the queries on the development server first. Test all the changes on the development server and validate all the results. Deploy to production only after careful consideration. Take all the advice here as a best practice but not as a strict rule. And if needed, evaluate a third-party tool to help you with your SQL Server index performance issues.

ABOUT PINAL DAVE:

Pinal Dave is a SQL Server Performance Tuning Expert and an independent consultant. He has authored 11 SQL Server database books, taught 21 Pluralsight courses, and has written over 4500 articles on database technology on his blog at https://blog.sqlauthority.com. Along with 16+ years of hands-on experience, he holds a Masters of Science degree and many database certifications.

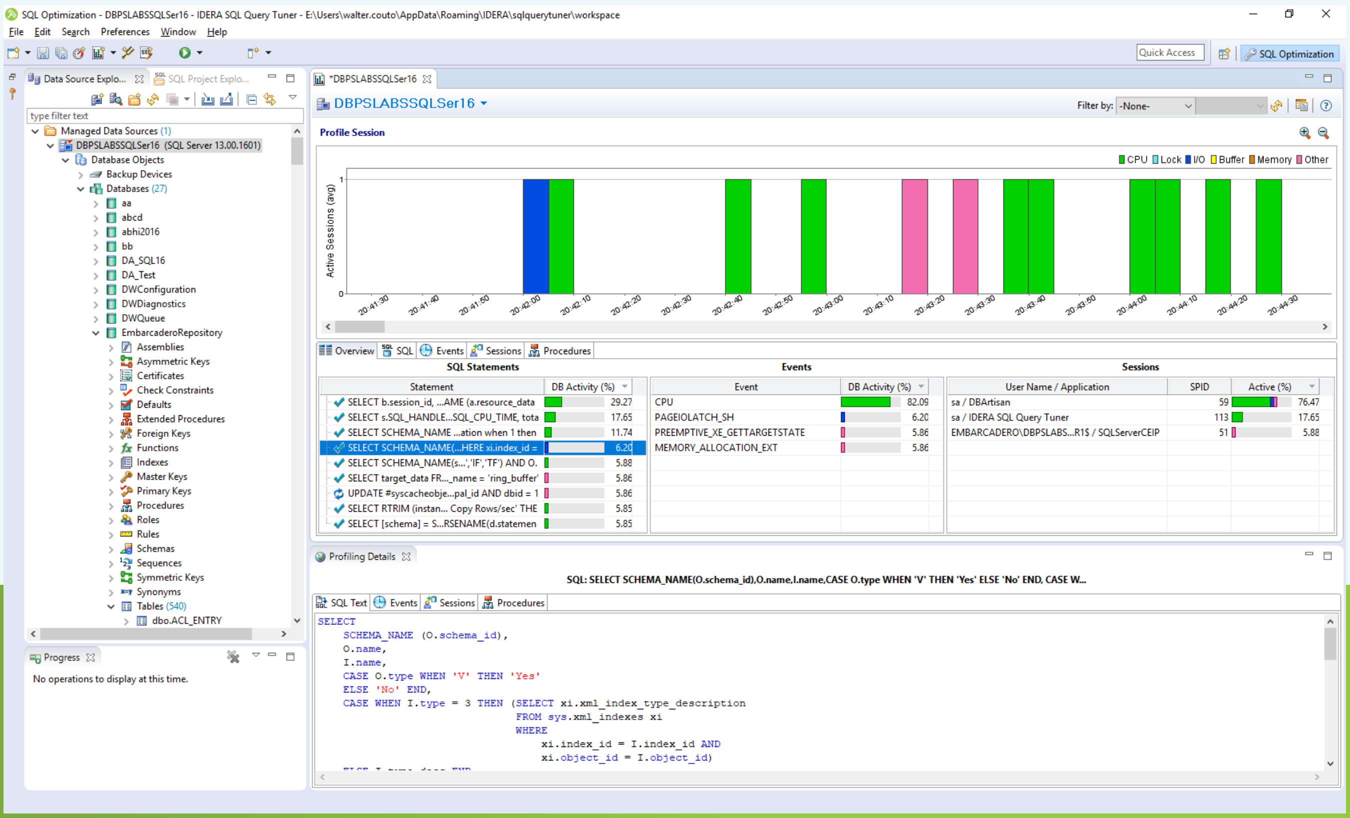
SQL QUERY TUNER

Automate SQL Tuning and Profiling

- **Streamline tuning of SQL code on SQL Server from one interface**
- **Tune SQL like a pro with automated performance optimization suggestions**
- **Tackle complex SQL queries with visual SQL tuning diagrams**
- **Pinpoint problem SQL with database profiling of wait-time analysis**
- **Load test alternative SQL queries in simulated production environment**

Start for FREE

(SQL Query Tuner is an add-on for SQL Diagnostic Manager for SQL Server)



SQL Statements

Statement	DB Activity (%)
SELECT b.session_id, ...AME (a.resource_data	29.27
SELECT s.SQL_HANDLE...SQL_CPU_TIME, tota	17.65
SELECT SCHEMA_NAME ...ation when 1 then	11.74
SELECT SCHEMA_NAME(...HERE xi.index_id =	6.20
SELECT SCHEMA_NAME(s...','IF','TF') AND O.	5.88
SELECT target_data FR..._name = 'ring_buffer'	5.86
UPDATE #syscacheobje...pal_id AND dbid = 1	5.86
SELECT RTRIM (instan... Copy Rows/sec' THE	5.85
SELECT [schema] = S...RSENAME(d.statemen	5.85

Events

Event	DB Activity (%)
CPU	82.09
PAGEIOLATCH_SH	6.20
PREEMPTIVE_XE_GETTARGETSTATE	5.86
MEMORY_ALLOCATION_EXT	5.86

Sessions

User Name / Application	SPID	Active (%)
sa / DBArtisan	59	76.47
sa / IDERA SQL Query Tuner	113	17.65
EMBARCADERO\DBPSLABS...R1$ / SQLServerCEIP	51	5.88

SQL: SELECT SCHEMA_NAME(O.schema_id),O.name,I.name,CASE O.type WHEN 'V' THEN 'Yes' ELSE 'No' END, CASE W...

```
SELECT
    SCHEMA_NAME (O.schema_id),
    O.name,
    I.name,
    CASE O.type WHEN 'V' THEN 'Yes'
    ELSE 'No' END,
    CASE WHEN I.type = 3 THEN (SELECT xi.xml_index_type_description
                           FROM sys.xml_indexes xi
                           WHERE
                               xi.index_id = I.index_id AND
                               xi.object_id = I.object_id)
```

IDERA.com

IDERA understands that IT doesn't run on the network – it runs on the data and databases that power your business. That's why we design our products with the database as the nucleus of your IT universe.

Our database lifecycle management solutions allow database and IT professionals to design, monitor and manage data systems with complete confidence, whether in the cloud or on-premises.

We offer a diverse portfolio of free tools and educational resources to help you do more with less while giving you the knowledge to deliver even more than you did yesterday.

Whatever your need, IDERA has a solution.

IDERA