Microsoft SQL Server OLTP (Transactional) Load Testing

The document Introduction to Transactional (OLTP) Load Testing for all Databases provides a general overview on the HammerDB OLTP workload and should be read prior to this database specific guide. This guide gives you an introduction to conducting OLTP (Online Transaction Processing) workloads on the Microsoft SQL Server Database. This will equip you with the essentials for assessing the ability of any system that runs the Microsoft SQL Server Database for processing transactional workloads. On completion of this guide you will be able to run detailed and comprehensive Microsoft SQL Server load tests. After building a basic skill set, you should be able to take a system from ‘bare metal’ to generation of a full performance profile within one day.

Database load testing is an advanced skill and therefore familiarity with the Microsoft SQL Server Database and basic Microsoft SQL Server DBA skills are assumed. You should already be able to install, create, administer and connect to a Microsoft SQL Server database. If you do not have these skills I recommend start with an Introduction to Microsoft SQL Server.

Test Network Configuration ................................................................. 1
Load Generation Server Configuration ............................................. 2
SUT Database Server Configuration .................................................. 2
Administrator PC Configuration ....................................................... 2
Installation and Configuration ......................................................... 2
Load Generation Server Installation .................................................. 3
Load Generation Server Configuration ............................................. 3
Authentication and ODBC Driver ..................................................... 3
SUT Database Server Installation ..................................................... 4
Creating the Test Schema ................................................................. 4
Build Options ................................................................................. 6
Starting the Schema Build ............................................................... 7
Pre-Testing and Planning ................................................................. 12
Driver Options .............................................................................. 14
Loading the Driver Script ............................................................... 17
Pre-Test 1 Verifying the Schema ..................................................... 18
Pre-Test 2 Single and Multiple Virtual User Throughput ..................... 25
Planning and Preparation ................................................................. 32
Running Timed Tests with the Timed Test Driver Script ....................... 33
Automating Tests with Autopilot Mode ............................................ 38
Best Practices .............................................................................. 42
Support and Questions .................................................................. 42

Test Network Configuration

You require the database server to be tested known as the system under test (SUT) installed and configured with the Microsoft SQL Server database. You also require a load generation server to run HammerDB.
installed with the HammerDB software and a Microsoft SQL Server client. Typically the load generation server is run on a separate system from the SUT with the load generated across the network. It is possible to run HammerDB on the same system as the SUT however this will be expected to produce different results from a network based load. Both the SUT and the load generation server may be virtualized or container databases although similarly results may differ from a native hardware based installation.

Load Generation Server Configuration

The most important component of the load generation server is the server processor. The overall load generation server capacity required depends on the system capabilities of the SUT. It is recommend to use an up to date multicore processor. HammerDB is a multithreaded application and implicitly benefits from a multicore server CPU. To determine whether CPU capacity is sufficient for testing you can monitor the CPU utilisation with HammerDB Metrics. CPU utilisation reaching 100% is an indication that the CPU on the load generation server is limiting performance. Load generation memory requirements are dependent on the operating system configuration and the number of virtual users created with each virtual user requiring its own sql server client. Typically server sizing guidelines should be within the limits expected to support a real user count. Multiple load generation servers connected in a “master-slave” configuration are enabled within HammerDB to exceed the capacity of a single load generation client. HammerDB consumes 15MB of disk space and you will also need to have installed a compatible sql server client. All sql server database installations include a sql server client. The load generation server does not need to be running the same version of SQL Server as the SUT.

SUT Database Server Configuration

The database server architecture to be tested must meet the standard requirements for a Microsoft SQL Server 2008 Database Server. Microsoft SQL Server can be installed on any supported Windows operating system, however the minimum version of Microsoft SQL Server that is required is 2008, no version of SQL Server prior to 2008 has been tested. To run a HammerDB transactional load test there are minimum requirements in memory and I/O (disk performance) to prevent these components being a bottleneck on performance. For a configuration requiring the minimal level of memory and I/O to maximize CPU utilization keying and thinking time should be set to FALSE (keying and thinking time is detailed later in this guide). To achieve this you should aim to create a schema with approximately 200-250 warehouses per CPU socket so for example 400-500 warehouses for a 2 socket system and 800-1000 for a 4 socket system. As long as it is not too small resulting in contention the schema size should not significantly impact results. You should have sufficient memory to cache as much of your test schema in memory as possible. If keying and thinking time is set to TRUE you will need a significantly larger schema and number of virtual users to create a meaningful system load. Reductions in memory will place more emphasis on the I/O performance of the database containing the schema. If the allocated SQL Server memory is sufficient most of the data will be cached during an OLTP test and I/O to the data area will be minimal. As a consequence the key I/O dependency will be to the transaction logs for both bandwidth and sequential write latency. Modern PCIe SSDs when correctly configured have been shown to provide the capabilities to sustain high performance transaction logging.

Administrator PC Configuration

The administrator PC has the minimal requirement to display the graphical output from the load generation server. The PC should also have the ability to connect to the SUT to monitor performance by the installation of a Microsoft SQL Server client.

Installation and Configuration

This sections describes the procedure to install and configure the Load Generation Server and the SUT
Database Server.

Load Generation Server Installation

On the Load Generation Server refer to the dedicated HammerDB Installation Guide.

Load Generation Server Configuration

All of HammerDB’s working data can be set using menu options. However if you wish in the HammerDB home directory there is a configuration file called config.xml that is read on startup. In this file you can preset your schema build and driver configurations by editing the xml file without having to change the data manually. If your xml file is well formed your variables will be applied to HammerDB when you selected the menu options. In particular you will want to change the value of rdbms to MSSQLServer in order that SQL Server is always preselected on start.

<?xml version="1.0" encoding="utf-8"?>
<hammerdb>
  <rdbms>MSSQLServer</rdbms>
  <bm>TPC-C</bm>
  ...
  <mssqlserver>
    <connection>
      <mssqls_server>{local}</mssqls_server>
      <mssqls_port>1433</mssqls_port>
      <mssqls_authentication>windows</mssqls_authentication>
      <mssqls_odbc_driver>SQL Server Native Client 11.0</mssqls_odbc_driver>
      <mssqls_uid>sa</mssqls_uid>
      <mssqls_pass>admin</mssqls_pass>
    </connection>
    <tpcc>
      <schema>
        <mssqls_count_ware>1</mssqls_count_ware>
        <mssqls_schema>original</mssqls_schema>
        <mssqls_num_threads>1</mssqls_num_threads>
        <mssqls_dbase>tpcc</mssqls_dbase>
      </schema>
      <driver>
        <mssqls_total_iterations>1000000</mssqls_total_iterations>
        <mssqls_raiseerror>false</mssqls.raiseerror>
        <mssqls_keyandthink>false</mssqls_keyandthink>
        <mssqls_checkpoint>false</mssqls_checkpoint>
        <mssqlsdriver>standard</mssqlsdriver>
        <mssqls_rampup>2</mssqls_rampup>
        <mssqls_duration>5</mssqls_duration>
      </driver>
    </tpcc>
    ...
  </mssqlserver>
  ...

Authentication and 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 1.
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 2:

Figure 2 ODBC Data Driver

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 ”}.

SUT Database Server Installation

Installation and configuration of the Microsoft SQL Server Database on your chosen operating system is beyond the scope of this document. You should have the Microsoft SQL Server database software installed and running. For tuning and configuration review the best practices guide.

Creating the Test Schema

To create the OLTP test schema based on the TPC-C specification you will need to select which benchmark and database you wish to use by either double-clicking on SQL Server under the Benchmark menu or choosing Benchmark from under the Options menu. The initial settings are determined by the values in
your config.xml file or the values selected under the treeview. Select Microsoft SQL Server and TPC-C and press OK as shown in Figure 5, SQL Server will move to the top of the treeview.

![Benchmark Options](image)

**Figure 3 Select Benchmark**

To create the TPC-C schema select the TPC-C schema build options from the treeview. This view will change dynamically according to your chosen database.

![Schema Build Options](image)

**Figure 4 Schema Build Options**

If selected from the Options 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. If selected from the benchmark treeview
only the “Build Options” are shown and these are the only options of importance at this stage. Note that in any circumstance you don’t have to rebuild the schema every time you change the “Driver Options”, once the schema has been built only the “Driver Options” may need to be modified. For the “Build Options” fill in the values according to the database where the schema will be built.

![Database Build Options](image)

**Figure 5 Database Build 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**

The Microsoft SQL ODBC Driver is the ODBC driver you will use to connect to the SQL Server database. To view which drivers are available on Windows view the ODBC Data Source Administrator, for example “SQL Server” and “SQL Server Native Client 10.0” may be listed and either specified for a default ODBC connection.

**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 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.

**Schema**

There are 2 schemas available, the original HammerDB SQL Server schema or an updated schema. The original schema should be used for backward compatibility however the updated schema offers enhanced performance and should be used where there is no requirement for results to be compared with historical tests.

**Number of Warehouses**

The Number of Warehouses is selected by a slider. For fine-tuning you may click either side of the slider to move the value by 1. You should set this value to number of warehouses you have chosen for your test based on the guidance given previously in the section SUT Database Server Configuration.

**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 either the number of warehouses you are going to create (You cannot set the number of threads lower than the number of warehouses value) or the number of cores/Hyper-Threads on your Load Generation Server.

**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_count_ware>1</mssqls_count_ware>
  <mssqls_schema>original</mssqls_schema>
  <mssqls_num_threads>1</mssqls_num_threads>
  <mssqls_dbase>tpcc</mssqls_dbase>
</schema>
```

To begin the schema creation at the buttons in the top level window click the "Create TPC Schema" button. This button is shown as three coloured boxes either in the treeview or as a button and "Create TPC Schema"
appears in the information box when moused over as shown in Figure 6.

![Figure 6 Create Schema](image)

**Figure 6 Create Schema**

On clicking this button a dialogue box such as the one shown in Figure 7 appears.

![Figure 7 Confirm Schema](image)

**Figure 7 Confirm Schema**

When you click Yes HammerDB will login to your chosen Microsoft SQL Server host with a monitor thread as your defined user with the password you have chosen. It will then create the database you have defined and then load the item 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 8. 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 network performance.
Figure 8 Schema Building

When the workers are complete the monitor thread will create the indexes, stored procedures and gather the statistics. When complete Virtual User 1 will display the message TPCC SCHEMA COMPLETE and all virtual users will show that they completed their action successfully as shown in figure 9.
Figure 9 Schema Build Complete

Press the button to destroy the virtual users as shown in figure 9 and clear the script editor as shown in figure 10.
The TPC-C 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 the 1-5000 warehouses available in the GUI 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 number you like to create more warehouses, for example the following will create 10000 warehouses.

do_tpcc {(local)\SQLEXPRESS} 1433 {SQL Server Native Client 11.0}
windows sa admin 10000 tpcc updated 8

Similarly change any other value to modify your script. If you have made a mistake simply close the application and in SQL Server Management Studio right-click the database and choose Delete. Select the Close existing connections checkbox and click OK.

When you have created your schema you can verify the contents with the SQL Server Management Studio or SQL Connection, for example:

C:\Users>sqlcmd -S (local)\SQLEXPRESS -E -Q "use tpcc; select name from sys.tables"

Changed database context to 'tpcc'.

name
----------------------------------------
CUSTOMER
DISTRICT
HISTORY
You can also browse the stored procedures you have created by looking in the creation script. At this point the data creation is complete and you are ready to start running a performance test. Before doing so it is worth noting that the schema has been designed in order that you can run multiple tests and it will return the same results. You therefore do not need to recreate your schema after every run for consistent results. Conversely if you do wish to recreate your schema for such a reason as you have exhausted your available disk space the results of tests against different sizes are comparable.

---

**Pre-Testing and Planning**

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 Oracle 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 and finally to ‘warm the cache’ by having your schema data cached in the buffer cache before beginning an extended sequence of tests. 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 select the TPC-C schema options menu tab from the top level Benchmark menu or the Driver Script Options from the treeview as shown in Figure 11.
At this stage your focus is now on the values for the driver script as shown in Figure 12 whilst noting that the connection values are the same ones that you successfully used to create your schema.
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.

TPC-C Driver Script

Under TPC-C Driver script you have the option of choosing either the Standard Driver Script or the Timed Test Driver Script. This choice will dynamically change the Driver Script that is loaded when the TPC-C Driver Script menu option is chosen. The Standard Driver Script as shown in figure 13 is a script run by all virtual users. This script should be chosen where you wish to create a load against the database and view the transaction rate but do not wish to run a timed test or wish to time the tests manually yourself. The Standard Driver Script may be run with Virtual User Output turned on, which will display all of the information each virtual user processes or with Virtual User Output turned off to be able to observe the transaction rate only. Your additional Driver Options choices are populated in the EDITABLE OPTIONS section.
Instead of the Standard Driver Script you can select the Timed Test Driver Script. As shown in Figure 14 this produces a number of additional options. You should select the Timed Test Driver Script when you wish to run timed tests and have HammerDB time these tests, measure the results and report on an average transaction rate for a period of time. With the Timed Test Driver Script the first virtual user will do the timing and generate the results with the additional virtual users running the workload, therefore you should always select the number of desired virtual users + 1 when running the Timed Test Driver Script. For example if you wish to measure a load generated by two virtual users you should select three virtual users before running the script. Additionally the Timed Test Driver Script is designed to be run with Virtual User Output enabled, this ensures that the information gathered by the first virtual user on the transaction rates are correctly reported. Whilst running the Timed Test Driver Script virtual user output for the virtual users generating the load is suppressed.

For both the Standard Driver Script and Timed Test Driver Script the further options selected within the Schema Options window are entered automatically into the EDITABLE OPTIONS section of the driver script as follows:

**Total Transactions per User**

Total transactions per user is reported as total_iterations within the EDITABLE OPTIONS section of the driver script.
script. This value will set the number of transactions each virtual user will process before logging off. You can use this value to determine how long the virtual user will remain active for. The length of time for activity will depend upon the performance of the Database Server under test. A higher performing server will process the defined number of transactions more quickly than a lower performing one.

It is important to draw the distinction between the total_iterations value and the Iterations value set in the Virtual User Options window. The Iterations value in the Virtual User Options window determines the number of times that a script will be run in its entirety. The total_iterations value is internal to the TPC-C driver script and determines the number of times the internal loop is iterated:

for {set it 0} {$it < $total_iterations} {incr it} { ... }

In other words if total_iterations is set to 1000 then the executing user will log on once execute 1000 transactions and then log off. If on the other hand Iterations in the Virtual User Options window is set to 1000 and total_iterations in the script set to 1 then the executing user will log on execute one transaction and then log off 1000 times. For the TPC-C driver script I recommend only modifying the total_iterations value.

When running the AWR Snapshot Driver Script as the test is timed you should ensure that the number of transactions is set to a suitably high value to ensure that the virtual users do not complete their tests before the timed test is complete, doing so will mean the you will be timing idle virtual users and the results will be invalid. Consequently it is acceptable when running timed tests to set the Total Transactions per User to a high value such as 1000000 (the default value) or more to ensure that the virtual users continue running for a long period of time. When the test is complete you can stop the test running by stopping the virtual users.

**Exit on SQL Server Error**

Exit on Microsoft SQL Server error is shown as the parameter RAISEERROR in the Driver Script. RAISEERROR impacts the behaviour of an individual virtual user on detecting a Microsoft SQL Server error. If set to TRUE on detecting a Microsoft 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.

**Keying and Thinking Time**

Keying and Thinking Time is shown as KEYANDTHINK in the Driver Script. A good introduction to the importance of keying and thinking time is to read the TPC-C specification. This parameter will have the biggest impact on the type of workload that your test will take.

---

**TIP:** The most common configuration error is to run a test with Keying and Thinking Time set to False with too many virtual users for the schema created. One virtual user without keying and thinking time will generate a workload equivalent to many thousands of users with keying and thinking time enabled.

---

Keying and thinking time is an integral part of an official TPC-C test in order to simulate the effect of the workload being run by a real user who takes time to key in an actual order and think about the output. If KEYANDTHINK is set to TRUE each user will simulate this real user type workload. An official TPC-C benchmark implements 10 users per warehouse all simulating this real user experience and it should
therefore be clear that the main impact of KEYANDTHINK being set to TRUE is that you will need a significant number of warehouses and users in order to generate a meaningful workload and hence an extensive testing infrastructure. The positive side is that when testing hundreds or thousands of virtual users you will be testing a workload scenario that will be closer to a real production environment. Whereas with KEYANDTHINK set to TRUE each user will execute maybe 2 or 3 transactions a minute you should not underestimate the radical difference that setting KEYANDTHINK to FALSE will have on your workload. Instead of 2 or 3 transactions each user will now execute tens of thousands of transactions a minute. Clearly KEYANDTHINK will have a big impact on the number of virtual users and warehouses you will need to configure to run an accurate workload, if this parameter is set to TRUE you will need at least hundreds or thousands of virtual users and warehouses, if FALSE then you will need to begin testing with 1 or 2 threads, building from here up to a maximum workload with the number of warehouses set to a level where the users are not contending for the same data. A common error is to set KEYANDTHINK to FALSE and then create hundreds of users for an initial test, this form of testing will only exhibit contention for data between users and nothing about the potential of the system. If you do not have an extensive testing infrastructure and a large number of warehouses configured then I recommend setting KEYANDTHINK to FALSE (whilst remembering that you are not simulating a real TPC-C type test) and beginning your testing with 1 virtual user building up the number of virtual users for each subsequent test in order to plot a transaction profile.

**Minutes of Rampup Time**

The Minutes of Rampup Time is shown as rampup in the Driver Script. The rampup time defines the time in minutes for the monitoring virtual user to wait for the virtual users running the workload to connect to the database and build up the transaction rate by caching data in the database buffer cache before taking the first timed value and timing the test. The rampup time should be sufficiently long enough for a workload to reach a steady transaction rate before the first timed value is taken.

**Minutes for Test Duration**

The Minutes for Test Duration is shown as duration in the Driver Script. The test duration defines the time of the test measured as the time the monitor thread waits after the first timed value before taking the second one to signal the test is complete and the active virtual users to complete their workload.

**Mode Options**

The mode value is taken from the operational mode setting set under the Mode Options menu tab under the Mode menu. If set to Local or Master then the monitor thread takes the timed values, if set to Slave no timed values are taken. This is useful if multiple instances of HammerDB are running in Master and Slave mode to ensure that only one instance takes the timed values.

**Loading the Driver Script**

Once you have selected and saved your driver options under the Benchmark Menu select TPC-C and TPC-C Driver Script or select Load from the treeview as shown in Figure 15.
This will populate the Script Editor window with the driver script shown in Figure 13 or 14 according to whether the standard or timed test driver script is chosen. These scripts provide the interaction from the Load Generation Server to the schema 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 the values given in the EDITABLE OPTIONS section. 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 Verifying the Schema

Figure 16 shows a successfully loaded Standard Driver Script which provides a useful first test against a newly created TPC-C Schema.
In this example we will create two virtual users and choose to display their output to verify the schema and database configuration. To do this Under the Options menu or from the treeview as shown in Figure 17 select the Virtual User Options and enter the number 2. Also check the Show Output button to see what your users are doing whilst the test is running. Note that displaying the output will reduce the overall level of performance (although HammerDB is multi-threaded many Window display systems are not and a display can only be updated by a single thread thereby limited performance) and click OK. Showing output is OK here as it is running a pre-test and not a performance test.
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-C driver script this can be thought of as the number of times a Virtual User logs on to the database, runs the number of transactions you defined in Total Transactions per User and logs off again. For example if Total Transactions per User was set to 1000 and the Virtual Users Iterations was set to 10, the Virtual User would complete 10000 transactions in total logging off and on between each run. Setting Total Transactions per User to 10000 and Virtual User Iterations to 1 would also complete 10,000 transactions per virtual user but all in one session. 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-C driver script the recommended approach is to leave the Iterations and User and Repeat Delays at the default settings and only modify the Total Transactions per User or total_iterations value inside the Driver Script. When you have completed the selection press OK. Click the Create Virtual Users button or the Create Virtual User treeview option as shown in Figure 18 to create the virtual users, they will be created but not start running yet.
Figure 18 Create Virtual Users

You can observe as shown in Figure 19 that the virtual users have been created but are showing a status of idle. You can destroy the Virtual Users by pressing the Red Traffic light icon that has appeared in place of the Create Virtual Users button. To begin the test press the button Run Virtual Users as shown in Figure 19, the name of the button will appear in the information pane.
You can observe the Virtual User icon change to signify activity. The Virtual Users have logged on to the database, you will be able to see their presence in the sysprocesses list for example:

```
C:\Users>sqlcmd -S (local)\SQLEXPRESS -E -Q "select dbid,program_name,status from master..sysprocesses where program_name like '%Tk%"
```

```
dbid  program_name
5 Tk 8.6 for Windows  status
5 Tk 8.6 for Windows  runnable
```

and are running transactions as can be observed in the Virtual User Output as shown in Figure 20.
When the Virtual Users have completed all of their designated transactions they will exit showing a positive status as shown in Figure 21. Once the Virtual User is displaying this positive status it has logged off the database and will not be seen in sysprocesses. 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 Users can be destroyed to stop a running test.
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. Figure 22 shows an example of an error, in this case it is a Microsoft SQL Server error illustrating an unknown database has been selected. 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.
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 Single and Multiple Virtual User Throughput**

Once the configuration has been verified the next stage is to focus upon performance. The best place to start with verifying performance is to monitor the workload of a single Virtual User. To do this follow all of the steps for Pre-Test 1 ensuring that you select the Standard Driver Script. Note that the Timed Test Driver Script is designed for multiple users with one Virtual User providing the monitoring capabilities for the other Virtual Users. Consequently if one Virtual User is configured to run the Timed Test Driver Script it will result in one Virtual monitoring an idle workload which is almost certainly not the desired outcome. Once the Standard Driver Script has been loaded configure a single Virtual User as shown in Figure 23. Configure One Virtual user without selecting the Show Output check box (The reason for suppressing output is described under Pre-Test 1).

Figure 22 Virtual User Error

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.
Note that a single Virtual User without output is the default configuration if you have not modified the config.xml file and therefore creating the Virtual Users will give you this single Virtual Configuration without specifically configuring the Virtual Users as shown in Figure 23. Figure 24 shows the single Virtual User created and the Standard Driver script loaded.

Press the Run Virtual Users button as described previously to begin generating the Single User Throughput test, the Virtual User icon will be updated to signify that the workload is running. To observe performance
during the test you can use the Transaction Counter. The Transaction Counter options can be selected from
the treeview or the options menu as shown in Figure 25.

Figure 25 TX Counter Options

This displays the Transaction Counter Options as shown in Figure 26.

Figure 26 Transaction Counter Options

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"  
cntr_value  
----------------------  
19  
(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 27 to begin observing the transaction rate.
The transaction Counter will become active and start collecting throughput data as shown in Figure 28.
Figure 28 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. Additionally once the transaction rate reaches a steady state you should observe the transaction counter maintaining a reasonably flat profile. Low transaction rates or excessive peaks and troughs in the transaction counter should be investigated for system bottlenecks on throughput.
Once you are satisfied with the single Virtual User throughput close both the Transaction Counter and destroy the Virtual Users also stopping the test by pressing both Red Traffic Light icons. You should also proceed to pre-testing the throughput multiple Virtual Users. To do so repeat the testing you have done for a single Virtual User however instead increase the value for the number of Virtual Users to run the test in the Virtual User Options as shown in Figure 30.
Similarly monitor the throughput for a higher number of Virtual Users as shown in Figure 31.

![HammerDB](image)

**Figure 31 Running Multiple Virtual Users**

**Planning and Preparation**

Planning and Preparation is one of the most crucial stages of successful testing but is often overlooked. Firstly you should fully document the configuration of your entire load testing environment including details such as hardware, operating system versions and settings and Oracle version and parameters. Once you have fully documented your configuration you should ensure that the configuration is not changed for an entire series of measured tests. This takes discipline but is an essential component of conducting accurate and measured tests. If you wish to change the configuration between tests to improve performance you should do so as part of the pre-test phase and not for the measured tests. If you change any aspect of the configuration you should conduct another full series of measured tests.

To plan your measured tests you should have a defined aim for what you wish to achieve and plan the tests accordingly. Often a test project can fail for having an unclear definition for the aim of what is desired to be achieved. Typically this aim will take the form of determining the performance characteristics of a server (or server) however this can have many forms, for example generating a performance profile, determining the maximum throughput, measuring transaction response times or determining the maximum number of supported virtual users. The tests will vary according to the aim, for example it is relatively meaningless to use a test without keying and thinking to determine the maximum number of supported virtual users (because each virtual user can use the maximum performance of one core or thread), similarly enabling keying and thinking time is not applicable to determining a performance profile. Alternative testing aims can be to compare multiple configurations on the same platform, for example looking at the impact on throughput of Virtualization, RAC or changing OS and Oracle parameters, the scope in this area for testing is
In this guide we will focus upon one of the most common testing scenarios, to generate a performance profile for server. This aim is used to identify for a given configuration of CPU, memory and I/O on a defined OS and Oracle configuration the maximum number of transactions that the system can support. This is tested for a given number of virtual users, starting with one virtual user scaling up to the maximum number that the system can support. This approach ensures that the full capabilities of a multithreaded server are tested. With this approach we will define our Virtual Users without keying and thinking time. The number of cores/threads in this example on the SUT Database Server is 16, therefore we will prepare a simple tracking spreadsheet to record the results of our tests as shown in Figure 32.

![Figure 32 Planning Spreadsheet](image)

With the configuration documented, the aim defined and a method to track the results of the tests prepared for our performance profile test project it is now possible to proceed to running timed tests with the Timed Test Driver Script.

---

**Running Timed Tests with the Timed Test Driver Script**

To run a timed and measured test there is an additional script to the Standard Driver Script called the Timed Test Driver Script that automates this functionality for you. To select the Timed Test driver script, open the TPC-C Driver Script Window as described previously in this guide. It is important to reiterate that you do not need to recreate the schema to modify the driver options or to change from using the Standard Driver Script to the Timed Test Driver Script or Vice Versa. Within the Driver Options shown in Figure 33, select the Timed Test Driver Script radio button.
Once the Timed Test Driver Script is selected this activates the options to select the Minutes of Rampup Time and Minutes for Test Duration as described previously in this guide. For a performance profile test you should plan to keep the Minutes of Rampup Time and the Minutes for Test Duration consistent for a number of tests with an increasing number of Virtual Users. For this reason you should plan to allocate sufficient rampup time for the higher number of Virtual Users at the end of your test sequence as well as the smaller number at the start. When you have selected your options click OK.

From under the Benchmark and TPC-C Menu select TPC-C Driver Script, this populates the Script Editor Window as shown in Figure 34 with the Timed Test Driver Script configured with your chosen options.

Figure 33 Timed Test Driver Options
To change these options you can either change them in the Schema Options window and reload the driver script or more advanced users can also change them directly in the Driver Script itself. To run the Timed Test Driver Script you must configure the Virtual Users as you did with the Standard Driver Script however there are two notable differences to observe. Firstly when running the Timed Test Driver Script one Virtual user will not run the Driver Script workload, instead this one Virtual User will monitor the timing of the test, measure the average transaction values and return the results. For this reason you should configure your Virtual Users with a Virtual User + 1 approach. ie to measure the workload for 1 Virtual User you should configure 2 Virtual Users, to measure the workload for 2 virtual Users you should configure 3 and so on. Additionally the Timed Test Driver Script is designed to be run with the Virtual User output enabled in order that you can view the Output from the Virtual User doing the monitoring, consequently the output for the Virtual Users running the workload is suppressed. The Virtual User configuration for the first test will look as Figure 35.
Click OK to save the configuration. Click the Create Virtual Users button as shown previously in this guide in Figure 18 and Start the Virtual Users running as shown in Figure 19. Note that the Virtual User output is now different as shown in Figure 36.

The output shows that rather than reporting the outcome of every transaction the worker Virtual User 2 reports that it is processing transactions, however the output is suppressed. The Virtual User will print its
message AFTER it has logged on and immediately BEFORE it runs its first transaction. If this message has not been printed the session is still in the process of logging into the database. You can check how this is proceeding by checking the process list as described previously to display the number of connections created. Increasing the UserDelay(ms) value in the virtual user options can on some systems prevent a "login storm" and have all users logged on and processing transactions more quickly (Note that the SQL Server best practices guide gives essential information on login times when running under a system with more than 64 CPUs and multiple processor groups). Your rampup time should allow enough time for all of the users to be fully connected. You will also be able to observe that in this example this single virtual User has logged on to the database and is running the workload. You can also observe that the monitor Virtual User, in this example Virtual User 1 is not running a workload but instead has logged on to measure the rampup time followed by taking the first transaction value, measuring the timed test, taking the second transaction value and reporting the outcome before logging off and ending the monitor script. It is worthwhile reiterating therefore that for the Timed Test Driver Script you need to configure and run n+1 Virtual Users with the additional Virtual User doing the monitoring and measuring. The sample output of this monitoring Virtual User is shown in Figure 37.

Figure 37 Timed Test Result
The monitoring user reports the TEST RESULT of TPM and NOPM. TPM measures the number of Microsoft SQL Server Transactions per minute and is not to be confused with the tpmC value from an official TPC-C benchmark. NOPM reports the number of New Orders per minute and is used as a database independent statistic. Consequently for example TPM cannot be used to compare the performance results of different databases but NOPM can. When you have stopped the test enter your data into your reporting spreadsheet. Once you are satisfied with the test results, repeat the test with the next value in the number of Virtual Users in your sequence remembering to add one for the monitor thread. Once this test is
complete either repeat the process with the next value in the sequence or automate your testing with autopilot mode as detailed in the following section. With either method do this until you have completed your spreadsheet with all of the desired values for database performance.

**Automating Tests with Autopilot Mode**

If you prefer to run all of your tests manually you do not need to use the Autopilot Mode. However if you wish to run your entire sequence of tests unattended then Autopilot Mode enables you to use your time most productively. It can help to understand Autopilot Mode as a feature that simulates the presence of a DBA instructed to run your desired sequence of tests at specified time intervals and report the entire results of all tests in one batch. To begin configuring Autopilot mode follow the steps described in the previous section for Running Timed Tests up to the steps illustrated in Figures 36 and 37. You only need to configure the correct driver script but not configure the Virtual Users, they will be configured automatically. To do this select Autopilot Options from the either the Options menu or the treeview as shown in Figure 38.

![Figure 38 Autopilot menu](image)

This shows the Autopilot Options menu as shown in Figure 39.
Configure the Autopilot options precisely in the same manner as you would use to instruct your Virtual DBA as follows:

**Autopilot Disabled/Autopilot Enabled**

This Autopilot Disabled/Autopilot Enabled Radio buttons give you the option to select whether the Autopilot button is enabled on the main window.

**Minutes per Test in Virtual User Sequence**

The minutes for test duration defines the time interval between which your virtual DBA will create the Virtual Users, stop the test and create the next Virtual Users in the sequence. You should configure this value in relation to the Minutes for Ramup Time and Minutes for Test Duration. For example if the values in the test script are 2 and 5 minutes respectively then 10 minutes for the Autopilot Options is a good value to allow the test to complete before the next test in the sequence is run. If however the test overruns the time interval and the Virtual Users are still running the sequence will wait for the Virtual Users to complete before proceeding.

**Virtual User Sequence (Space Separated Values)**

The Virtual User Sequence defines the number of Virtual Users to be configured in order for a sequence of tests separated by the Minutes for Test Duration. For example as shown in Figure 46, firstly a test with 2 Virtual Users will be run, then after 10 minutes a test with 3 Virtual Users will be run, then 5 Virtual Users and so on to the end of the sequence. Note that the default Values are given as odd numbers to account for the Monitoring Virtual User when running the Timed Test Driver Script. Therefore in this example the actual Users running the workload will be 1, 2, 4, 8, 12, 16, 20 and 24.

**Show Virtual User Output/Log Virtual User Output to Temp**

These values are exactly the same as set when defining the Virtual Users, the Autopilot Options gives you the opportunity to set them when configuring Autopilot Mode to ensure that you have a permanent record of the output of the tests that you run.

Once your Autopilot Options are defined, press OK to save the values. Close down all running virtual Users and the transaction counter and press the Autopilot button as shown in Figure 40.
You can now leave the autopilot mode to run your chosen sequence of tests without any further intervention. The Autopilot screen as shown in Figure 41 becomes active and reports your progress. In particular note the timer in the top right hand corner tracking the interval times at which your tests should be run. This interval must be long enough to allow for your ramp up time, test time and any post-test workload such as running a checkpoint. Note that if the autopilot interval is too short it will stop the test running during the timing interval and therefore no results will be reported. If you do not see results when running autopilot often the resolution is to increase the autopilot interval.
The Autopilot will continue to run through your chosen sequence, creating virtual users and running the test in the test script as shown in Figure 42.
When your tests has completed you may retrieve all of your results from the main autopilot window. You can collect the results for an entire sequence of tests into your spreadsheet without having run each test manually.

**Best Practices**

You should now be equipped with all of the performance data you need to generate performance profiles and begin your analysis as described [Introduction to Transactional (OLTP) Load Testing for all Databases](#). Advanced optimization and tuning for SQL Server is in separate [best practices](#) guide.

**Support and Questions**

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