**Index and Statistics**

**Explore database maintenance checks**

The query optimizer utilizes statistical information from the indexes to attempt to build the most optimal execution plan.

Within Azure SQL maintenance tasks such as backups and integrity checks are handled for you, and while you may be able to get away with automatic updates keeping your statistics up-to-date, sometimes it's not enough.

Having healthy indexes and statistics will ensure that any given plan will perform at optimal efficiency. Index maintenance should be performed regularly as data in your databases changes over time. You could change your index maintenance strategy based on the frequency of modifications to your data.

**Rebuild and reorganize**

Index fragmentation occurs when logical ordering within index pages doesn't match the physical ordering. Pages can out of order during routine data modification statements such as UPDATE, DELETE, and INSERT. Fragmentation can introduce performance issues because of the extra I/O that is required to locate the data that is being referenced by the pointers within the index pages.

As data is inserted, updated, and deleted from indexes the logical ordering in the index will no longer match the physical ordering inside of the pages, and between the pages, making up the indexes. Also, over time the data modifications can cause the data to become scattered or fragmented in the database. Fragmentation can degrade query performance when the database engine needs to read extra pages in order to locate needed data.

A reorganization of an index is an online operation that will defrag the leaf level of the index (both clustered and nonclustered). This defragmentation process will physically reorder the leaf-level pages to match the logical order of the nodes from left to right. During this process, the index pages are also compacted based on the configured fillfactor value.

A rebuild can be either online or offline depending on the command executed or the edition of SQL Server being utilized. An offline rebuild process will drop and re-create the index itself. If you can do so online, a new index will be built in parallel to the existing index. Once the new index has been built, the existing one will be dropped and then the new one will be renamed to match the old index name. Keep in mind that the online version will require more space as the new index is built in parallel to the existing index.

The common guidance for index maintenance is:

* **> 5% but < 30%** - Reorganize the index
* **> 30%** - Rebuild the index

Use these numbers as general recommendations. Depending on your workload and data, you may need to be more assertive, or in some cases you may be able to defer index maintenance for databases that mostly perform queries that seek specific pages.

The SQL Server and Azure SQL platforms offer DMVs that allow you to detect fragmentation in your objects. The most commonly used DMVs for this purpose are sys.dm\_db\_index\_physical\_stats for b-tree indexes, and sys.dm\_db\_column\_store\_row\_group\_physical\_stats for columnstore indexes.

One other thing to note is that index rebuilds cause the statistics on the index to be updated, which can further help performance. Index reorganization doesn't update statistics.

Microsoft introduced resumable rebuild index operations with SQL Server 2017. Resumable rebuild index operations option provides more flexibility in controlling how much time a rebuild operation might impose on a given instance. With SQL Server 2019, the ability to control an associated maximum degree of parallelism was introduced further providing more granular control to database administrators.

**Statistics**

When doing performance tuning in Azure SQL, understanding the importance of statistics is critical.

Statistics are stored in the user database as binary large objects (blobs). These blobs contain statistical information about the distribution of data values in one or more columns of a table or indexed view.

Statistics contain information about the distribution of data values within a column. The query optimizer uses column and index statistics in order to determine cardinality, which is the number of rows a query is expected to return.

Cardinality estimates are then used by the query optimizer to generate the execution plan. Cardinality estimates also help the optimizer determine what type of operation (for example, index seek or scan) to use to retrieve the data requested.

To see the list of user defined statistics with the last updated date, run the query below:

SQLCopy

SELECT sp.stats\_id,

name,

last\_updated,

rows,

rows\_sampled

FROM sys.stats

CROSS APPLY sys.dm\_db\_stats\_properties(object\_id, stats\_id) AS sp

WHERE user\_created = 1

**Create statistics**

When you have AUTO\_CREATE\_STATISTICS option to ON, the query optimizer creates statistics on the indexed column by default. The query optimizer also creates statistics for single columns in query predicates.

These methods provide high-quality query plans for most queries. At times, you may need to create more statistics using CREATE STATISTICS statement to improve specific query plans.

It's recommended to keep the AUTO\_CREATE\_STATISTICS option enabled as it will allow the query optimizer to create statistics for query predicate columns automatically.

Whenever you encounter the following situations, consider creating statistics:

* The Database Engine Tuning Advisor suggests creating statistics
* The query predicate contains multiple columns that aren't already in the same index
* The query selects from a subset of data
* The query has missing statistics

**Maintenance tasks automation**

Azure SQL provides native tools to perform database maintenance tasks for automation purposes. Different tools are available depending on the platform where the database is running.

**SQL Server on an Azure Virtual Machine**

You have access to scheduling services such as the SQL Agent or the Windows Task Scheduler. These automation tools can help keeping the amount of fragmentation within indexes to a minimum. With larger databases, a balance between a rebuild and a reorganization of indexes must be found to ensure optimal performance. The flexibility provided by SQL Agent or Task Scheduler allows you to run custom jobs.

**Azure SQL Database**

Due to the nature of Azure SQL Database, you don't have access to SQL Server Agent nor Windows Task Scheduler. Without these services, index maintenance must be created using other methods. There are three ways to manage maintenance operations for SQL Database:

* Azure Automation runbooks
* SQL Agent Job from SQL Server in an Azure Virtual Machine (remote call)
* Azure SQL elastic jobs

**Azure SQL Managed Instance**

As with SQL Server on an Azure Virtual Machine, you can schedule jobs on a SQL Managed Instance through SQL Server Agent. Using SQL Server Agent provides flexibility to execute code designed to reduce fragmentation within the indexes in the database.