Microsoft SQL Server Decision Support (DSS) Load Testing

This guide gives you an introduction to conducting Decision Support or analytical workloads on the Microsoft SQL Server Database. This guide will equip you with the essentials for assessing the ability of any system that runs SQL Server for processing query based workloads, such workloads are also known as Data Warehousing, Business Intelligence and Analytical workloads. On completion of this guide you will be able to run detailed and comprehensive query based SQL Server load tests. If you have not already done so you should read the Introduction to Decision Support, Data Warehousing, Business Intelligence, and Analytical Load Testing for all Databases before proceeding with this guide.

You should ensure that your version of SQL Server supports both Parallel Query and for SQL Server 2012 and 2014 the In-Memory Columnstore Index Option. SQL Server Enterprise Edition does, however other versions do not and are therefore not suitable for running Query based workloads. This DSS HammerDB workload is the ideal workload for testing the features of In-Memory Column Stores.

Introduction .................................................................................................................. 1
SUT Database Server Configuration .............................................................................. 2
Installation and Configuration ...................................................................................... 2
SUT Database Server Installation .................................................................................. 2
Creating the Test Schema ........................................................................................... 6
Build Options .............................................................................................................. 10
Starting the Schema Build ........................................................................................ 12
Pre-Testing and Planning ........................................................................................... 19
Driver Options ............................................................................................................ 21
Loading the Driver Script .......................................................................................... 22
Pre-Test 1 Running a Power Test Query .................................................................. 24
Pre-Test 2 Optimal DOP and In-memory for Power and Throughput Tests ............ 29
Running the DSS Tests ............................................................................................... 34
Power Test ................................................................................................................... 34
Throughput Test ......................................................................................................... 38
Calculating QphH ....................................................................................................... 41
Support and Questions .............................................................................................. 42

Introduction

The basis of Analytic or Decision Support Systems is the ability to process complex ad-hoc queries on large volumes of data. Processing this amount of data within a single process or thread on traditional row-oriented database is time consuming. Consequently SQL Server employs Parallel Query to break down such queries into multiple sub tasks to complex the query more quickly. Additional features such as compression and partitioning are also used with Parallel Query to improve performance. From SQL Server version there 2012 is also the option of utilising in-memory columnar storage. In-memory column stores can offer significant performance benefits and reduce query times for analytic workloads. Note that the SQL Server in-memory columnar storage is also used in conjunction with parallel query. As a consequence when planning analytic workloads for optimal performance you should consider both your in-memory (from SQL Server 2012) and parallel execution configuration. In similarity to the HammerDB OLTP workload,
HammerDB implements a fair usage of a TPC workload however the results should not be compared to official published TPC-H results in any way.

**SUT Database Server Configuration**

For query based workloads there is no requirement for a load testing client although you may use one if you wish. It is entirely acceptable to run HammerDB directly on the SUT (System Under Test) Database system if you wish, the client workload is minimal compared to an OLTP workload. In the DSS workload the client sends long running queries to the SUT and awaits a response therefore requiring minimal resources on the client side. As with an OLTP configuration however the database server architecture to be tested must meet the standard requirements for a SQL Server system. Similarly SQL Server can be installed on any supported operating system, there is no restriction on the version of SQL Server that is required.

Before running a HammerDB DSS test depending on your configuration you should focus on memory and I/O (disk performance). Also in turn the number and type of multi-core and multi-threaded processors installed will have a significant impact on parallel performance to drive the workload. When using in-memory column store features processors that support SIMD/AVX instructions sets are also required for the vectorisation of column scans.

HammerDB by default provides TPC-H schemas at Scale Factors 1,10,30,100,300 and 1000 (larger can be configured if required). The Scale Factors correspond to the schema size in Gigabytes. As with the official TPC-H tests the results at one schema size should not be compared with the results derived with another schema size. As the DSS workload utilizes parallel query it is possible for a single virtual user to use all of the hardware resources on the SUT at any schema size. Nevertheless there is still a relation with all of the hardware resources available including memory and I/O and a larger system will benefit from tests run a larger schema size. The actual sizing of hardware resources of hardware resources is beyond the scope of this document however at the basic level with traditional parallel execution and modern CPU capabilities I/O read performance is typically the constraining factor. Note that also in contrast to an OLTP workload high throughput transaction log write performance is not a requirement, however in similarity to the OLTP workload storage based on SSD disks will usually offer significant improvements in performance over standard hard disks although in this case it is the benefits of read bandwidth as opposed to the IOPs benefits of SSDs for OLTP. When using the in-memory column store memory capacity and bandwidth feature and if fully cached in memory storage performance is not directly a factor for query performance. Nevertheless data loads are an important consideration for in-memory data and therefore I/O and SSD read performance remain important for loading the data into memory to be available for scans.

**Installation and Configuration**

This sections describes the procedure to install and configure the Load Generation Server (if one is used) and the SUT Database Server.

**SUT Database Server Installation**

Installation and configuration of SQL Server on your chosen operating system is beyond the scope of this document. You should have the Microsoft SQL Server software installed, a test database created and running and be aware of the database features you wish to configure for query performance. During the installation make a note of your connection information, you will need it for the test schema creation. You may at your discretion use an existing database however that database must be empty before proceeding.
with a schema build. When using parallel query ensure that the instance is configured for parallel execution. Under Server Properties on the advanced tab, check the Max Degree of Parallelism. Note that the setting defines the number of processors to use as opposed to the number of parallel threads as shown in Figure 1.

![Figure 1 Max Degree of Parallelism](image1)

Once the schema is built you can also verify from the execution plan as shown in Figure 2 that parallelism is being used.

![Figure 2 Parallel execution plan](image2)

The execution plan in Figure 2 also illustrates an additional feature to enhance query performance, namely in-memory columnstore indexes that you can also create after the schema is built. Within these you also have two options nonclustered columnstore indexes and clustered columnstore indexes. Within SQL Server 2012 only nonclustered columnstore indexes are available and both are supported within SQL Server 2014. It is important to be aware that nonclustered columnstore indexes are non-updateable. This means that firstly they must be created after the schema is built and additionally do not support the running of the refresh functions detailed further in this document. Similarly within HammerDB clustered columnstore indexes should also be created after the schema is built however these do support updating and therefore
the refresh functions. After the schema build to create a columnstore index using SQL Server Management Studio select New Index and clustered or nonclustered columnstore index. For each table add all columns to the columnstore as shown in Figure 3.

Figure 3 Create Columnstore Index

And ensure that you have configured the degree of parallelism to the number of processors you wish to use as shown in Figure 4.

Figure 4 Columnstore DOP

You can run the schema creation from the Management Studio however you may experience a SQL Server
Management Studio timeout for larger indexes. If this is the case select the option to script the result and run the columnstore index creation from here as shown in Figure 5.

Figure 5 Columnstore Script

Note that you should leave existing indexes in place (except where clustered indexes are replaced with columnstores) and add columnstore to all tables. The following query:

```sql
select so.name as TableName, si.name as IndexName, si.type_desc as IndexType
from sys.indexes si
join sys.objects so on si.[object_id] = so.[object_id]
where so.type = 'U'
and si.name is not null
order by so.name, si.type
```

will produce results such as follows:

<table>
<thead>
<tr>
<th>TableName</th>
<th>IndexName</th>
<th>IndexType</th>
</tr>
</thead>
<tbody>
<tr>
<td>customer</td>
<td>customer_pk CLUSTERED</td>
<td></td>
</tr>
<tr>
<td>customer</td>
<td>NonClusteredColumnStoreIndex-20120427-163646 NONCLUSTERED</td>
<td></td>
</tr>
<tr>
<td>lineitem</td>
<td>l_shipdate_ind</td>
<td>CLUSTERED</td>
</tr>
<tr>
<td>lineitem</td>
<td>l_orderkey_ind</td>
<td>NONCLUSTERED</td>
</tr>
<tr>
<td>lineitem</td>
<td>l_partkey_ind</td>
<td>NONCLUSTERED</td>
</tr>
<tr>
<td>lineitem</td>
<td>NonClusteredColumnStoreIndex-20120427-155924 NONCLUSTERED</td>
<td></td>
</tr>
<tr>
<td>nation</td>
<td>nation_pk</td>
<td>CLUSTERED</td>
</tr>
<tr>
<td>nation</td>
<td>n_regionkey_ind</td>
<td>NONCLUSTERED</td>
</tr>
<tr>
<td>orders</td>
<td>o_orderdate_ind</td>
<td>CLUSTERED</td>
</tr>
<tr>
<td>orders</td>
<td>orders_pk</td>
<td>NONCLUSTERED</td>
</tr>
<tr>
<td>orders</td>
<td>NonClusteredColumnStoreIndex-20120427-162558 NONCLUSTERED</td>
<td></td>
</tr>
<tr>
<td>part</td>
<td>part_pk</td>
<td>CLUSTERED</td>
</tr>
<tr>
<td>part</td>
<td>NonClusteredColumnStoreIndex-20120427-163749 NONCLUSTERED COLUMNSTORE</td>
<td></td>
</tr>
<tr>
<td>partsupp</td>
<td>partsupp_pk</td>
<td>CLUSTERED</td>
</tr>
</tbody>
</table>
and as shown as in Figure 6, their use can be verified by the execution plan.

![Columnstore Execution Plan](image)

**Figure 6 Columnstore Execution Plan**

Once you have decided on your plan for the Degree of Parallelism and use of Columnstore indexes proceed with creating the test schema with the awareness of these enhancements you can make for performance once the build is complete.

### Creating the Test Schema

To create the DSS test schema based on the TPC-H specification you will need to select which benchmark and database you wish to use by choosing select benchmark from under the TPC menu. The initial settings are determined by the values in your config.xml file. Select MSSQL Server and TPC-H and press OK as shown in Figure 7.
Figure 7 Select Benchmark

Confirm the benchmark by pressing OK as shown in Figure 2.

Figure 8 Confirm Options

HammerDB will now be ready to create and run a TPC-H based workload as shown in Figure 9.
As with the OLTP workloads if selected from the top menu the schema options window is divided into two sections. The “Build Options” section details the general login information and where the schema will be built and the “Driver Options” for the Driver Script to run after the schema is built. “Build Options” are of importance at this stage and “Driver Options” will be considered further in this guide, however note that you don’t have to rebuild the schema every time you change the “Driver Options”, once the schema has been built only these “Driver Options” may need to be modified and therefore you can also select to view the Build Options only from the treeview as shown in Figure 10.
Figure 10 Build Options

For the “Build Options” fill in the values according to the database where the schema will be built as shown in Figure 11. (Note that for illustrative purposes SQL Server Express is used however this version does not support Parallel Query and you should therefore use Enterprise Edition).
Figure 11 Schema Options

**Build Options**

The Build Option values have the following meanings.

**SQL Server**

The Microsoft SQL Server is the host name or host name and instance that your load generation server will use to connect to the database running on the SUT database server.

**Microsoft SQL Server Port**

The Microsoft SQL Server Port is the network port that your load generation server will use to connect to the database running on the SUT database server. In most cases this will be the default port of 1433 and will not need to be changed.

**SQL Server ODBC Driver**

You will have configured Microsoft SQL Server during installation to authenticate either with Windows Authentication or with SQL Server and Windows Authentication. HammerDB will permit either method to be used however you must have the corresponding configuration on your SQL Server. Additionally your chosen method of authentication is required to be compatible with your chosen ODBC driver. To discover the available drivers use the ODBC Data Source Administrator tool as shown in Figure 12.
The driver name should be entered into HammerDB (as detailed further in this document) exactly as shown in the Data Source Administrator. The default value is “SQL Server Native Client 11.0”. For earlier versions of SQL Server modify this value accordingly. If you specify the ODBC Driver only as shown in Figure 13:

![ODBC Data Source Administrator](image)

**Figure 12 ODBC Data Source Administrator**

You must then use SQL Server (and not Windows Authentication) with a valid username and password otherwise SQL Server will be unable to log on to the system.

Error in Virtual User 1: 28000 18456 {{Microsoft][ODBC Driver 11 for SQL Server][SQL Server]Login failed for user "".

**Authentication**

As detailed previously in this document you will have configured SQL Server for Windows or Windows and SQL Server Authentication. If you specify Windows Authentication then SQL Server will use a trusted connection to your SQL Server using your Windows credentials without requiring a username and password. If SQL Server Authentication is specified and SQL Authentication is enabled on your SQL Server then you will be able connect by specifying a username and password that you have already configured on your SQL Server.

**SQL Server User ID**

The SQL Server User ID is the User ID of a user that you have already created on your SQL Server.
SQL Server User Password
The SQL Server User Password is the Password configured on the SQL Server for the User ID you have specified. Note that when configuring the password on the SQL Server there is a checkbox that when selected enforces more complex rules for passwords or if unchecked enables a simple password such as “admin”.

SQL Server TPCH Database
The SQL Server Database is the name of the Database to be created on the SQL Server to contain the schema. If this database does not already exist then HammerDB will create it, if the database does already exist and the database is empty then HammerDB will use this existing database. Therefore if you wish to create a particular layout or schema then pre-creating the database and using this database is an advanced method to use this configuration.

MAXDOP
The MAXDOP setting defines the Maximum degree of parallelism to be set as a default on the schema objects.

Scale Factor
The Scale Factor is selected by a radio button with a choice of scale factors of 1, 10, 30, 100, 300 and 1000 corresponding to 1GB, 10GB, 30GB, 100GB and 1000GB respectively (larger schema sizes can also be created) although the required space will be larger than these values due to the indexes required.

Virtual Users to Build Schema
The Virtual Users to Build Schema is the number of Virtual Users to be created on the Load Generation Server that will complete your multi-threaded schema build. You should set this value to the number of cores/Hyper-Threads on your Load Generation Server or SUT if HammerDB is running there.

Starting the Schema Build
When you have completed your Build Options click OK to store the values you have entered. For a permanent record the values can be entered directly into the config.xml file. On starting HammerDB the schema options will already contain the values you have entered in the corresponding fields, for example:

```xml
<schema>
    <mssqls_scale_fact>1</mssqls_scale_fact>
    <mssqls_maxdop>2</mssqls_maxdop>
    <mssqls_tpch_dbase>tpch</mssqls_tpch_dbase>
    <mssqls_num_tpch_threads>1</mssqls_num_tpch_threads>
</schema>
```

To begin the schema creation at the buttons in the top level window click the "Create TPC Schema" button. as shown in Figure 14.
On clicking this button a dialogue box such as the one shown in Figure 15 appears.

When you click Yes HammerDB will login to your chosen service name with a monitor thread as the system user and create the user with the password you have chosen. It will then log out and log in again as your chosen user, create the tables and then load the region and nation table data before waiting and monitoring the other threads. The worker threads will wait for the monitor thread to complete its initial work. Subsequently the worker threads will create and insert the data for their assigned warehouses as shown in Figure 16. There are no intermediate data files or manual builds required, HammerDB will both create and load your requested data dynamically. Data is inserted in a batch format for optimal performance.
Figure 16 Schema Building

When the workers are complete the monitor thread will create the indexes and gather the statistics. When complete Virtual User 1 will display the message TPCH SCHEMA COMPLETE and all virtual users will show that they completed their action successfully, press the button to destroy the virtual users as shown in Figure 17.
Figure 17 Schema Build Complete

And clear the script editor as shown in Figure 18.
Figure 18 Schema Creation End

The schema build is now complete with the following tables created and populated. If required the columnstore indexes can also be created at this point in time.

The TPC-H schema creation script is a standard HammerDB script like any other so you can save it, modify it and re-run it just like any other HammerDB script. For example if you wish to create more than a scale factor 1000 schema you may notice that the last line in the script calls a procedure with all of the options that you gave in the schema options. Therefore change the second value to any larger scale factor, for example the following will create a scale factor 10000 schema.

```
  do_tpch {(local)\SQLEXPRESS} 1433 10000 {SQL Server Native Client 11.0} windows
  sa admin tpch 2 4
```

Similarly change any other value to modify your script. If you have made a mistake simply delete the TPC-H schema as shown in Figure 19 and recreate it.
When you have created your schema you can verify the contents with SQL*PLUS or your favourite admin tool as the newly created user.

C:\Users>sqlcmd -S (local)\SQLEXPRESS -E -Q "use tpch; select name from sys.tables"
Changed database context to 'tpch'.

name

- customer
- lineitem
- nation
- part
- partsupp
- region
- supplier
- orders

(8 rows affected)

At this point the data creation is complete and you are ready to start running a performance test. Before doing so note that as this is a query based workload you have the potential to run multiple tests and it will
return the same results as the data is not modified during tests, however there is one exception, under the Driver Options the option to choose a Refresh Function. Further details will be given on the refresh function in the next section however at this point it is sufficient to note that the refresh function when enabled will modify data and no two same refresh functions can be run on the same data set. This means if you choose to include a refresh function and then attempt to re-run the test you will receive an error as the data has been modified. Therefore as shown in Figure 20 you should backup your schema before running a workload.

![Backup TPCH Schema](image)

**Figure 20 Backup TPCH Schema**

And if you have run a refresh function, restore the workload as shown in Figure 21.
Ensuring that you have a backup means you are prepared to restore your schema more quickly to the previous state before running subsequent tests as opposed to creating a new schema each time.

**Pre-Testing and Planning**

For DSS tests as with OLTP tests, after schema creation but before you start running measured tests an important phase is pre-testing and planning. Pre-testing is a phase also known as ‘testing the tests’, in this phase you verify that you have the optimal system, operating system and SQL Server configuration which you then document and hold consistent for a series of tests. Pre-testing enables you to ensure that your configuration is suitable for testing and the time invested will generate valid results. Pre-testing also enables you to gain familiarity with the HammerDB driver script settings.

For DSS testing a key focus is determining the optimal Maximum Degree of Parallelism (MAXDOP) and if being used columnstore index configuration. Once you are satisfied with your testing configuration you should then thoroughly plan your measured tests to ensure that all of your tests and results are fully documented.

To begin pre-testing if you have not already done so read the Introduction to Decision Support, Data Warehousing, Business Intelligence, and Analytical Load Testing for all Databases and therefore familiar with the Power Test, Throughput Test and Refresh Function. You should also have
downloaded the [HammerDB TPC-H Calculator](#) to enable automatic calculation of the results of your tests. Your first pre-test should be based upon the Power test. From the Options menu or treeview select the Driver Options as shown in Figure 22.

![Figure 22 Driver Options.](image)

At this stage your focus is now on completing the settings under Driver Options as shown in Figure 23.
Driver Options

Under the Driver Options section you have SQL Server, SQL Server Port, SQL server ODBC Driver, Authentication, SQL Server User ID, SQL Server User Password and SQL Server Database all identical to the options set for the schema build used to define your connections to the database.

Scale Factor

Although not visible under the Driver Options the Scale Factor value is also inherited from the Build Options and must be the set to the same value for running the Driver Script as was used for the Build. This is especially important if you have restarted HammerDB as you may need to set the Scale Factor in the Build Options again or manually in the script for correct functionality or set the value you are using in the config.xml file.

Under the Driver Options section you have the following choices:

MAXDOP

Within the driver options MAXDOP sets the value to be used for query hints to set the Maximum Degree of Parallelism that a particular query will use.

Total Query Sets per User

As you have already seen from the introduction document a Query Set is a sequence of 22 queries. The Total number of query sets is the number of times after logging on that the virtual user completes an entire sequence of queries before logging off again. The difference between this and using iterations value in the Virtual User options is that the virtual user only logs on and off once and completes all of the query sets in between whereas with the iterations value the entire script is run multiple times.
Exit on SQL Server Error

Exit on SQL Server Error is shown as the parameter RAISEERROR in the Driver Script. RAISEERROR impacts the behaviour of an individual virtual user on detecting an SQL Server error. If set to TRUE on detecting an SQL Server error the user will report the error into the HammerDB console and then terminate execution. If set to FALSE the virtual user will ignore the error and proceed with executing the next transaction. It is therefore important to be aware that if set to FALSE firstly if there has been a configuration error resulting in repeated errors then the workload might not be reported accurately and secondly you may not be aware of any occasional errors being reported as they are silently ignored. I recommend running pre-tests with RAISEERROR set to TRUE to ensure a configuration is valid before setting it to FALSE for a measured test run.

Verbose Output

Verbose Output is shown as VERBOSE in the Driver Script. Setting this value to TRUE will print both the Queries and their results for each virtual user however will add to the Query time by the time required to print the results.

Refresh Function

The refresh function checkbox corresponds to refresh on in the Driver Script. When this checkbox is enabled the first virtual user will run the refresh function as opposed to running a query set. Note that if you choose only one virtual user and select the refresh function checkbox then your virtual user will run a power test as detailed further in this document. The refresh function as the name implies inserts and deletes rows from the ORDERS and LINEITEM tables and the times of this function are required as input to calculating the QphH.

Number of Update Sets/Trickle Refresh Delay(ms)/Refresh Verbose

If you have enabled the refresh function then the values for Number of Update Sets/Trickle Refresh Delay(ms)/Refresh Verbose become active and these correspond to update_sets trickle_refresh and REFRESH_VERBOSE in the driver script respectively. The update sets determines how many times the virtual users will cycle through the refresh functions whilst noting that the function always starts at 1 and therefore cannot be restarted against the same schema until the schema has been refreshed. The Trickle Refresh Delay value sets the delay between each insert and delete with a default of 1 second ensuring that the refresh function does not place a significant load on the system, The Refresh Verbose value means that the virtual user running the refresh function reports on its activities.

When you have completed defining the Schema Options click OK to save your values. As noted previously you can also enter these values into the config.xml file to save a permanent record of your values for pre-populating the values after restarting HammerDB.

Loading the Driver Script

Once you have selected and saved your driver options under the treeview and Driver Script select Load as shown in Figure 24.
Figure 24 Select Driver Script

This will populate the Script Editor window with the driver script shown in Figure 25.
These scripts provide the workload on the SUT Database Server. If you have correctly configured the parameters in the Driver Options section you do not have to edit in the script. If you so choose however you may also manually edit the values given in the EDITABLE OPTIONS section paying close attention to the scale factor that must match the scale factor used in the schema build. Additionally the driver scripts are regular HammerDB scripts and a copy may be saved externally and modified as you desire for a genuinely Open Source approach to load testing.

**Pre-Test 1 Running a Power Test Query**

In this example we will create a single virtual users and choose to display their output to verify the schema and database configuration. To do this Under the Virtual Users menu selected from the Options menu or treeview as shown in Figure 26 select Virtual user Options and enter the number 1 for the Virtual Users. Also check the Show Output button to see what your users are doing whilst the test is running. Note that in contrast to the OLTP tests displaying the output will not reduce the overall level of performance as each query will be long-running and take significantly more time on the database server than is used by the client. The additional VERBOSE option within the script is available to provide a further level of detail, when complete click OK.
There are three other related options under the Virtual User Options dialogue, namely User Delay(ms), Repeat Delay(ms) and Iterations. Iterations defines the number of times that HammerDB should execute a script in its entirety. With regards to running the TPC-H driver script this can be thought of as the number of times a Virtual User logs on to the database, runs the defined number of query sets and logs off again. Consequently you already have an option for similar functionality within the TPC-H driver script itself to define the number of query sets and should therefore should use this instead. User Delay(ms) defines the time to wait between each Virtual User starting its test and the Repeat Delay(ms) is the time that each Virtual User will wait before running its next Iteration. For the TPC-H driver script the recommended approach is to leave the Iterations and User and Repeat Delays at the default settings. You also have the option to log the output, if you wish to generate a performance profile with the breakdown of the timing of each query you should also select this option. When you have completed the selection press OK. Click the Create Virtual Users button as shown in Figure 27 to create the Virtual User, it will be created but not start running yet.
You can observe as shown in Figure 28 that the virtual user has been created but is showing a status of idle. You can destroy the Virtual User by pressing the Red Traffic light icon that has appeared in place of the Create Virtual Users button. To begin running the power test queries press the button Run Virtual Users as shown in Figure 28, the name of the button will appear in the information pane.
Figure 28 Virtual Users Created

You can observe the Virtual User icon change to signify activity. The Virtual User has logged on to the database and is running queries as can be observed in the Virtual User Output as shown in Figure 29. Note that the Queries do not run in sequence order from Query 1 to Query 22 and instead run according to a pre-defined random order depending on the virtual user number that you can observe defined within the script.
When the Virtual User has completed all of its designated query set it will exit showing a positive status as shown in Figure 30. Once the Virtual User is displaying this positive status it has logged off the database. The Virtual User is once again idle and not running transactions. The Virtual User does not need to be destroyed and recreated to re-run the test from this status. The Virtual User can be destroyed to stop a running test, however will exit only after the current query has finished executing.
If there is an error when running the Driver Script it will be reported in the Virtual User icon with the detail of the error shown in the Console window.

At this stage in pre-testing the test configuration has been verified and it has been demonstrated that the load generation server can log on to the SUT Database Server and run a test.

**Pre-Test 2 Optimal DOP and In-memory for Power and Throughput Tests**

Once the configuration has been verified the next stage is to focus upon performance. The best place to start with verifying performance is to determine optimal DOP and In-memory settings. To do this follow all of the steps for Pre-Test 1 and focus on modifying the value of the parameter maxdop as shown in Figure 31 before each run (Also make note of the system MAXDOP parameters as you may look to also modify these during pre-testing).
Additionally note that if you have enabled In-memory functionality this will also have a significant impact on performance testing and note the time for the query set to complete and then repeat the test with a DOP of 2 and so on. Plotting the time taken for a query set in a graph will produce a result such as that shown in Figure 32.
Figure 32 Optimal MAXDOP

Note that although you have found the optimal DOP for a single user power test you should be aware that to produce this result may require significantly more resources than that achieved by a lower DOP with a slightly longer completion time. Therefore you should also calculate the DOP for the number of virtual users required for the throughput test at your chosen scale factor size by noting the maximum completion time of any of the multiple virtual users. For example Figure 33 shows the maximum completion time for the required 2 virtual users on a SF1 schema to be the 29 seconds of virtual user one.

Figure 33 Throughput DOP

To observe performance during the test you can use the Transaction Counter in exactly the same way as you use the transaction counter for OLTP tests. The Transaction Counter options be selected from the Options menu or the treeview and this displays the Transaction Counter Options as shown in Figure 34.
**Transaction Counter Options**

Under the Transaction Counter Options section you have the following choices:

**SQL Server/SQL Server Port/SQL Server ODBC Driver/Authentication/SQL Server User ID/SQL Server User Password**

The Connection details must be for a User with permission to select from `sys.dm_os_performance_counters`, you can validate by logging on with this user and running the following command.

```
C:\Users>sqlcmd -S (local)\SQLEXPRESS -E -Q "select cntr_value from sys.dm_os_performance_counters where counter_name = 'Batch Requests/sec'"
```

<table>
<thead>
<tr>
<th>cntr_value</th>
</tr>
</thead>
<tbody>
<tr>
<td>4490932</td>
</tr>
</tbody>
</table>

(1 rows affected)

This value is the same as shown in the Management Studio Activity Monitor.

**Refresh Rate**

The refresh rate defines the time in seconds between when the transaction counter will refresh its values. Setting this value too low may impact the accuracy of the data reported by the Microsoft SQL Server database and the default value of 10 seconds is a good choice for an accurate representation.

**Autorange Data Points**

By default the Data Points in the transaction counter will be anchored to the data point Zero. By selecting Autorange data points you enable the transaction counter to zoom in to show a finer detail of peaks and troughs in your transaction Data.

When you have completed the transaction counter options press OK to save your values and press the Transaction Counter button as shown in Figure 35 to begin observing the transaction rate.
Figure 35 Waiting for Data

After the first refresh time interval you will be able to observe the transaction counter updating according to the throughput of your system. The actual throughput you observe for a single Virtual User will vary according to the capabilities of your system as shown in Figure 36.
The main distinction between the transaction counter measurements for an OLTP test and a DSS test is that in this case the transaction counter will show the value for queries per hour. Note that this is not the same metric used to calculate the QphH value and as such is solely an indication of the query throughput of your system based on the number of SELECT calls processed. For larger schema sizes or smaller systems without significant resources you may find that the Query transaction counter does not register significant query throughput as the majority of the period of time is spent waiting for queries to be processed, this is normal and to be expected, additionally you should also not expect a flat transaction counter profile as would be expected from an OLTP workload as query times are significantly longer than transaction times.

**Running the DSS Tests**

To compile data similar to that collected for a TPC-H workload you need to run a Power Test including refresh functions following by a throughput test. Note that as previously observed if you are using Nonclustered columnstore indexes you will not be able to run the refresh tests as part of the power or throughput tests.

**Power Test**

A component of the TPC-H test is the refresh function and to adhere as closely as possible to a TPC-H test the refresh function should be run either side of the Power Test. To enable this functionality HammerDB has a special power test mode, whereby if refresh_on is set to true as shown in Figure 37 and only one
virtual user is configured then HammerDB will run a Power Test. Note that once you selected refresh_on for a single Virtual User in Power Test Mode the value of update_sets will be set to 1 and the value of trickle_refresh set to 0 and the value of REFRESH_VERBOSE set to false, all these values will be set automatically to ensure optimal running of the Power Test.

Figure 37 Enabling the Refresh Function

It is important to reiterate before running a Power Test that as shown in Figure 38 you cannot re-run the same refresh functions against the same schema due to constraint violations and therefore need to refresh the schema each and every time you run a refresh function. For this reason you should ensure that you have already followed the guidance previously given in this document to restore your schema to the original data.
Figure 38 Refresh function Constraint Violation

Create a single Virtual User as shown in Figure 39 and ensure that you enable logging to capture all of the query times.

Figure 39 Virtual User with Logging

Create a single Virtual User as you did previously, you should see and accept a prompt that logging has been activated as shown in Figure 40.
Run the Virtual User as previous, you should now see that the Virtual User performs the New Sales Refresh a single Query Set and the Old Sales Refresh function which comprises a complete Power Test.

Fetch the data from the HammerDB log as shown below you should see the data for both the refresh functions as well as all of the query times that can be used to graph a performance profile.
Vuser 1: New Sales refresh
Vuser 1: New Sales refresh complete in 7.463 seconds
Vuser 1: Completed 1 update set(s)
Vuser 1: Executing Query 14 (1 of 22)
Vuser 1: query 14 completed in 0.476 seconds
Vuser 1: Executing Query 2 (2 of 22)
Vuser 1: query 2 completed in 0.453 seconds
Vuser 1: Executing Query 9 (3 of 22)
Vuser 1: query 9 completed in 3.446 seconds
Vuser 1: Executing Query 20 (4 of 22)
Vuser 1: query 20 completed in 0.274 seconds
Vuser 1: Executing Query 6 (5 of 22)
Vuser 1: query 6 completed in 0.64 seconds
Vuser 1: Executing Query 17 (6 of 22)
Vuser 1: query 17 completed in 0.459 seconds
Vuser 1: Executing Query 18 (7 of 22)
Vuser 1: query 18 completed in 2.167 seconds
Vuser 1: Executing Query 8 (8 of 22)
Vuser 1: query 8 completed in 0.886 seconds
Vuser 1: Executing Query 21 (9 of 22)
Vuser 1: query 21 completed in 3.271 seconds
Vuser 1: Executing Query 13 (10 of 22)
Vuser 1: query 13 completed in 6.002 seconds
Vuser 1: Executing Query 3 (11 of 22)
Vuser 1: query 3 completed in 0.887 seconds
Vuser 1: Executing Query 22 (12 of 22)
Vuser 1: query 22 completed in 0.584 seconds
Vuser 1: Executing Query 16 (13 of 22)
Vuser 1: query 16 completed in 0.61 seconds
Vuser 1: Executing Query 4 (14 of 22)
Vuser 1: query 4 completed in 1.169 seconds
Vuser 1: Executing Query 11 (15 of 22)
Vuser 1: query 11 completed in 0.351 seconds
Vuser 1: Executing Query 15 (16 of 22)
Vuser 1: query 15 completed in 0.202 seconds
Vuser 1: Executing Query 1 (17 of 22)
Vuser 1: query 1 completed in 3.782 seconds
Vuser 1: Executing Query 10 (18 of 22)
Vuser 1: query 10 completed in 0.759 seconds
Vuser 1: Executing Query 19 (19 of 22)
Vuser 1: query 19 completed in 2.1 seconds
Vuser 1: Executing Query 5 (20 of 22)
Vuser 1: query 5 completed in 0.996 seconds
Vuser 1: Executing Query 7 (21 of 22)
Vuser 1: query 7 completed in 0.714 seconds
Vuser 1: Executing Query 12 (22 of 22)
Vuser 1: query 12 completed in 0.523 seconds
Vuser 1: Completed 1 query set(s) in 31 seconds
Vuser 1: Old Sales refresh
Vuser 1: Old Sales refresh complete in 1.072 seconds
Vuser 1: Completed 1 update set(s)

**Throughput Test**

After the power test you should run the throughput test (if the refresh function has been run it is necessary to refresh the schema). For the throughput test you need to also run the refresh function however this time the aim is to trickle the refresh function slowly while multiple query streams are run. To find the correct number of query streams refer to the HammerDB DSS introduction document. Configure the options as shown in Figure 42 and therefore enable the refresh function, this time the update sets, trickle refresh and REFRESH_VERBOSE options will also be enabled when refresh_on is set to true.
Figure 42 Running the Throughput Test with Refresh

Configure the correct number of Virtual Users to enable the first Virtual User to run the Refresh Functions and additional Virtual Users to run the Query Streams as defined in the specification for the test. Note that before running a long running query at the same time as the inserts of the refresh function you should enable snapshot isolation on the database as shown in Figure 43. Failure to do so will mean the Query streams will hang under a shared lock (LCK_M_S) whilst the refresh function is running.
Figure 43 Snapshot Isolation On

Run the Virtual Users as shown in Figure 43, note that the Refresh Function will run more slowly as expected.

Figure 44 Running the Throughput Test with Refresh
When the Virtual Users running the Query sets have completed the throughput tests, note the longest (not the shortest) time taken for a full query set to complete. As shown in Figure 45 this value is 30 seconds. You do not need to wait for the trickled refresh function to complete, however must have configured enough update sets to ensure that the refresh function remains running whilst the throughput test completes.

![Image of HammerDB interface with throughput test results]

Figure 45 Completed Throughput Test with Refresh

**Calculating QphH**

It is important to reiterate that HammerDB results are not official TPC-H results and cannot be compared with official HammerDB results in any way. Nevertheless you can gain a close approximation to official TPC-H data by entering the results of your Power and Throughput tests into the [HammerDB TPC-H Calculator](#) where the text is marked red as shown in Figure 46.
Coupled with performance profiles of query times this data provides the optimal way to test SQL Server Query performance, parallel query and in-memory column store features.

**Support and Questions**

For help use the HammerDB Sourceforge forum available at the HammerDB sourceforge project.