/\* ------------------------------ \*\

© Copyright 2021 by Wingenious

read License information below

\\* ------------------------------ \*/

------

README

------

SQLFacts is a comprehensive suite of 28 tools with awesome features. The toolkit includes plenty to love for everybody, with tools for SQL Server database development, database administration, and performance tuning.

All of them consist of T-SQL code, so nothing is hidden from users.

All the code can be reviewed to see how things are done.

All the code can be customized to suit various needs and situations.

None of the tools modify the server or databases in any way. They are strictly read-only, other than using tempdb for some local temporary tables.

The SQLFacts suite of tools requires the user to have VIEW SERVER STATE permission.

The SQLFacts suite of tools was created using the practical experience of more than 20 years of work as a SQL Server database developer and administrator. This collection of tools is new, but routines like these have proven extremely useful for SQL Server database engineering and performance tuning for a very long time. The suite of tools is also a great way to learn more about how SQL Server operates by observing how different actions affect various result sets.

The functionality of these tools is not entirely unique. Some of the same functionality can be obtained from various articles written by very smart people with deep SQL Server experience. This new collection of tools has the advantage of obtaining everything all at once, from one source, with consistency in T-SQL coding style and format. These tools also have many unique and beneficial features. They are much more than slapped together, piecemeal, blog post scripts.

The SQLFacts suite of tools is intended for SQL Server 2012 and newer. However, most of them work fine with some prior versions. A few of the tools have a disabled block of SQL code to exchange with an active block for use with prior versions.

The SQLFacts suite of tools is great for SQL Server database development, database administration, and performance tuning.

The people who do development often focus on matters INSIDE the database(s). They are served by these tools: SQLFacts, Browse, References, Search, GenerateKeys, GenerateSQL.

The people who do administration often focus on matters OUTSIDE the database(s). They are served by these tools: BACKUP, RESTORE, MetricsNow, MetricsHistory, Auditor, SQLAgent, Bufferin, Databases, Sessions, Resources, Blocking, Locksmith, Deadlocks, AGLatency.

The people who do performance tuning might focus on matters OUTSIDE the database(s). They are served by these tools: MetricsNow, MetricsHistory, Auditor, SQLAgent, Bufferin, Databases, Sessions, Resources, Blocking, Locksmith, Deadlocks, AGLatency.

The people who do performance tuning might focus on matters INSIDE the database(s). They are served by these tools: Statistics, QueryHistory, IndexHistory, IndexActivity, IndexNeeds, IndexNeedsPlus, Indexer, Janitor.

The tools for performance tuning provide a wealth of detailed information for guiding performance tuning efforts.

They make it easy to identify patterns of resource contention or server issues.

They make it easy to identify opportunities for creating, dropping, or consolidating indexes.

They make it easy to identify which SQL routines and which SQL statements are consuming the most resources.

-------

License

-------

The SQLFacts suite of tools is provided "AS IS" and there's no warranty of any kind. The creator shall not be held liable for any claim of damages arising from use of the tools.

The SQLFacts suite of tools is the intellectual property of Wingenious and all rights are reserved by Wingenious.

The SQLFacts suite of tools may be modified for your own purposes and/or distributed inside your immediate organization, provided the original copyright notice remains intact.

The SQLFacts suite of tools may not be distributed outside your immediate organization, in whole, in part, or in any form, without explicit written permission from Wingenious.

Please contact license@SQLFacts.com for permission if you are interested in distributing parts of SQLFacts. In general, we would ask for only two small considerations, proper attribution and a link to the SQLFacts website.

----------------

General Comments

----------------

Several tools in the SQLFacts suite do their analysis on a single database (the current database). Those tools include a block of SQL code near the top that determines which schemas to include in the analysis. The default behavior is to include the dbo schema and all user schemas. There's a disabled block of SQL code to specify a list of schemas.

Several tools in the SQLFacts suite reference SQL Server DMVs (Dynamic Management Views) whose output would be affected by a restart. Those tools (and some others) include a block of SQL code near the top that determines which SQL Server version is running and when the instance was last restarted. The information is displayed in the Messages tab.

Several tools in the SQLFacts suite return a subset of the columns available. The columns included by default are generally the most useful, but sometimes there's a need for more information. The additional columns, if any, can be included by enabling them in the final SELECT statement. The column names try to balance between clarity and brevity.

SQLFacts is the name of the suite as a whole, but it's also the name of an individual tool in the suite. This documentation often qualifies the SQLFacts name to make it clear what's being discussed, but sometimes it's beneficial to pay attention to the context in which the SQLFacts name is used.

-------------------

General Information

-------------------

PK means primary key

AK means alternate key (unique constraint)

FK means foreign key

U means unique

UF means unique filtered

S means simple

SF means simple filtered

Index\_0 means a table as heap

Index\_1 means a table as clustered index

Index\_5 means a table as clustered index (columnstore)

Index\_2 means a nonclustered index

Index\_6 means a nonclustered index (columnstore)

index\_type (0) means a table as heap

index\_type (1) means a table as clustered index

index\_type (5) means a table as clustered index (columnstore)

index\_type (2) means a nonclustered index

index\_type (6) means a nonclustered index (columnstore)

table\_type (0) means a table as heap

table\_type (1) means a table as clustered index

table\_type (5) means a table as clustered index (columnstore)

GeneralType is the type for a major object, such as a table or a routine

GeneralSchema is the schema for a major object, such as a table or a routine

GeneralObject is the name for a major object, such as a table or a routine

SQLServerType is the type for a minor object, such as a column or a parameter

SQLServerName is the name for a minor object, such as a constraint (PK, AK, FK, check) or an index

SQLServerFile is the name for a filegroup

GeneralType (U ) means table

GeneralType (V ) means view

GeneralType (P ) means stored procedure

GeneralType (FN) means user-defined function, scalar

GeneralType (IF) means user-defined function, table-valued, inline

GeneralType (TF) means user-defined function, table-valued, multi-statement

GeneralType (TR) means trigger

GeneralType (SN) means synonym

GeneralType (SO) means sequence

ReferenceBy means object caller

ReferenceOf means object called

---------------

Getting Started

---------------

The SQLFacts suite of tools is very simple to use. There's absolutely nothing to install, anywhere, unless you choose to install the monitoring system (MetricsHistory). The tools consist of T-SQL code that can be run in SQL Server Management Studio (SSMS). The process involves opening one of the T-SQL code files with SSMS and clicking the Execute button.

The SQLFacts suite of tools can satisfy countless use cases. There are tools for database development, database administration, and performance tuning.

The SQLFacts tool might be used by developers and administrators to research objects in a database. For example, it's common to wonder which tables in a database are the largest. The question can be answered very quickly by a few columns in one result set.

The Browse, References, Search, GenerateKeys, and GenerateSQL tools might be used by developers on an interactive basis as they build or modify a database. For example, the Search tool can find a given string within the T-SQL code of database objects.

The BACKUP and RESTORE tools might be used by administrators to research database backup history and generate a lengthy set of RESTORE statements. For example, a set of RESTORE statements can be generated by simply choosing a particular transaction log BACKUP. The statements will RESTORE the previous full BACKUP, the following differential BACKUP (if any), and the intermediate transaction log BACKUPs (if any).

The MetricsNow, Sessions, Resources, Blocking, and Locksmith tools might be used by administrators for on-demand monitoring of what a SQL Server instance is currently doing. For example, the Sessions tool reveals what T-SQL code is running, who's running it, when it started, where it came from, and what resources it's using.

The MetricsHistory tool is a complete SQL Server monitoring system. It creates objects in a chosen database to support monitoring and researching what a SQL Server instance is doing over time. For example, it can send email alerts when Page Life Expectancy (or many other performance metrics) suddenly change by a significant amount.

The Auditor, SQLAgent, Bufferin, Databases, Deadlocks, and AGLatency tools might be used by administrators to investigate various components of a SQL Server instance. For example, the Databases tool provides many details about used/free space within each database.

The Statistics tool might be used by performance tuners to check whether index statistics are being refreshed in a timely manner.

The QueryHistory tool might be used by performance tuners to see which ad hoc T-SQL statements and stored procedures are consuming the most CPU resources.

The IndexHistory tool might be used by performance tuners to evaluate consolidating existing indexes and eliminating redundancy.

The IndexActivity tool might be used by performance tuners to research index access patterns and look for potential resource contention issues.

The IndexNeeds and IndexNeedsPlus tools might be used by performance tuners to evaluate the index suggestions provided by SQL Server itself and see which stored procedures contributed to those suggestions.

--------

SQLFacts

--------

• SQLFacts.sql

• SQLFacts.txt

• SQLFacts.exe

SQLFacts is a tool for conducting research on a SQL Server database. It also serves as a tool for generating documentation of the database architecture.

The SQLFacts SQL file is intended for SQL Server 2012 and newer. There's a disabled block of SQL code to exchange with an active block for use with some prior versions.

The SQLFacts SQL file contains the SQL code for SQLFacts. The SQL code generates more than 40 SQLFacts result sets (facts). The result sets can be modified by changing existing SQL statements. A new result set can be added by writing a new SQL statement, including a comment line and a PRINT line (following the pattern). The result sets can be viewed by executing the SQL code in SQL Server Management Studio. The Messages tab will contain a key/legend for the result sets. The result sets can be saved as tables by doing a simple global SQL code change.

The SQL code provides an easy way to customize which schemas are included in the analysis. There's a disabled list of schemas near the start of the SQL code. It can be exchanged for the SQL statement which includes all schemas by default.

The SQLFacts.txt file contains general information and descriptions of the SQLFacts result sets (facts).

The SQLFacts.exe file is a very small application that loads the SQLFacts SQL file and executes it in the context of the specified SQL Server database. The application captures the SQLFacts result sets (facts) and saves them in the specified location as HTML pages and CSV files. It also creates a TOC.htm file as a table of contents for the HTML pages it creates.

The SQLFacts tool is great for anybody who designs, develops, supports, or administers SQL Server databases.

------

Browse

------

The Browse SQL file is a tool for conducting research while doing database development or performance tuning. The schema and name of any table or routine (view, stored procedure, function, or trigger) is entered near the start of the SQL code.

If the specified object is a table then the Results tab will contain 12 SQLFacts result sets (facts) for the table. The columns in the first result set define the order and content of the subsequent result sets. The Messages tab will contain a key/legend for the result sets. The Messages tab will also contain some generated SQL statements to be used as templates for other SQL statements.

If the specified object is a routine (view, stored procedure, function, or trigger) then the Results tab will contain five SQLFacts result sets (facts) for the routine. The Messages tab will contain a key/legend for the result sets. The Messages tab will also contain the SQL code definition of the routine. The Messages tab will also contain some generated SQL statements to be used in an emergency if/when the routine suddenly runs much longer than expected. Try the sp\_recompile line first. Try the UPDATE STATISTICS line(s) next if the problem still persists. These SQL statements are NOT intended to be a substitute for careful analysis of the problem and appropriate performance tuning activity.

The SQL code provides an easy way to customize which schemas are included in the analysis. There's a disabled list of schemas near the start of the SQL code. It can be exchanged for the SQL statement which includes all schemas by default.

The SQLFacts.txt file contains general information and descriptions of the SQLFacts result sets (facts).

The Browse tool is great for SQL Server database development.

----------

References

----------

The References SQL file is a tool for conducting research while doing database development. The schema and name of any table or routine (view, stored procedure, function, or trigger) is entered near the start of the SQL code.

If the specified object is a table then the Results tab will contain two result sets. The first result set contains a row for every ancestor (parent, grandparent, more) of the table. The second result set contains a row for every descendant (child, grandchild, and more) of the table.

If the specified object is a routine (view, stored procedure, function, or trigger) then the Results tab will contain two result sets. The first result set contains a row for every ancestor (called, called by called, more) of the routine. The second result set contains a row for every descendant (caller, caller of caller, and more) of the routine.

The References tool is great for SQL Server database development.

------

Search

------

The Search SQL file is a tool for conducting research while doing database development. A search string is entered near the start of the SQL code.

The SQL code will find the search string within the SQL code definition of any routine (view, stored procedure, function, or trigger) in the current database. The routine definitions are parsed into lines and each line is compared to the search string.

The SQL code returns two result sets for the two main variants (see below) and one result set for the third variant.

Result set 1 is a summary of the search results. It contains the number of matching lines for each involved routine.

Result set 2 is the actual details of the search results. It contains every matching line of every involved routine.

The SQL code will find the search string within SQL Server Agent job steps. The subsystem for a matching step must be "TSQL" and the database must be the current database.

The GeneralSchema column will contain the job name.

The GeneralObject column will contain the job step name.

There are three variants of the Search SQL file:

Search\_Fast does not support any wildcard searches, but it's very fast.

Search\_Wild supports LIKE-style searches, but it's more time-consuming.

Search\_Name searches names of objects/columns. The single result set is different from the other two variants.

The Search tool is great for SQL Server database development.

------------

GenerateKeys

------------

The GenerateKeys SQL file generates DDL SQL statements to implement primary key constraints, foreign key constraints, and foreign key indexes.

The tool assumes the tables are using single-column surrogate primary keys.

The tool assumes the primary key column names are derived via the table names.

The tool assumes the primary keys migrate to child tables to become foreign keys.

The generated foreign key indexes are merely a starting point for a comprehensive indexing strategy. They are not intended to be a final set of indexes.

The GenerateKeys tool is great for SQL Server database development.

-----------

GenerateSQL

-----------

The GenerateSQL SQL file returns information for potential JOIN operations. It also generates a set of SELECT statements with JOIN clauses.

NOTE: All table-valued functions (TVFs) appear with DEFAULT parameter values in the generated SQL code.

There are four variants of the GenerateSQL SQL file:

GenerateSQL\_All generates SQL statements for all tables, views, and table-valued functions.

GenerateSQL\_2 generates SQL statements for a pair of objects with any connecting objects.

GenerateSQL\_IN generates INSERT statements for all tables.

GenerateSQL\_UP generates UPDATE statements for all tables.

GenerateSQL\_All:

The Messages tab will contain generated SELECT statements, one for each table, view, or table-valued function. The SELECT statements include JOIN clauses for parents and children. The SELECT statements are not intended to be used verbatim. Instead, copy column list elements and JOIN clauses from one SELECT statement, or several SELECT statements, to create a custom query as needed.

There's a variable near the start of the SQL code to determine if brackets are used on object/column names in the generated SQL code.

Result set 1 contains all unique indexes, whether they are primary keys, unique constraints, or standard unique indexes. They are the primary (parent) objects for JOIN operations. The involved columns are included.

Result set 2 contains all discernable ways to JOIN primary (parent) objects to foreign (child) objects. The foreign (child) objects can be tables, views, or table-valued functions. The potential JOIN operations come from foreign keys, or by matching column names and column data types with unique index keys. The involved columns are included.

GenerateSQL\_2:

The Messages tab will contain generated SELECT statements, one for each row in result set 1. The SELECT statement column list includes everything from each object in the query path. The SELECT statement FROM clause is for object 1, followed by a JOIN clause for each connecting object, followed by a JOIN clause for object 2. The SELECT statements can be used verbatiom, or modified as needed.

There's a variable near the start of the SQL code to determine if brackets are used on object/column names in the generated SQL code.

There are variables near the start of the SQL code to specify the schema names and object names for a pair of objects. The objects can be tables, views, or table-valued functions.

The result set contains a list of possible query paths between the pair of objects. The list includes only the shortest paths, involving the fewest JOIN operations.

GenerateSQL\_IN:

The Messages tab will contain generated INSERT statements, one for each table. The INSERT statements are not intended to be used verbatim.

There's a variable near the start of the SQL code to determine if brackets are used on object/column names in the generated SQL code.

GenerateSQL\_UP:

The Messages tab will contain generated UPDATE statements, one for each table. The UPDATE statements are not intended to be used verbatim.

There's a variable near the start of the SQL code to determine if brackets are used on object/column names in the generated SQL code.

The GenerateSQL tool is great for SQL Server database development.

----------------

BACKUP & RESTORE

----------------

The BACKUP SQL file and the RESTORE SQL file work together. They can be used when a RESTORE operation must be performed. They can be very handy when a point-in-time RESTORE operation must be done in an emergency.

The BACKUP SQL file returns two result sets drawing from the BACKUP history retained by SQL Server.

Result set 1 contains a summary of BACKUP operations.

Result set 2 contains a list of all BACKUP operations. Identify the row for the last BACKUP operation desired in a RESTORE operation. Copy the BACKUP\_ID (backup\_set\_id) value.

The RESTORE SQL file generates a series of RESTORE statements up through the BACKUP identified. Paste the BACKUP\_ID (backup\_set\_id) value from BACKUP SQL into the appropriate line of RESTORE SQL (near the start of the SQL code). Modify the next line of RESTORE SQL to specify a date/time for a point-in-time restore, if desired. Execute RESTORE SQL. Copy the contents of the SQLCode column of the result set, paste into another connection window, and adjust if/as necessary.

These two tools are great for SQL Server database administration.

----------

MetricsNow

----------

The MetricsNow SQL file is intended for SQL Server 2012 and newer. There are two disabled lines of SQL code to exchange with active lines for use with some prior versions.

The MetricsNow SQL file is a tool for an emergency situation to assist with diagnosing unexpectedly slow performance. It provides current values for many performance metrics.

There are metrics about RAM usage, SQL code plan caching, CPU activity, file read/write activity, wait statistics, and much more.

Please refer to the Metrics.txt file for more information.

The MetricsNow tool is great for SQL Server database administration.

--------------

MetricsHistory

--------------

The MetricsHistory SQL file is intended for SQL Server 2012 and newer. There are two disabled lines of SQL code to exchange with active lines for use with some prior versions.

The MetricsHistory SQL file creates 21 objects in the current database to implement a very capable SQL Server monitoring system. The system is a tool for gathering and examining historic performance metrics.

There are metrics about RAM usage, SQL code plan caching, CPU activity, file read/write activity, wait statistics, and much more.

Please refer to the Metrics.txt file for more information.

The MetricsHistory tool is great for SQL Server database administration.

-------

Auditor

-------

The Auditor SQL file returns information for SQL Server Audits. Result sets 1/2/3 contain an increasing level of detail. Result set 4 contains audit results. It's included only when there's exactly one audit with a file destination. The Messages tab will contain a generated SQL statement for each audit with a file destination.

Result set 1 contains information for SQL Server Audits.

Result set 2 contains information for SQL Server Audit server/database specifications.

Result set 3 contains information for SQL Server Audit server/database specification details, which includes what's being audited.

Result set 4 contains audit results, IF there's exactly one audit with a file destination.

The Auditor tool is great for SQL Server database administration.

--------

SQLAgent

--------

The SQLAgent SQL file returns basic information for SQL Server Agent jobs and job schedules. The information for schedules is presented in an hour-by-hour format for each day of the week. It provides a simple, visual representation for job starts during each hour of the day. A numeric value in an hour column is the minute for the first job start during the hour.

NOTE: This tool does not include every type of job schedule. It includes schedules with a Daily (every 1 day) frequency or a Weekly (every 1 week) frequency. These are the most common types of job schedules.

Result set 1 contains basic information for SQL Server Agent jobs.

Result sets 2 through 8 contain basic information for SQL Server Agent job schedules in a crosstab format for each day. The days of the week to include can be customized in a WHERE clause.

The SQLAgent tool is great for SQL Server database administration.

--------

Bufferin

--------

Bufferin is an analgesic for discomfort due to an abnormally small PLE... Please forgive us for trying to be funny!

The Bufferin SQL file returns a list of the data in the SQL Server buffer cache, aggregated by database object. The information reveals how much of which tables/indexes are currently available in the buffer cache. The information is useful for diagnosing small PLE values (see Page\_Life in Metrics).

Please be aware, this process may run for several minutes if the buffer cache is extremely large and fully loaded.

The Bufferin tool is great for SQL Server database administration.

---------

Databases

---------

The Databases SQL file returns information for all user databases (and tempdb) on the SQL Server instance.

The information includes owner name, recovery model, create date, detailed information about size/usage for data files and transaction logs, and dates for the most recent BACKUP operations.

Result set 1 includes owner name, recovery model, create date, and dates for the most recent BACKUP operations.

Result set 2 includes aggregated information about size/usage for data files.

Result set 3 includes aggregated information about size/usage for transaction logs.

Result set 4 includes aggregated information about size/usage for data files by filegroup.

Result set 5 includes information about size/usage for individual data files and the volumes where they are stored.

The Percent\_File column and the GBs\_ADD\_File column are mutually exclusive. Only one of them will contain a non-zero file growth value.

The Databases tool is great for SQL Server database administration.

--------

Sessions

--------

The Sessions SQL file returns information for all user connections (sessions) to the SQL Server instance. The SQL code can be quickly adjusted for the current need (see the WHERE clause), such as including idle sessions or including only lead blockers.

The information includes exactly what SQL code is running in the moment (SQL\_code), who's running it (login\_name), where it came from (host\_name), how it was executed (program\_name), when it started (batch\_time), blocker (blocking\_id), how much RAM is being used (GBs\_RAM), transaction state (trans\_state), and more.

There's a disabled result set which summarizes the waits.

There's a disabled result set which summarizes the waits by database.

A high amount of waiting of type PAGELATCH\_SH for tempdb may indicate a need to increase the number of tempdb data files.

A high amount of waiting of type PAGELATCH\_UP for tempdb may indicate a need to increase the number of tempdb data files.

A high amount of waiting of type PAGELATCH\_EX for tempdb may indicate a need to decrease the usage of temporary tables or use the memory-optimized tempdb metadata feature of SQL Server 2019.

A high amount of waiting of type PAGELATCH\_EX for user databases may indicate a need to reconsider the clustered index keys or use the OPTIMIZE\_FOR\_SEQUENTIAL\_KEY feature of SQL Server 2019.

There's an additional stand-alone query which returns information about user sessions/requests that are occupying tempdb space.

There's an additional stand-alone query which returns information about user sessions/requests that are occupying transaction log space.

The Sessions tool is great for SQL Server database administration.

---------

Resources

---------

The Resources SQL file returns information for processes that are consuming a large amount of memory and/or a large amount of tempdb space. It watches for such processes over a specified period of time. It's like a cross between MetricsNow and Sessions.

There are variables near the start of the SQL code to specify a date/time to begin collecting data, how many times to check for offending processes, and how long to wait between checks. The default values cause checking to begin immediately, doing 120 checks, with 30 seconds between checks. In other words, it monitors for offending processes for one hour.

There's a variable near the start of the SQL code to specify a threshold percentage of total amount (size) for memory. Any processes meeting or exceeding the threshold will be included in the results.

There's a variable near the start of the SQL code to specify a threshold percentage of total amount (size) for tempdb. Any processes meeting or exceeding the threshold will be included in the results.

Result set 1 is for excessive memory usage.

Result set 2 is for excessive tempdb usage.

The Resources tool is great for SQL Server database administration.

--------

Blocking

--------

The Blocking SQL file returns information for processes that have been blocking other processes for an excessive amount of time. It watches for such processes over a specified period of time. It's like a cross between MetricsNow and Sessions.

There are variables near the start of the SQL code to specify a date/time to begin collecting data, how many times to check for offending processes, and how long to wait between checks. The default values cause checking to begin immediately, doing 120 checks, with 30 seconds between checks. In other words, it monitors for offending processes for one hour.

There's a variable near the start of the SQL code to specify a threshold threshold number of seconds being blocked. Any processes blocked for this length of time, or longer, will be included in the analysis.

The result set is a list of blocking processes with a summary of the amount of blocking each one caused during the specified period of time.

The Blocking tool is great for SQL Server database administration.

---------

Locksmith

---------

The Locksmith SQL file returns information for locks, granted and/or waiting, in the SQL Server instance. The SQL code can be quickly adjusted for the current need (see the WHERE clause), such as including idle sessions or including only lead blockers.

The information includes who's running the SQL code (login\_name), where it came from (host\_name), how it was executed (program\_name), when it started (batch\_time), and what database resources are involved in the locks.

The information is very beneficial for determining the reason(s) for blocking caused by lock contention.

The Locksmith tool is great for SQL Server database administration.

---------

Deadlocks

---------

The Deadlocks SQL file returns information about recent deadlocks. The default behavior is to examine the "system\_health" extended event session, but it supports a custom extended event session as well. The SQL file includes a disabled block of code for creating a custom extended event session to capture the "xml\_deadlock\_report" event.

NOTE: This tool requires SQL Server 2012 or newer.

There's a variable near the start of the SQL code to specify a maximum number of minutes to look back for deadlocks that have occurred. The default is one day. The amount of history available is dependent on several variables.

The routine returns three result sets. In each case the rows occur in pairs, two rows for each deadlock. The rows for each deadlock are for the processes involved in the deadlock.

Result set 1 is for general information and identifying which process was the deadlock victim.

Result set 2 is for the SQL code (often stored procedures) involved in the deadlock.

Result set 3 is for the database resources involved in the deadlock.

There's a disabled SELECT statement which combines all three result sets. It may be a bit easier to use if there are many deadlocks to research.

The Deadlocks tool is great for SQL Server database administration.

---------

AGLatency

---------

The AGLatency SQL file returns information for databases in Always On Availability Groups. The SQL code must be run on the primary replica.

Result set 1 is for transaction log usage of each database and general information for the Availability Group it's in.

Result set 2 is for latency between the primary replica server and the secondary replica server(s), for each database.

The AGLatency tool is great for SQL Server database administration.

----------

Statistics

----------

The Statistics SQL file returns information for statistics associated with indexes and columns. The statistics guide the query optimizer as it prepares execution plans.

The information for each statistic includes date/time when last updated, number of rows when last updated, number of changes since last updated, threshold where SQL Server would automatically perform an update, and a generated UPDATE STATISTICS statement. The threshold value is a number of changes since last updated. SQL Server will automatically perform an update after a certain number of changes (rows) have occurred. This tool can be used to determine when the threshold of changes is being approached and prepare to perform UPDATE STATISTICS at a more opportune time. This is more likely with versions prior to SQL Server 2016.

There's a variable near the start of the SQL code to specify a percentage of the threshold. This affects whether the generated UPDATE STATISTICS statement is disabled or not.

Result set 1 is for statistics associated with indexes.

Result set 2 is for statistics associated with columns.

The Statistics tool is great for SQL Server performance tuning.

------------

QueryHistory

------------

The QueryHistory SQL file returns information for SQL statements (or stored procedures) that have been executed recently. These performance statistics are retained by SQL Server until the instance is restarted or until the execution plan is evicted from the cache. The historical information can be very beneficial for identifying opportunities for performance improvement. It allows you to quickly focus your attention on the most expensive SQL statements (or stored procedures).

When evaluating how expensive a routine is, it's not just about how long the code runs, it's also about how often the code is executed. The Seconds\_SUM column, which represents the total time for all executions combined, is used for filtering and ordering. The number of reads/writes is also an important factor in evaluating performance.

There's a variable near the start of the SQL code to specify a minimum total run time (or CPU time), in seconds, for including a query or object in the list. It filters on the Seconds\_SUM column.

There's a variable near the start of the SQL code to quickly switch between run time and CPU time.

Result set 1 is for individual SQL statements.

Result set 2 is for stored procedures.

The QueryHistory tool is great for SQL Server performance tuning.

------------

IndexHistory

------------

The IndexHistory SQL file returns a list of all existing rowstore indexes. The information for each index includes name, type, row count, storage size, column definitions, redundancy indicator, and any usage statistics that have accumulated since the last restart of the SQL Server instance. This information is very beneficial for identifying opportunities for consolidating, or simply removing, redundant indexes or unused indexes.

Please be aware there are some additional columns available, currently disabled to minimize horizontal scrolling. They are very useful for certain tasks.

The IndexHistory tool is great for SQL Server performance tuning.

-------------

IndexActivity

-------------

The IndexActivity SQL file returns a list of all existing rowstore indexes. The information for each index includes name, type, row count, storage size, and many low-level performance statistics that have accumulated with recent activity, often since the last restart of the SQL Server instance. This information is very beneficial for analyzing index access patterns, identifying areas of resource contention, and resolving problems.

Please be aware there are some additional columns available, currently disabled to minimize horizontal scrolling. They are very useful for certain tasks.

The IndexActivity tool is great for SQL Server performance tuning.

----------

IndexNeeds

----------

The IndexNeeds SQL file returns a list of "missing indexes" that SQL Server thinks would be beneficial. These potential indexes are only suggestions. They should be considered in the context of the overall indexing strategy. SQL Server does not provide information about which stored procedures contributed to the index suggestions for versions prior to SQL Server 2019.

There's a variable near the start of the SQL code to specify a minimum impact value (Benefit) for the suggested index.

Result set 1 is ordered by index benefit, for database/schema/object and then for index.

Result set 2 is ordered by name, for database/schema/object and then for index key.

Result set 3 is information about which stored procedures contributed to the index suggestions (SQL Server 2019 and newer).

The IndexNeeds tool is great for SQL Server performance tuning.

--------------

IndexNeedsPlus

--------------

The IndexNeedsPlus SQL file returns a list of "missing indexes" that appear in execution plans. In many cases (depending upon several factors), the information is very similar to IndexNeeds (see above). This routine provides an easy method to determine which SQL statements or stored procedures contributed to the index suggestions for versions prior to SQL Server 2019.

There's a variable near the start of the SQL code to specify a minimum datetime (Last\_Run) for when the plan was last used.

Result set 1 is for individual SQL statements.

Result set 2 is for stored procedures.

The IndexNeedsPlus tool is great for SQL Server performance tuning.

-------

Indexer

-------

The Indexer tool returns information about existing indexes. It also generates SQL statements to (re)create or maintain existing indexes.

There are three variants of the Indexer tool:

Indexer\_Columns provides a list of all columns used in indexes.

Indexer\_DBAdmin assists with maintaining existing indexes.

Indexer\_SQLCode generates scripts for existing indexes.

Indexer\_Columns:

The result set includes schema/table/column names, data type, properties, number of occurrences in various index types, number of occurrences in various positions, and number of times used as a descending key. This information is very beneficial for examining an indexing strategy at a column level.

Indexer\_DBAdmin:

The result set includes schema/table/index names, partition number (if applicable), and percentage of index fragmentation for each index that meets the specified criteria (see below). The messages tab will contain a generated SQL statement for each index that meets the specified criteria (see below). The SQL statement will do an INDEX REORGANIZE or an INDEX REBUILD, depending upon the amount of index fragmentation.

NOTE: Indexer\_DBAdmin is not intended to provide all the functionality necessary for a comprehensive server/database maintenance program. It's a quick-and-dirty utility for a database administrator.

There's a variable near the start of the SQL code to specify the minimum number of pages an index must have to qualify for possible maintenance.

There's a variable near the start of the SQL code to specify the minimum percentage of index fragmentation for an INDEX REORGANIZE statement.

There's a variable near the start of the SQL code to specify the minimum percentage of index fragmentation for an INDEX REBUILD statement.

Indexer\_SQLCode:

The messages tab will contain two generated SQL statements for each index, in a standardized format. The SQL statements will do a DROP INDEX and a CREATE INDEX to match the index as it currently exists. The CREATE INDEX statement handles the vast majority of common index configurations, but not all of the many possibilities.

NOTE: Indexer\_SQLCode does not include indexes that are part of primary keys or unique constraints.

The Indexer tool is great for SQL Server performance tuning.

-------

Janitor

-------

The Janitor SQL file returns information about read/write activity for tables/indexes. The information includes name, type, storage size, number of days since the last occurrence of various operations (seeks/scans/lookups/updates), and a generated DROP statement. The tool makes it easy to remove debris from the database. The debris is abandoned tables and unused indexes. The tool can identify tables that are not being accessed for any read/write operations and identify indexes that are not being accessed for any read operations while being accessed for some write operations.

NOTE: The initial releases of SQL Server 2012 incorrectly remove data from the sys.dm\_db\_index\_usage\_stats DMV during an index rebuild. Be sure to use the latest release of SQL Server 2012.

NOTE: The initial releases of SQL Server 2014 incorrectly remove data from the sys.dm\_db\_index\_usage\_stats DMV during an index rebuild. Be sure to use the latest release of SQL Server 2014.

There's a variable near the start of the SQL code to specify a minimum number of days without read/write activity for table DROP statement eligibility.

There's a variable near the start of the SQL code to specify a minimum number of days without read/write activity for index DROP statement eligibility.

Result set 1 is for tables/indexes with 0 (zero) seeks/scans/lookups within the minimum number of days AND 0 (zero) updates within the minimum number of days. This result set includes an enabled DROP statement for tables and a disabled DROP statement for indexes.

Result set 2 is for tables/indexes with 0 (zero) seeks/scans/lookups within the minimum number of days AND > 0 (zero) updates within the minimum number of days. This result set includes a disabled DROP statement for tables and an enabled DROP statement for indexes.

Result set 3 is for tables/indexes with > 0 (zero) seeks/scans/lookups within the minimum number of days. This result set does not include a generated DROP statement.

There are two variants of the Janitor SQL file:

Janitor\_Tables examines tables and all the associated indexes as a unit. Result set 1 includes an enabled DROP statement. Result set 2 includes a disabled DROP statement. Result set 3 includes no DROP statement.

Janitor\_Indexes examines individual indexes other than the actual tables. Result set 1 includes a disabled DROP statement. Result set 2 includes an enabled DROP statement. Result set 3 includes no DROP statement.

The Janitor tool is great for SQL Server performance tuning.