hammerora

the open source oracle load test tool

MySQL OLTP (Transactional) Load Testing

This guide gives you an introduction to conducting OLTP (Online Transaction Processing) workloads on the MySQL Database and database server that offer drop-in replacement functionality for MySQL. This guide will equip you with the essentials for assessing the ability of any system that runs the MySQL Database for processing transactional workloads. On completion of this guide you will be able to run detailed and comprehensive MySQL 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. If you have not already done so you should read the Quick Start tutorial before proceeding with this guide.

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

Introduction

In this introduction we give an an overview of the correct approach to take for MySQL load testing and discuss the tests that Hammerora implements.

Single Threaded Tests

Historically user database performance tests were often conducted with simple scripts and using the time to completion for a simple select statement, function or Cartesian join as a prediction of CPU performance , for example:

```
mysql>  select count(*) from information_schema.columns;
+----------+
| count(*) |
+----------+
|      736 |
+----------+
1 row in set (0.11 sec)


mysql>  select count(*) from information_schema.columns a,
information_schema.columns b, information_schema.columns c;
+-----------+
| count(*)  |
+-----------+
| 398688256 |
+-----------+
1 row in set (10.65 sec)
```

The challenge was that such a statement runs as single-threaded and therefore was valid in the era of single-core processors to test the performance of one processor but became obsolete in the era of multi-core processors. In other words with an example eight core processor such a test would give indications on the performance of one core and leave the other cores idle testing only a fraction of the performance potential of the CPU as a whole. Additionally such tests focused on CPU performance only without testing any of the storage component. Such a simple approach is flawed and to test a multiple CPU or multicore database environment requires a multithreaded test framework. Fortunately Hammerora is multi-threaded and therefore ready to test your multi-core environments with multiple virtual users all interacting independently with the database simultaneously.

What is TPC-C?

Designing and implementing a benchmark is a significant challenge. Many performance tests and tools experience difficulties in comparing system performance especially in the area of scalability, the ability of a test conducted on a certain system and schema size to be comparable with a test on a larger scale system. When system vendors wish to publish benchmark information about database performance they have long had to access to such sophisticated test specifications to do so. In particular all major database vendors recognise the TPC-C as the standard for Online Transaction Processing, the type of workload we are looking to simulate. Fortunately the *"TPC benchmarks are industry standards. The TPC, at no charge, distributes its benchmark specifications to the public."* For this reason Hammerora includes an implementation of the specification of the TPC-C benchmark that can be run in any MySQL environment. This implementation has the significant advantage that you know that the test is reliable, scalable and tested to produce consistent results. It is important to emphasise that the implementation is not a full specification TPC-C benchmark and the transaction results cannot be compared with the official published benchmarks in any way. Instead the implementations in Hammerora take the best designed specifications for a database transactional workload available in the world and enable you to run an accurate and repeatable workload against your own MySQL database. Audited TPC-C benchmarks are extremely costly and time consuming to establish and maintain, the Hammerora implementation of the TPC-C benchmarks is designed to capture the essence of TPC-C in a form that can be run at low cost bringing professional load testing to all MySQL environments.

TPC-C implements a computer system to fulfil orders from customers to supply products from a company. The company sells 100,000 items and keeps its stock in warehouses. Each warehouse has 10 sales districts and each district serves 3000 customers. The customers call the company whose operators take the order, each order containing a number of items. Orders are usually satisfied from the local warehouse however a small number of items are not in stock at a particular point in time and are supplied by an alternative warehouse. Figure 1 shows this company structure.

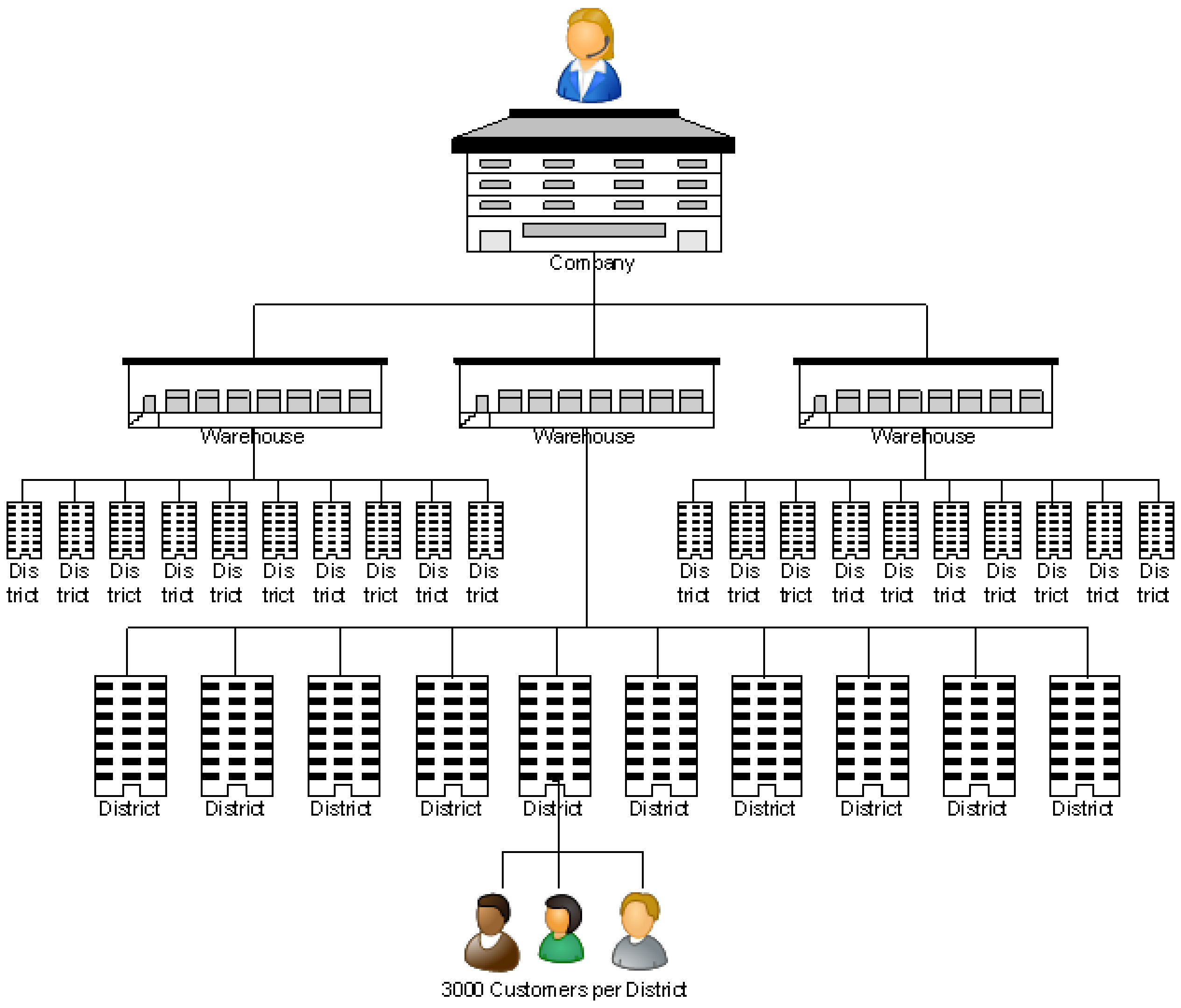


Figure 1 TPC-C Company Structure

It is important to note that the size of the company is not fixed and can add Warehouses and sales districts as the company grows. For this reason your test schema can be as small or large as you wish with a larger schema requiring a more powerful computer system to process the increased level of transactions. Figure 2 shows the TPC-C schema, in particular note how the number of rows in all of the tables apart from the ITEM table which is fixed is dependent upon the number of warehouses you choose to create your schema.

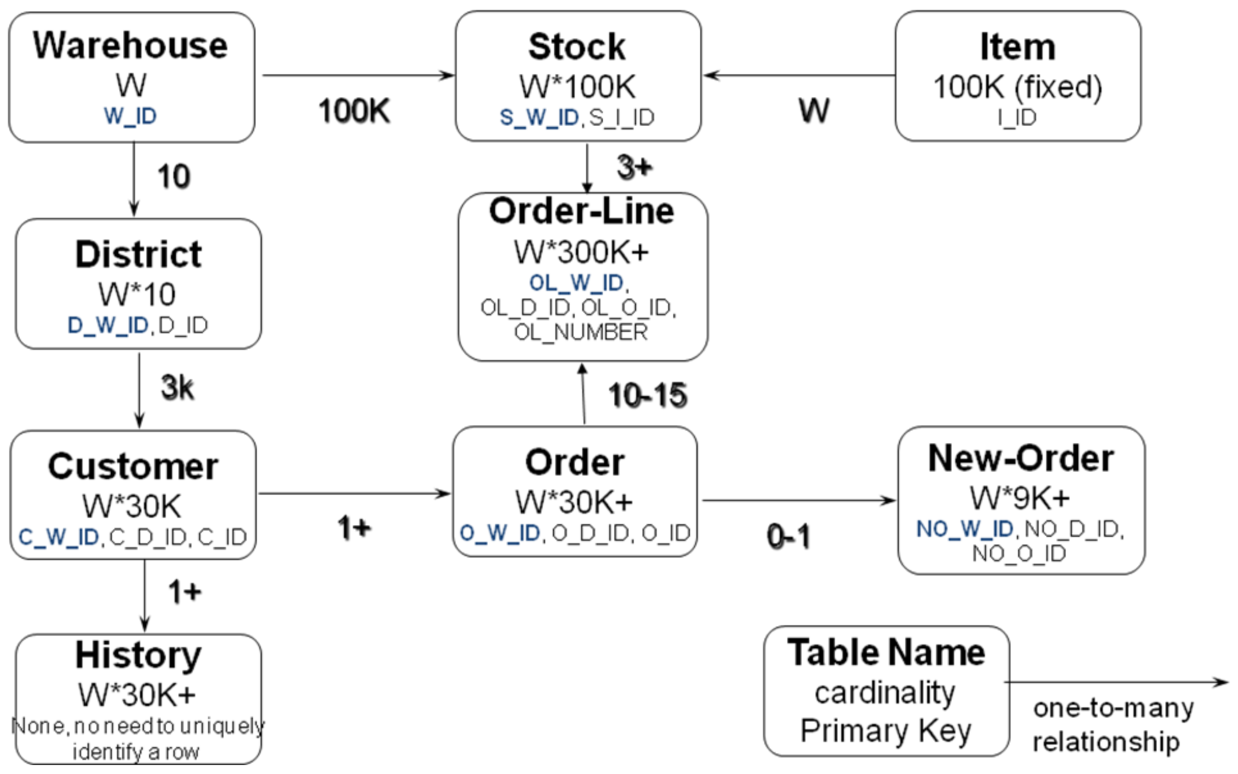


Figure 2 TPC-C Schema

For additional clarity please note that the term Warehouse in the context of TPC-C bears no relation to a Data Warehousing workload, TPC-C defines a transactional based system and not a decision support (DSS) one.

In addition to the computer system being used to place orders it also enables payment and delivery of orders and the ability to query the stock levels of warehouses. Consequently the workload is defined by a mix of 5 transactions as follows:

- New-order: receive a new order from a customer: 45%
- Payment: update the customers balance to record a payment: 43%
- Delivery: deliver orders asynchronously: 4%
- Order-status: retrieve the status of customer's most recent order: 4%
- Stock-level: return the status of the warehouse's inventory: 4%

Performance Profiles

For an official audited TPC-C benchmark the result of the tests is detailed as tpmC which represents the number of New Orders processed only. One particular advantage of Hammerora is is the ability to generate a performance profile as the load increases on your system. Whereas an official TPC-C benchmark gives you a single data-point and a typical single-threaded test (such as timing SQL Statements) also gives you a single data-point. Consider the graph shown in figure 3.

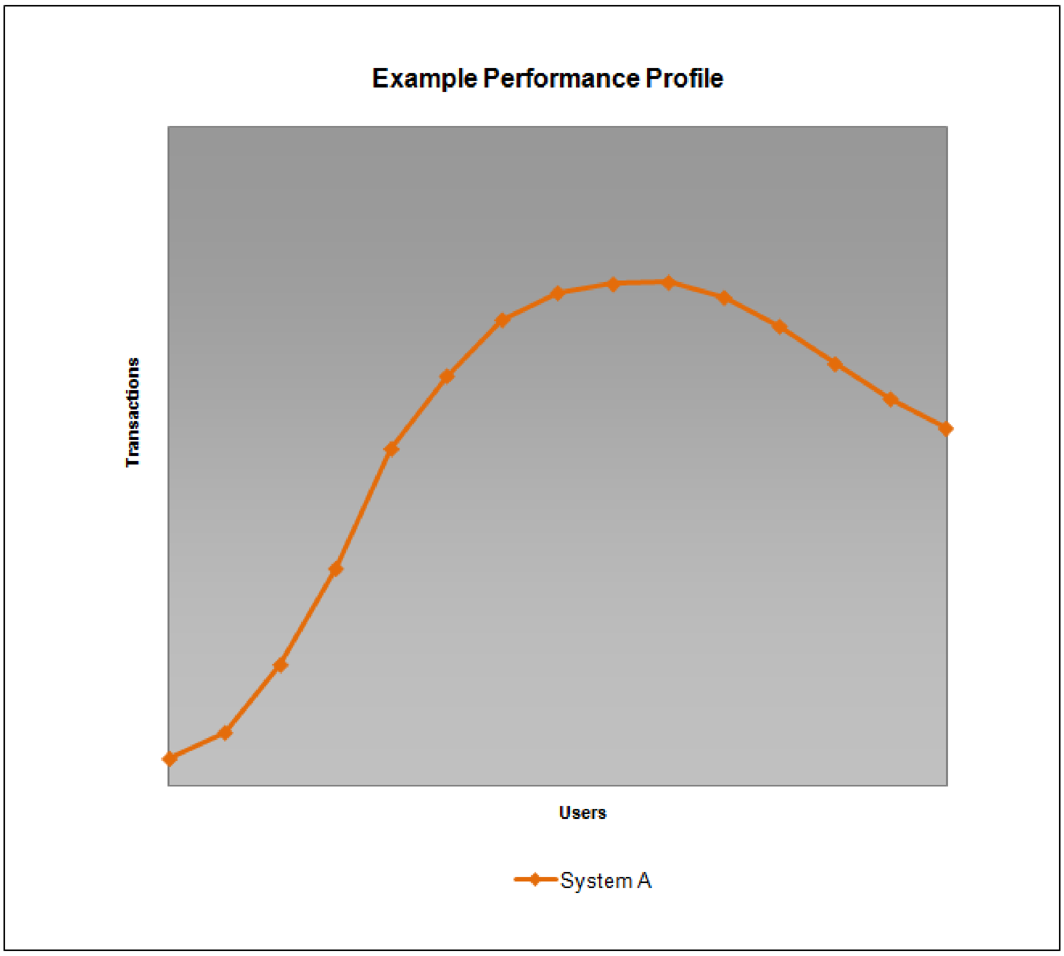


Figure 3 Performance Profile Example

This graph shows the performance of a real tests on a MySQL configuration. If observed as a single data point the tenth data point from the right shows the highest performance. However the performance profile shows a point beyond this at which performance starts to reduce as utilisation increases. The best returns from System A will be to keep utilisation within or below the bounds of optimal performance. It should be clear that your testing goal should be to measure the performance profile of your system across all levels of utilisation.

Test Network Configuration

As shown in figure 4 the network architecture required for Hammerora is both simple and straightforward. You require the database server to be tested known as the system under test (SUT) installed and configured with the MySQL database. You also require a load generation server to run Hammerora installed with the Hammerora software and a MySQL client. Typically the administrator will monitor and manage the load testing from a separate notebook or PC. All systems will be connected across a network. Technically it is possible to run Hammerora on the SUT however this is not recommended. Firstly it makes it comparatively harder to distinguish between the load generation server workload and the database workload. Secondly running the Hammerora workload will skew the results. By eliminating the network component of the workload results for a smaller number of virtual users will be comparatively higher however as the workload increases performance will be comparatively lower. To eliminate this skew in results a dedicated load generation server should be used.

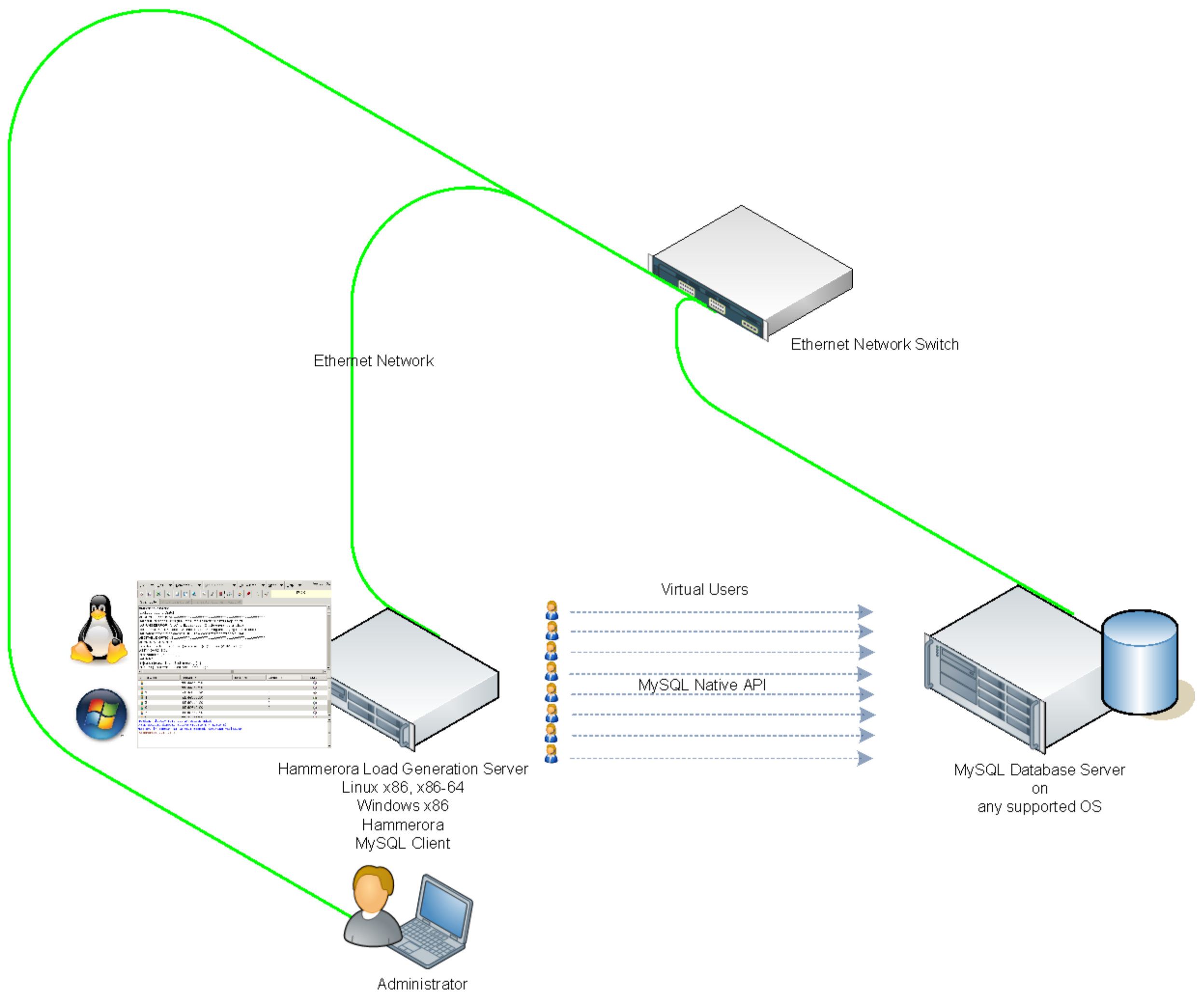


Figure 4 Hammerora Network Architecture

Load Generation Server Configuration

The most important component of the load generation server is the server processor. It is recommend to use an up to date multicore and multithreaded processor. Hammerora is a multithreaded application and implicitly benefits from a multicore server CPU such as the Intel Xeon 5XXX series range. To determine whether CPU capacity is sufficient for testing you can monitor the CPU utilisation with utilities such as **top** on Linux or **Task Manager** on Windows during testing. CPU utilisation reaching 100% is an indication that the CPU on the load generation server is limiting performance. It is important to note however that Hammerora is highly efficient and a high-performance multicore processor such as the Xeon 55XX and upwards will likely only see utilisation in the 10% range to drive a much larger database server to 100% utilisation. For the load generation memory requirement a rough guide is that Hammerora requires approximately 10MB for the application and 2MB per virtual user, for example 64 virtual users will need 138MB of memory. Again this represents a highly efficient load testing environment in comparison to commercial database load testing applications. Consequently it is it is entirely feasible to load test with a 32-bit x86 operating system on the load generation client with a 64-bit operating system only required when conducting tests in excess of 1000 virtual users. For the load testing operating system, Hammerora is available pre-compiled for 32-bit Windows, 32-bit Linux and 64-bit Linux however you may compile the packages used for Hammerora manually for another operating system if you wish. Hammerora is a graphical application and therefore on Linux the operating system installation must include the X-windows packages. Storage requirements on the load generation server are minimal and all modern servers are likely to meet the storage required. Hammerora consumes approximately 15MB of disk space and you will also need an MySQL client. All MySQL database installations include a MySQL client and therefore the MySQL client will often be the same MySQL software that you have installed on the server as this also includes the client libraries that Hammerora requires. The load generation server does not need to be running the same version of MySQL, however Hammerora does require MySQL 5.1 upwards for the correct version of the client libraries. You should note that a load generation server built with Linux or Windows will be able to connect to and test a MySQL Database running on any operating system you choose.

SUT Database Server Configuration

The database server architecture to be tested must meet the standard requirements for a MySQL Database Server. MySQL can be installed on any supported operating system, there is no restriction on the version of MySQL that is required. To run a Hammerora transactional load test there are minimum requirements in memory and I/O (disk performance) to prevent these components being a bottleneck on performance. In turn the memory and I/O is determined by the capabilities of the CPUs installed. For a test database that requires the minimal level of memory and I/O and therefore keying and thinking time is set to FALSE (keying and thinking time is detailed later in this guide) you should aim to create a database with approximately a factor of 5 warehouses multiplied by the number of cores/threads (threads refers to Hyper Threads, so if a processor has 4 cores but supports 8 Hyper Threads then the value of 8 should be used) supported by system processor rounded up to the nearest 100. For example for a system with 64 threads, 64 * 5 = 320, rounded up to 400. A system with 24 threads would be configured with 200 warehouses. To calculate the disk storage required by your test database allow a minimum of 100MB per warehouse plus 50% for additional growth. For the example system that supports 64 threads that would be 60GB of storage space. You should have sufficient memory to cache as much of your test database in memory as possible without factoring in the additional storage for growth therefore for 64 threads a minimum memory requirement for the db_cache_size parameter would be 40GB. Reductions in memory will place more emphasis on the I/O performance. The database server should be connected to external storage typically in the form of a SAN or NAS. A database server without external storage will not sustain the level of I/O throughput necessary to conduct a load test. I recommend SSD disks to minimise the risk of I/O constraining performance and a RAID configuration of 8 SSD disks is a good minimum configuration. For hard disks a minimum configuration of 12 disk drives is recommended. Where possible RAID 0 should be used to maximise the potential performance of the disk drives available. If your I/O configuration does not meet minimum requirements it is worthwhile taking a moment to question whether to proceed with a performance test that will result in a conclusion that the storage is inadequate. For transactional tests I/O should be configured in particular to maximize write performance to the log files, depending on the performance of your processors you should allow for the largest log files possible. You should consider that the log files of drop-in replacement databases for MySQL may have larger limits thereby offering improved performance. There is less emphasis on the storage performance of the data files provided that enough memory is available to cache the test database and database checkpoints are not frequent during the test. It is worth reiterating that these guidelines define the minimum requirements for testing and you should aim for your SUT to exceed these minimum standards where possible. In particular where you are using keying and thinking times the number of virtual users required and the memory and I/O requirements will increase considerably.

Administrator PC Configuration

The administrator PC has the minimal requirement to display the graphical output from the load generation server. for example for a Linux load generation server the ability to display X windows. The PC should also have the ability to connect to the SUT to monitor performance by the installation of a MySQL client.

Installation and Configuration

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

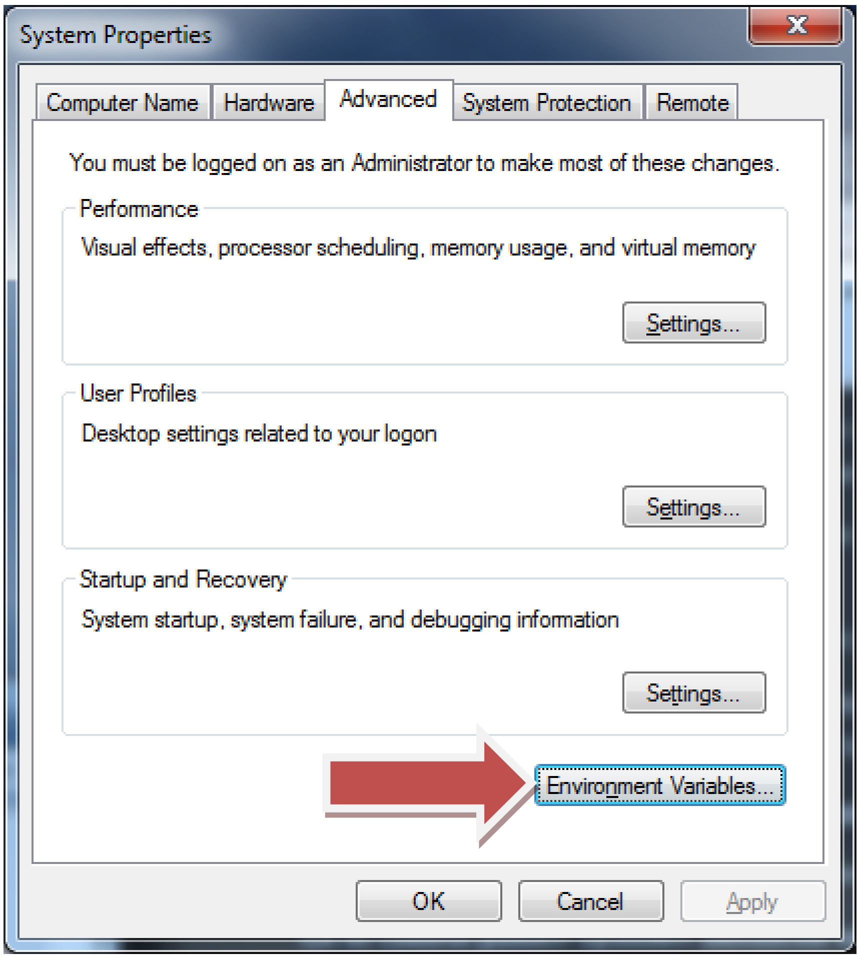
Load Generation Server Installation

On the Load Generation Server you must have already installed the MySQL client software and once Hammerora is installed it must know where to find the client libraries. The MySQL client software is also included in a MySQL database software installation. On Linux the important environment variables are PATH and LD_LIBRARY_PATH which should be set to the location of the MySQL client library. The MySQL client library will be found in the MySQL lib directory such as the /usr/local/mysql/ lib directory. Once done so Hammerora will be able to find the client library such as shown example:

```
/usr/local/mysql/lib/libmysqlclient.so.16
```

After installation on starting Hammerora if the client library is not accessible Hammerora will report an error. This must be resolved before Hammerora will interact with a MySQL environment. Ensure however that the error reported relates to the MySQL libraries. Hammerora will report an error when it cannot find either the Oracle or MySQL libraries and therefore when testing MySQL the error related to finding Oracle libraries can be safely ignored.

On Windows the PATH environment variable is important and can be set on your System Properties, Advanced Tab as shown in figure 5.


System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

You must be logged on as an Administrator to make most of these changes.

Performance

Visual effects, processor scheduling, memory usage, and virtual memory

Settings...

User Profiles

Desktop settings related to your logon

Settings...

Startup and Recovery

System startup, system failure, and debugging information

Settings...

Environment Variables...

OK | Cancel | Apply

Figure 5 Windows Environment Variables

Note that with a graphical MySQL install the installer gives you the option of setting the PATH environment variable automatically.

Once you have installed your MySQL client download Hammerora from Sourceforge here:

http://sourceforge.net/projects/hammerora/

The page will show you the right version for your operating system. If you need a different version click "view all files". For Linux make sure that you have the correct version for your 32-bit or 64-bit Linux operating system respectively. If you have a 32-bit operating system you will need the

hammerora-2.5-Linux-x86 package

If you have a 64-bit operating system you will need the

hammerora-2.5-Linux-x86-64 package

Make sure that you use the correct version the 32-bit install will not function on 64-bit and Vice Versa.

On Linux Hammerora should be installed as the mysql user, the same user that owns the MySQL software.

To start the installer on Linux make the installer file executable

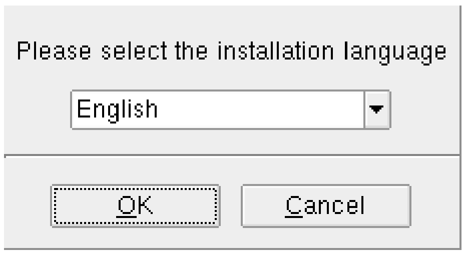
```
mysql@test:~> chmod u+x hammerora-2.5-Linux-x86
```

and run the executable

```
mysql@test:~> ./hammerora-2.5-Linux-x86
```

On Windows double-click on the installer file. For Windows 7 if you have the correct permissions to do so you should right click on the installer and choose the option “Run as Administrator”

The installer will start giving you the option of selecting the installation language shown in figure 6.


Please select the installation language

English

Figure 6 Select installation language

You can then choose whether to continue with the installation as shown in figure 7.

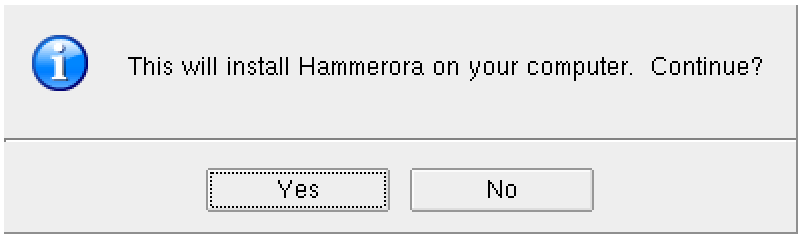
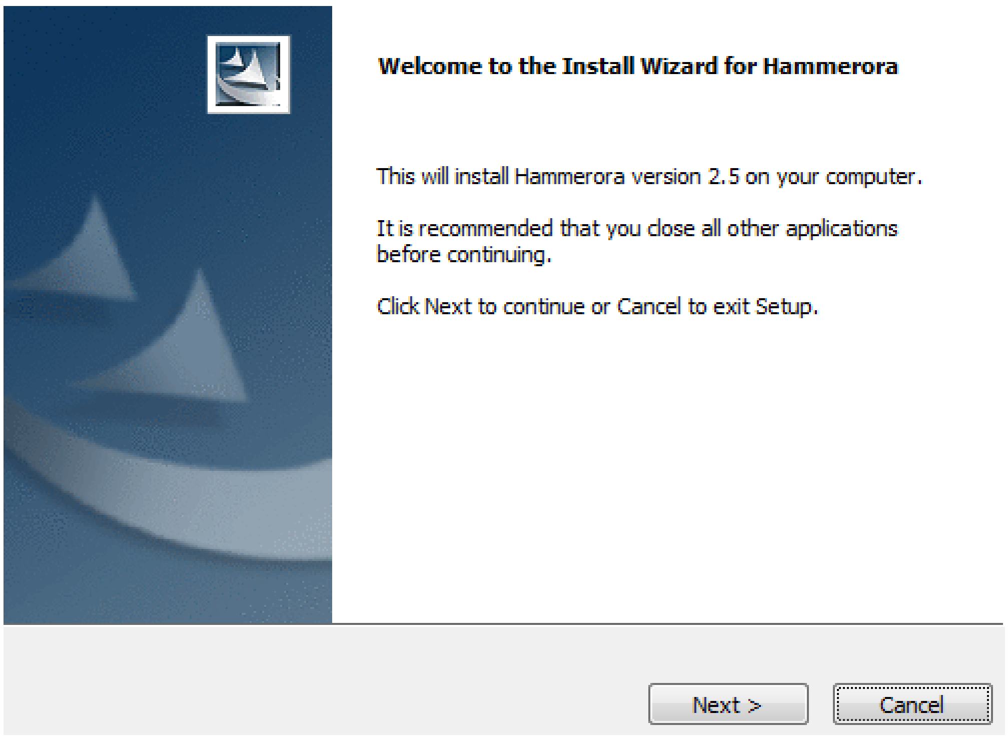

This will install Hammerora on your computer. Continue?

Figure 7 Continue?

At the start of the install wizard, click next as shown in figure 8.


Welcome to the Install Wizard for Hammerora

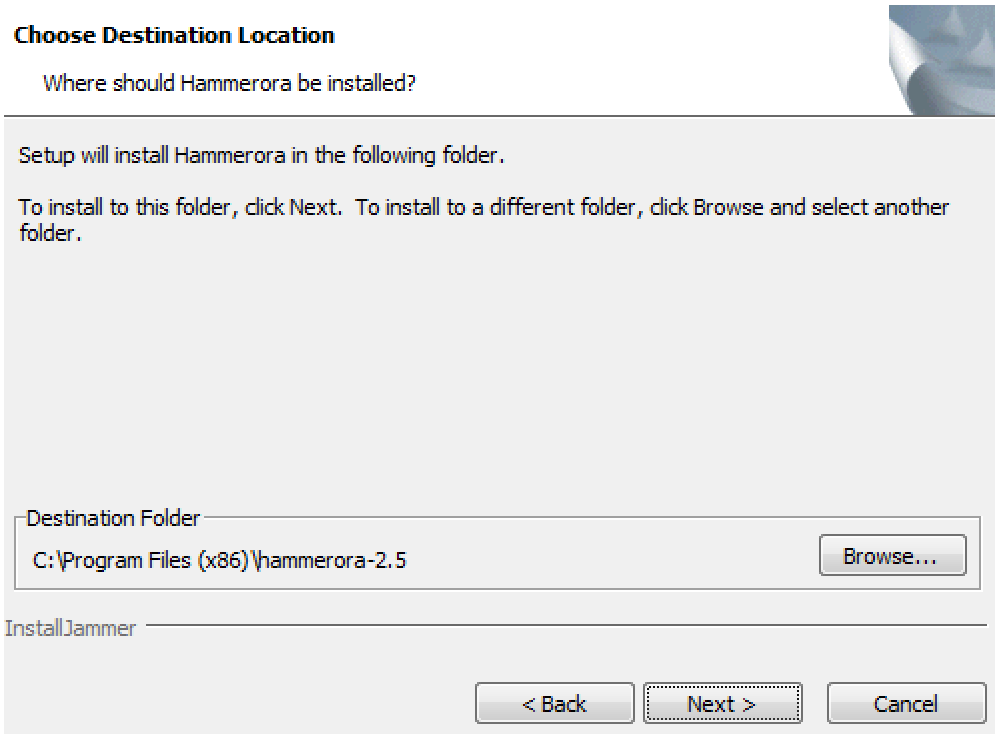
This will install Hammerora version 2.5 on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue or Cancel to exit Setup.

Figure 8 Welcome

Choose the destination location as shown in figure 9 and Click Next. To change the default location Click Browse and select a new location


Choose Destination Location

Where should Hammerora be installed?

Setup will install Hammerora in the following folder.

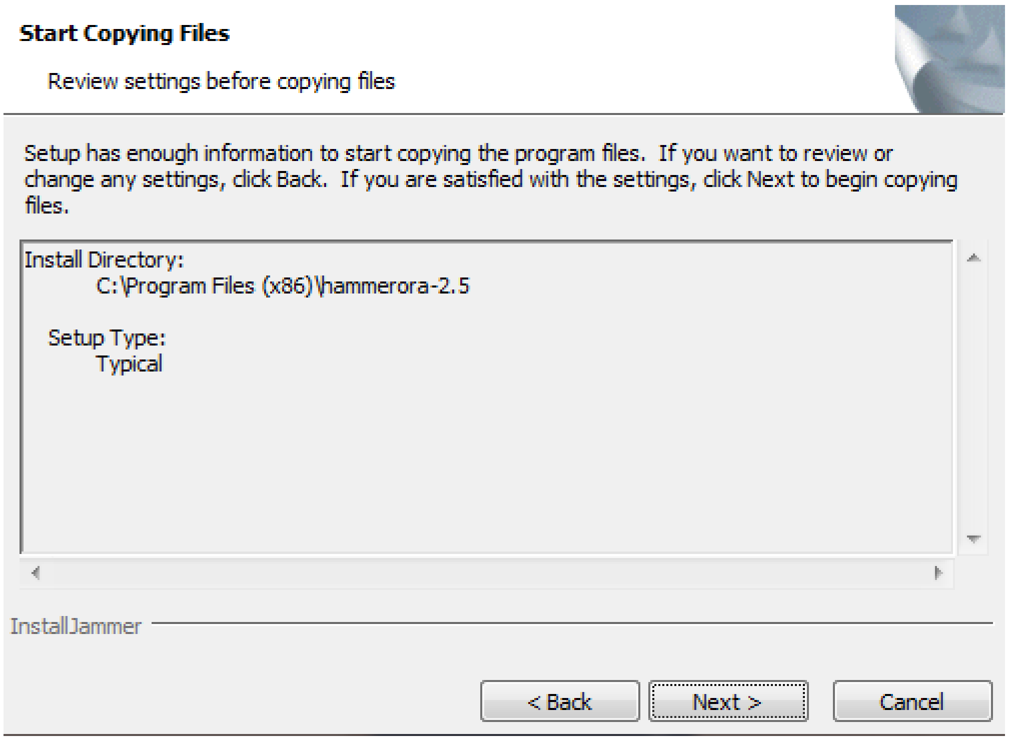
To install to this folder, click Next. To install to a different folder, click Browse and select another folder.

Destination Folder

C:\Program Files (x86)\hammerora-2.5

Figure 9 Choose Destination

Click next as shown in figure 10 to start copying the installation files


Start Copying Files

Review settings before copying files

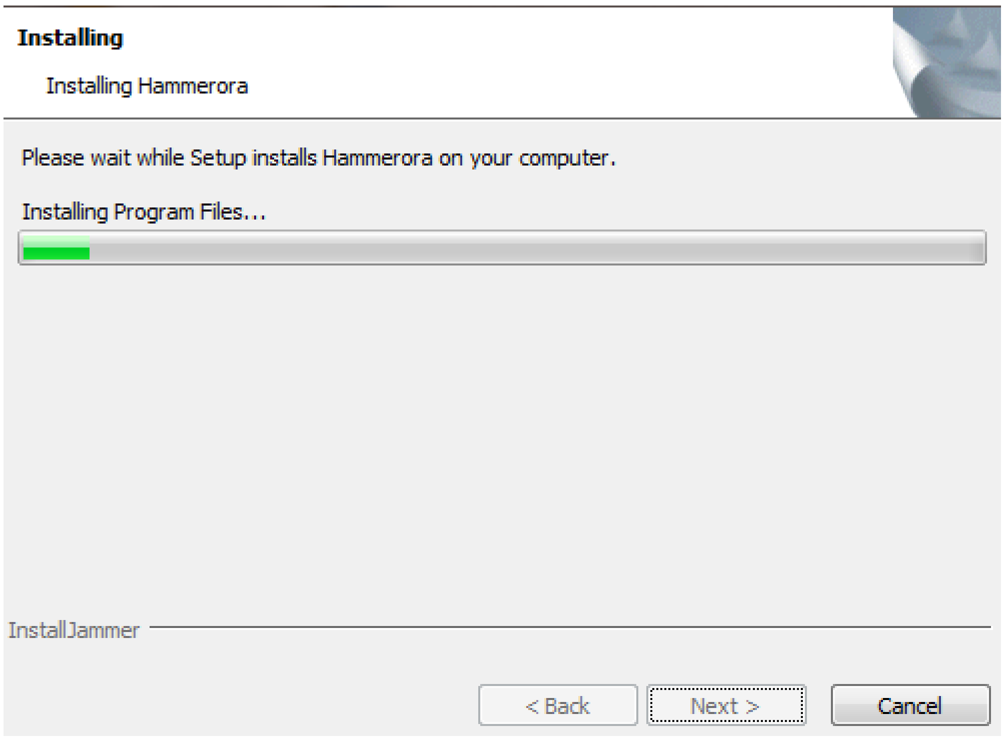
Setup has enough information to start copying the program files. If you want to review or change any settings, click Back. If you are satisfied with the settings, click Next to begin copying files.

Install Directory:
    C:\Program Files (x86)\hammerora-2.5

Setup Type:
    Typical

Figure 10 Start copying Files

Hammerora will be installed in your selected location as shown in figure 11. Note that the installation is entirely self contained. No software is installed external to the selected directory.


Installing

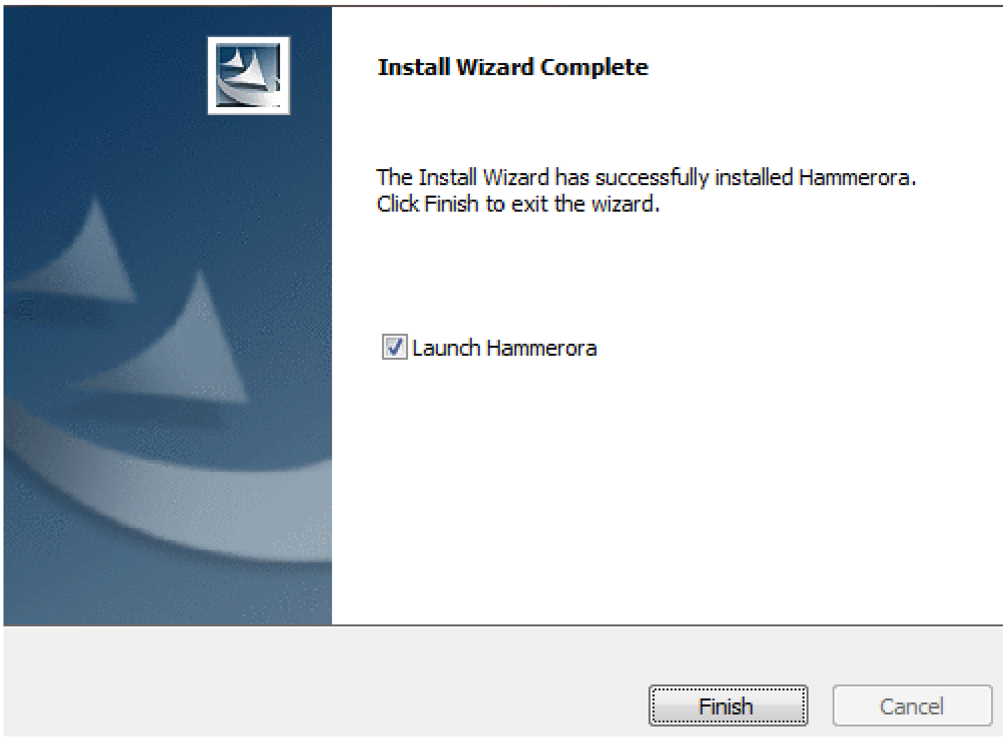
Installing Hammerora

Please wait while Setup installs Hammerora on your computer.

Installing Program Files...

Figure 11 Installing

On the completion screen shown in figure 12 click finish and optionally launch Hammerora at this time.

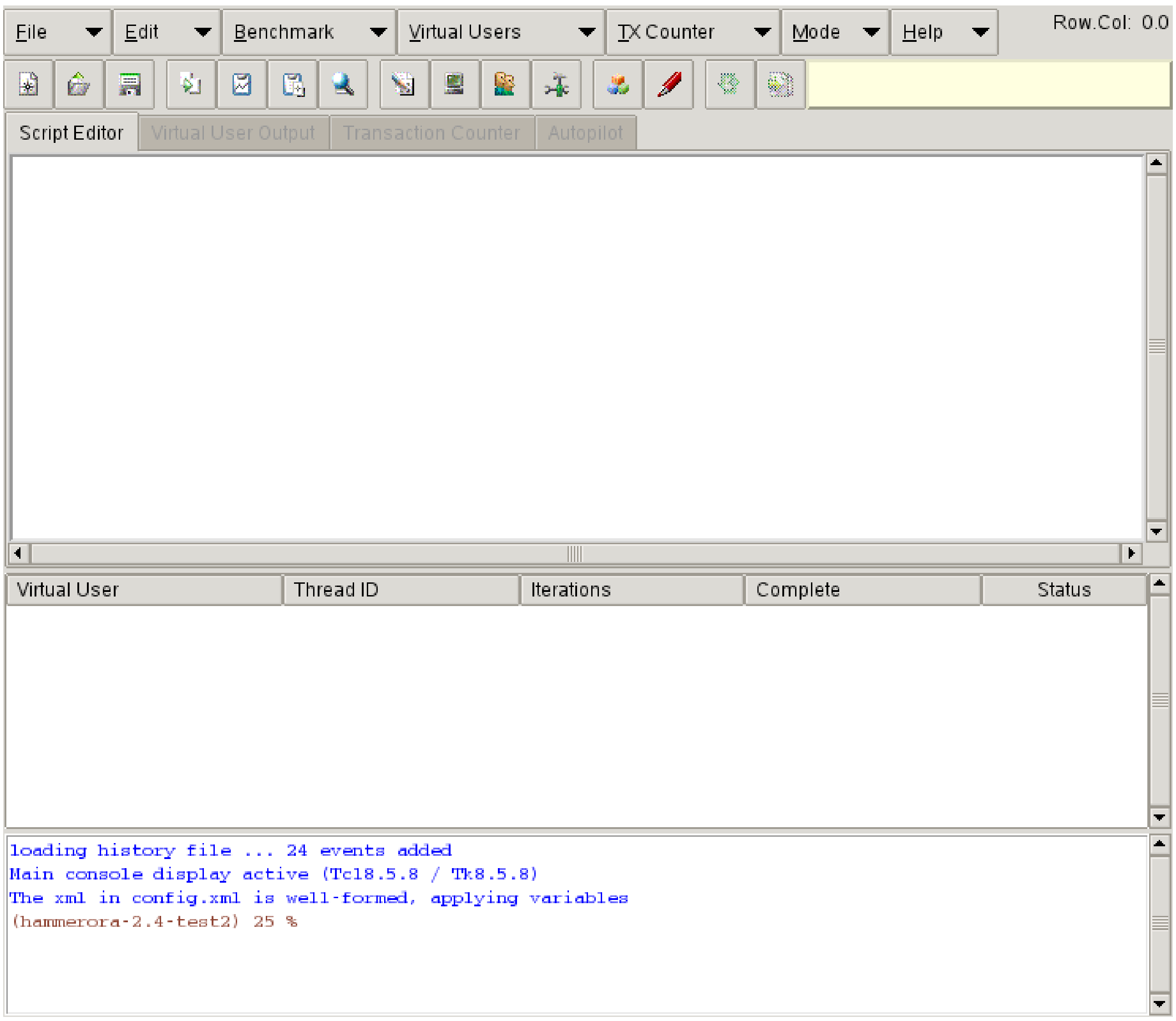

Install Wizard Complete

The Install Wizard has successfully installed Hammerora.
Click Finish to exit the wizard.

☑ Launch Hammerora

Figure 12 Install Complete

If you opt to launch Hammerora as shown in figure 13 a the main application window is displayed



```
loading history file ... 24 events added
Main console display active (Tcl8.5.8 / Tk8.5.8)
The xml in config.xml is well-formed, applying variables
(hammerora-2.4-test2) 25 %
```

Figure 13 Hammerora

Hammerora is now installed. If you close the application using Flie -> Exit you can restart it again under Linux by running

./hammerora.tcl as the mysql user or windows by double clicking on hammerora.bat.

Load Generation Server Configuration

All of Hammerora's working data can be set using menu options. However if you wish in the Hammerora home directory there is a configuration file called config.xml that is read on startup. In this file you can preset your database build and driver configurations by editing the xml file without having to change the data manually. You should always set the <rdbms> option to MySQL with further MySQL related choices in the <mysql> section of the file. If your xml file is well formed your variables will be applied to Hammerora when you selected the menu options.

```
<?xml version="1.0" encoding="utf-8"?>
<hammerora>
    <rdbms>MySQL</rdbms>
    <bm>TPC-C</bm>
<oracle>
...
```

```
</oracle>
<mysql>
    <connection>
        <mysql_host>127.0.0.1</mysql_host>
        <mysql_port>3306</mysql_port>
    </connection>
      <tpcc>
        <schema>
            <my_count_ware>1</my_count_ware>
            <mysql_num_threads>1</mysql_num_threads>
            <mysql_user>root</mysql_user>
            <mysql_pass>mysql</mysql_pass>
            <mysql_dbase>tpcc</mysql_dbase>
            <storage_engine>innodb</storage_engine>
        </schema>
        <driver>
            <my_total_iterations>1000000</my_total_iterations>
            <my_raiseerror>false</my_raiseerror>
            <my_keyandthink>false</my_keyandthink>
            <mysqldriver>standard</mysqldriver>
          <my_rampup>2</my_rampup>
            <my_duration>5</my_duration>
        </driver>
    </tpcc>
</mysql>
…
```

SUT Database Server Installation

Installation and configuration of the MySQL Database on your chosen operating system is beyond the scope of this document. We recommend using one of the many tutorials available on the web for example: http://dev.mysql.com/doc/refman/5.5/en/installing.html . You should have the MySQL database software installed and running. Make sure you set a password for either the root user or a user with the correct privileges to create the TPC-C database, for example:

```
[root@sut bin]# ./mysqladmin -u root password mysql
```

Network Connectivity

By default a MySQL installation will allow connection to the local server only, you must grant permission to connect to the MySQL database from your load generation server, the following example grants all permissions to the root user on the system called test.example.com.

```
mysql> grant all on *.* to root@'test.example.com' identified by
'mysql';

Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;

Query OK, 0 rows affected (0.00 sec)
```

Alternatively after the test database is created you can restrict the privileges to that databases only.

```
mysql> grant all on tpcc.* to root@'test.example.com' identified by
'mysql';
```

When choosing a MySQL Server to test note that Hammerora load testing can drive your system utilization to maximum levels and therefore testing an active production system is not recommended. When you have installed the load generation server and SUT database and have verified that you can communicate between them by logging in remotely you can proceed to building a test schema.

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 choosing select benchmark from under the TPC menu. The initial settings are determined by the values in your config.xml file. Select MySQL and TPC-C and press OK as shown in Figure 14.

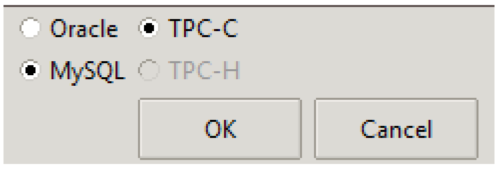

Oracle [✓ TPC-C]
[✓ MySQL] TPC-H

Figure 14 Select Benchmark

To create the TPC-C schema select the TPC-C schema options menu tab from the top level TPC menu. This menu will change dynamically according to your chosen database.

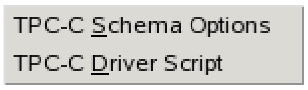
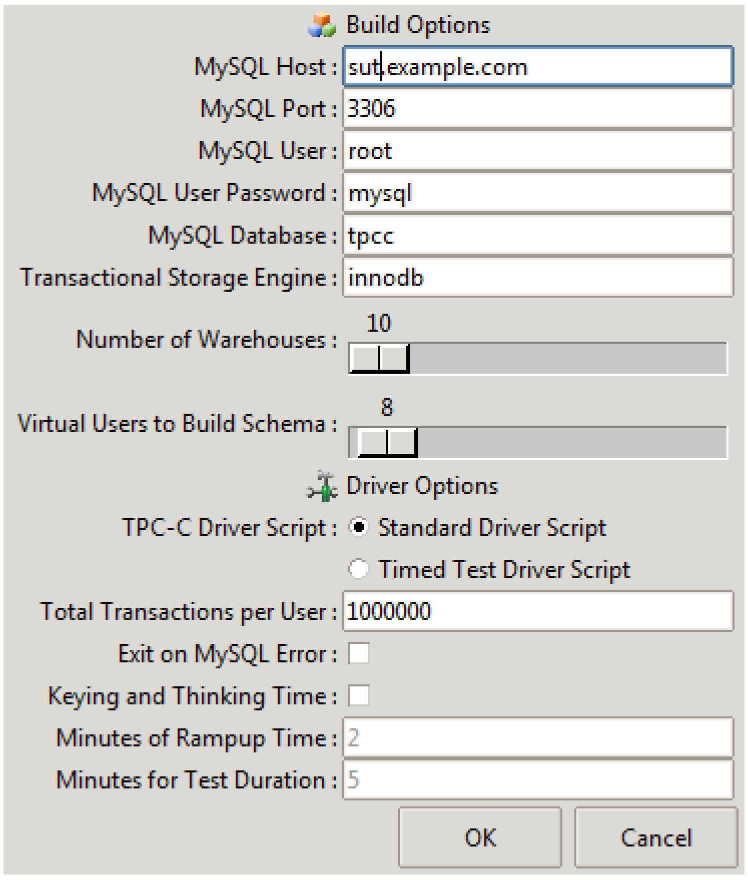


Figure 15 Select Schema Options

The schema options window is divided into two sections. The “Build Options” section details the general login information and where the database will be built and the “Driver Options” for the Driver Script to run after the database 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 database every time you change the “Driver Options” , once the database has been built only these “Driver Options” may need to be modified. For the “Build Options” fill in the values according to the database where the database will be built.



Build Options

MySQL Host :	sut.example.com
MySQL Port :	3306
MySQL User :	root
MySQL User Password :	mysql
MySQL Database :	tpcc
Transactional Storage Engine :	innodb
Number of Warehouses :	10
Virtual Users to Build Schema :	8

Driver Options

TPC-C Driver Script :	[✓ Standard Driver Script] Timed Test Driver Script
Total Transactions per User :	1000000
Exit on MySQL Error :	☐
Keying and Thinking Time :	☐
Minutes of Rampup Time :	2
Minutes for Test Duration :	5

Figure 16 Database Options

Build Options

The Build Option values have the following meanings.

MySQL Host

The MySQL Host Name is the host name that your load generation server will use to connect to the database running on the SUT database server. You verified connectivity to this host name previously in this document.

MySQL Port

The MySQL 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 3306.

MySQL User

The MySQL User is the user which has permission to create a database and you previously granted access to from the load generation server. The root user already exists in all MySQL databases and has the necessary permissions to create the TPC-C database.

MySQL User Password

The MySQL user password is the password for the user defined as the MySQL User. You previously set this password with the mysqladmin command. You will need to remember the MySQL user name and password for running the TPC-C driver script after the database is built.

MySQL Database

The MySQL Database is the database that will be created containing the TPC-C schema creation. There must have sufficient free space for the database to be created.

Transactional Storage Engine

The Transactional Storage Engine is the name of a Storage Engine that supports transactions. You can query your MySQL database to find the storage engines available as follows:

```
mysql> show engines \G
*************************** 1. row ***************************
      Engine: FEDERATED
     Support: NO
     Comment: Federated MySQL storage engine
Transactions: NULL
          XA: NULL
  Savepoints: NULL
*************************** 2. row ***************************
      Engine: MRG_MYISAM
     Support: YES
     Comment: Collection of identical MyISAM tables
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 3. row ***************************
      Engine: MyISAM
     Support: YES
     Comment: MyISAM storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 4. row ***************************
      Engine: BLACKHOLE
     Support: YES
     Comment: /dev/null storage engine (anything you write to it disappears
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 5. row ***************************
      Engine: CSV
     Support: YES
     Comment: CSV storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 6. row ***************************
      Engine: MEMORY
     Support: YES
     Comment: Hash based, stored in memory, useful for temporary tables
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 7. row ***************************
      Engine: ARCHIVE
     Support: YES
     Comment: Archive storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 8. row ***************************
      Engine: InnoDB
     Support: DEFAULT
     Comment: Supports transactions, row-level locking, and foreign keys
Transactions: YES
          XA: YES
  Savepoints: YES
*************************** 9. row ***************************
      Engine: PERFORMANCE_SCHEMA
     Support: YES
     Comment: Performance Schema
Transactions: NO
          XA: NO
  Savepoints: NO
9 rows in set (0.00 sec)
```

As shown here in this default installation only the InnoDB storage engine supports transactions and is therefore the only option available. However with a drop in replacement for MySQL you may have other options available that support transactions such as the Aria or XtraDB Storage Engines, Hammerora testing can help you find the right one for your requirements.

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

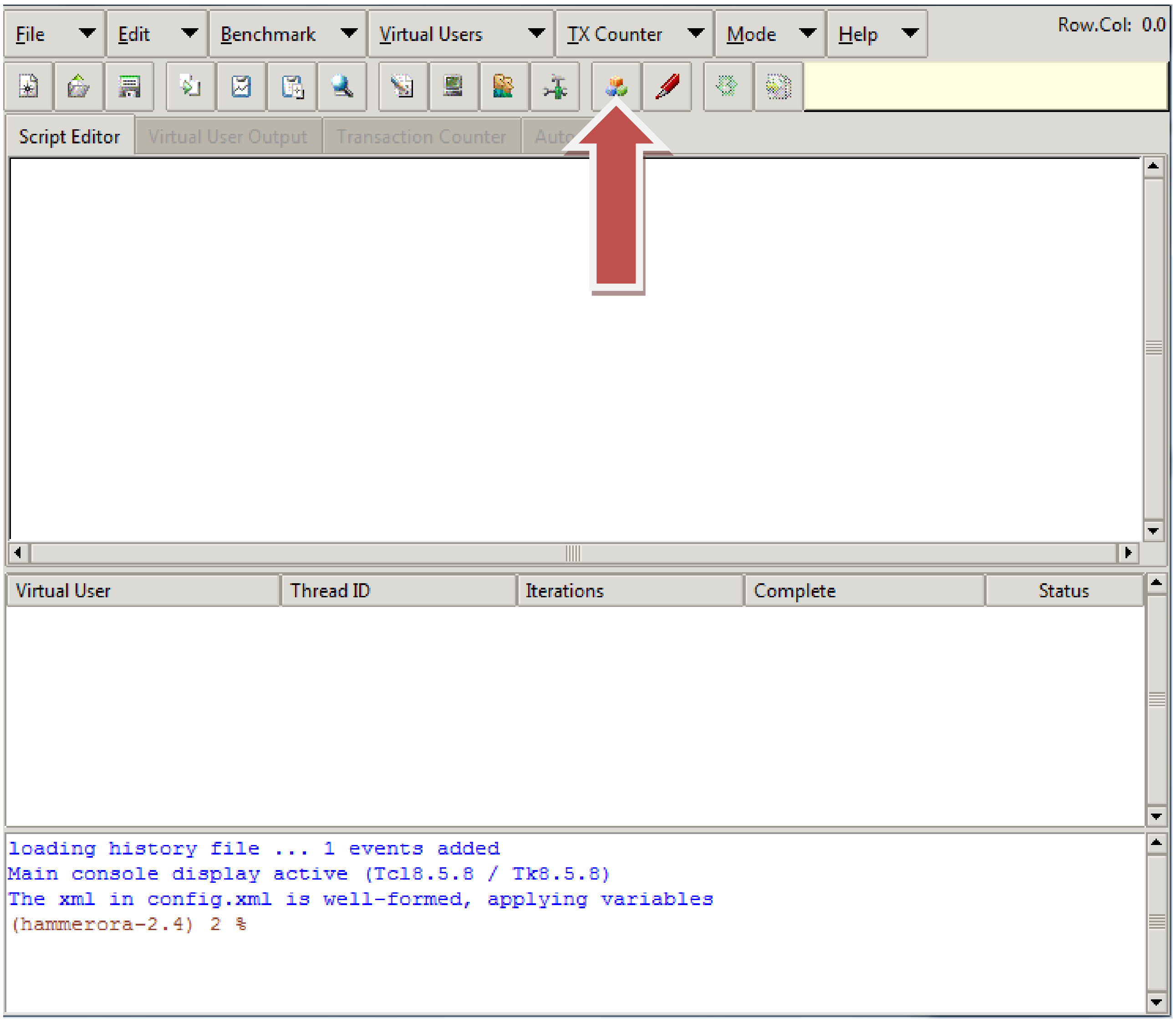
The Virtual Users to Build Schema is the number of Virtual Users to be created on the Load Generation Server that will complete your multi-threaded schema build. You should set this value to either the number of warehouses you are going to create (You cannot set the number of threads lower than the number 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 Hammerora the schema options will already contain the values you have entered in the corresponding fields, for example:

```
<tpcc>
    <schema>
      <my_count_ware>1</my_count_ware>
      <mysql_num_threads>1</mysql_num_threads>
      <mysql_user>root</mysql_user>
      <mysql_pass>mysql</mysql_pass>
      <mysql_dbase>tpcc</mysql_dbase>
      <storage_engine>innodb</storage_engine>
    </schema>
```

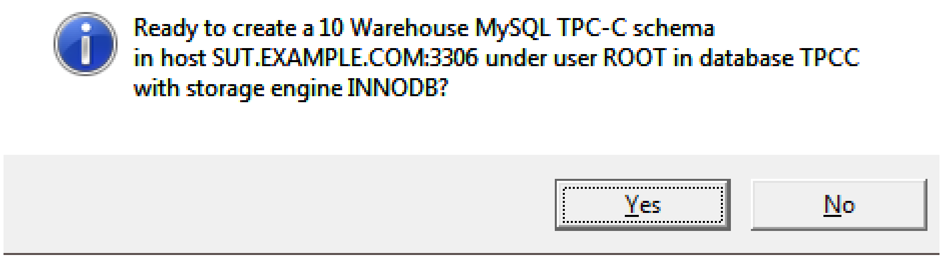
To begin the schema creation at the buttons in the top level window click the "Create TPC Schema" button. This button is shown as three coloured boxes and "Create TPC Schema" appears in the information box when moused over as shown in Figure 17.



```
loading history file ... 1 events added
Main console display active (Tcl8.5.8 / Tk8.5.8)
The xml in config.xml is well-formed, applying variables
(hammerora-2.4) 2 %
```

Figure 17 Create Schema

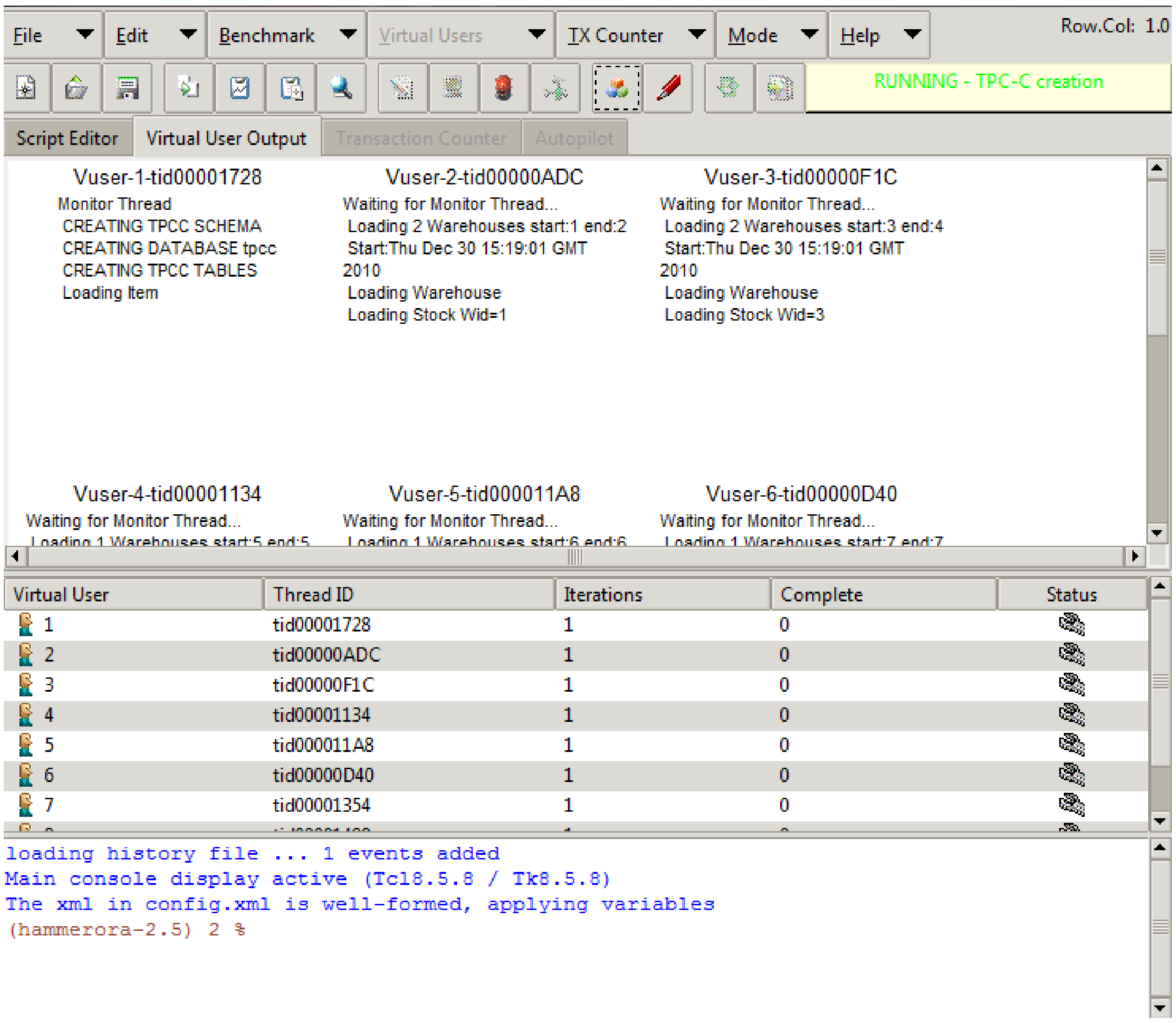
On clicking this button a dialogue box such as the one shown in Figure 18 appears.



Ready to create a 10 Warehouse MySQL TPC-C schema
in host SUT.EXAMPLE.COM:3306 under user ROOT in database TPCC
with storage engine INNODB?

Figure 18 Confirm Schema

When you click Yes Hammerora will login to your chosen MySQL 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 19. There are no intermediate data files or manual builds required, Hammerora will both create and load your requested data dynamically. Data is inserted in a batch format for optimal network performance.



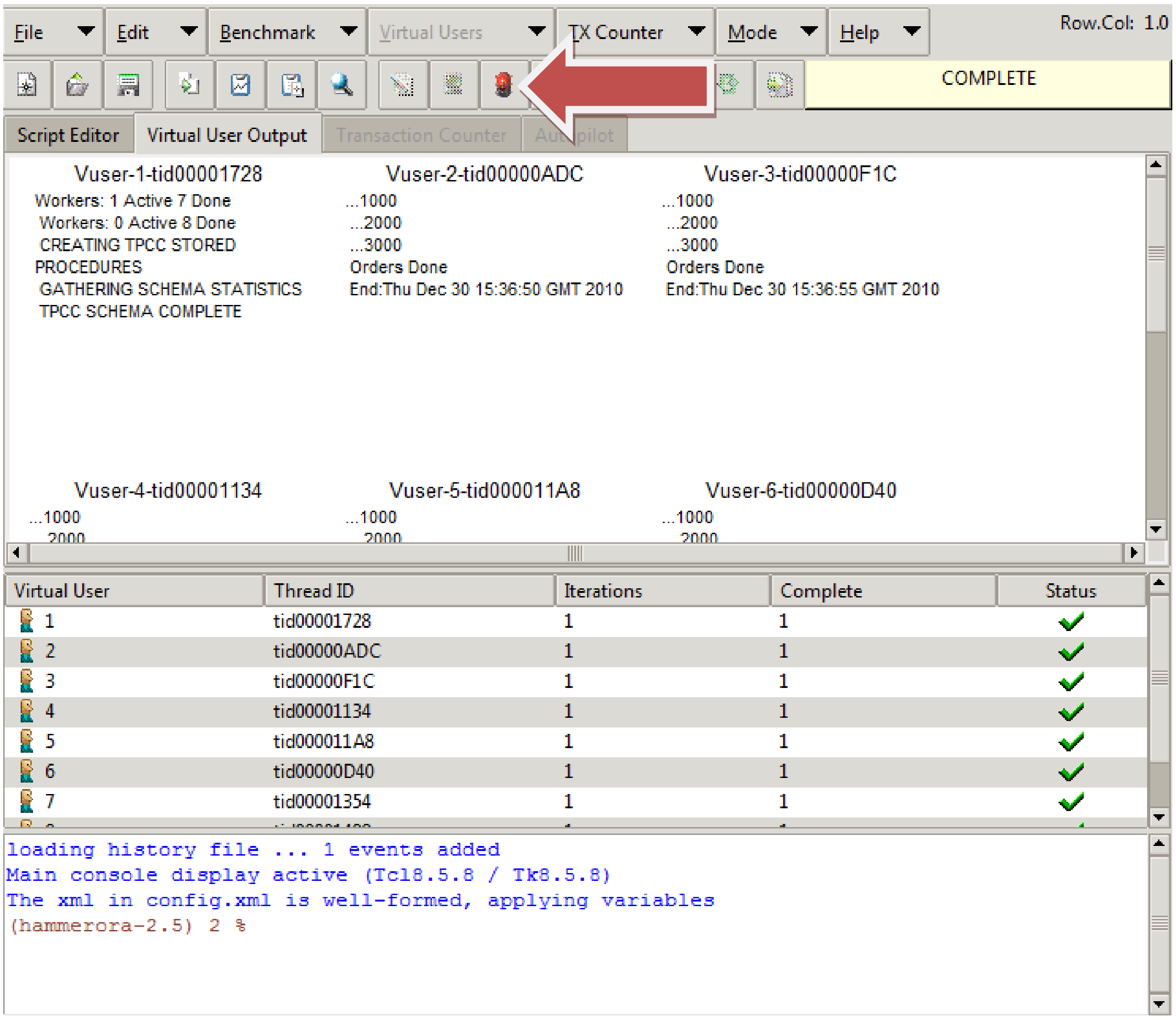
```
Vuser-1-tid00001728
Monitor Thread
 CREATING TPCC SCHEMA
 CREATING DATABASE tpcc
 CREATING TPCC TABLES
 Loading Item

Vuser-2-tid00000ADC
Waiting for Monitor Thread...
 Loading 2 Warehouses start:1 end:2
 Start:Thu Dec 30 15:19:01 GMT
2010
 Loading Warehouse
 Loading Stock Wid=1

Vuser-3-tid00000F1C
Waiting for Monitor Thread...
 Loading 2 Warehouses start:3 end:4
 Start:Thu Dec 30 15:19:01 GMT
2010
 Loading Warehouse
 Loading Stock Wid=3

Vuser-4-tid00001134
Waiting for Monitor Thread...

Vuser-5-tid000011A8
Waiting for Monitor Thread...

Vuser-6-tid00000D40
Waiting for Monitor Thread...
```

Virtual User	Thread ID	Iterations	Complete	Status
1	tid00001728	1	0	
2	tid00000ADC	1	0	
3	tid00000F1C	1	0	
4	tid00001134	1	0	
5	tid000011A8	1	0	
6	tid00000D40	1	0	
7	tid00001354	1	0	

```
loading history file ... 1 events added
Main console display active (Tcl8.5.8 / Tk8.5.8)
The xml in config.xml is well-formed, applying variables
(hammerora-2.5) 2 %
```

Figure 19 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 20.



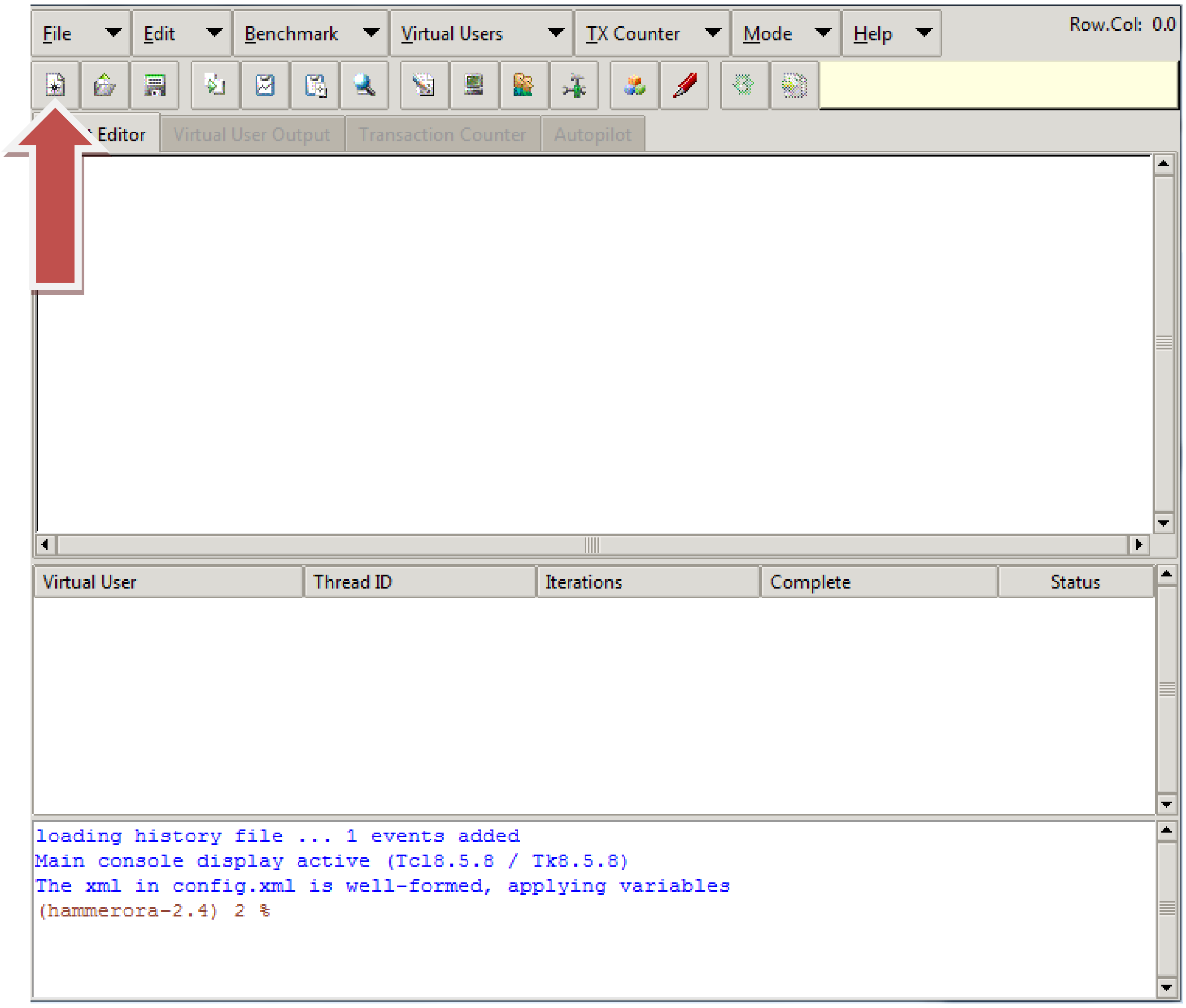
```
Vuser-1-tid00001728              Vuser-2-tid00000ADC              Vuser-3-tid00000F1C
Workers: 1 Active 7 Done         ...1000                          ...1000
 Workers: 0 Active 8 Done        ...2000                          ...2000
 CREATING TPCC STORED            ...3000                          ...3000
PROCEDURES                       Orders Done                      Orders Done
 GATHERING SCHEMA STATISTICS     End:Thu Dec 30 15:36:50 GMT 2010 End:Thu Dec 30 15:36:55 GMT 2010
 TPCC SCHEMA COMPLETE

Vuser-4-tid00001134              Vuser-5-tid000011A8              Vuser-6-tid00000D40
...1000                          ...1000                          ...1000
```

Virtual User	Thread ID	Iterations	Complete	Status
1	tid00001728	1	1	✓
2	tid00000ADC	1	1	✓
3	tid00000F1C	1	1	✓
4	tid00001134	1	1	✓
5	tid000011A8	1	1	✓
6	tid00000D40	1	1	✓
7	tid00001354	1	1	✓

```
loading history file ... 1 events added
Main console display active (Tcl8.5.8 / Tk8.5.8)
The xml in config.xml is well-formed, applying variables
(hammerora-2.5) 2 %
```

Figure 20 Schema Build Complete

Press the button to destroy the virtual users as shown in figure 20 and clear the script editor as shown in figure 21.



```
loading history file ... 1 events added
Main console display active (Tcl8.5.8 / Tk8.5.8)
The xml in config.xml is well-formed, applying variables
(hammerora-2.4) 2 %
```

Figure 21 Schema Creation End

The schema build is now complete as an example a 200 warehouse build as follows with nearly 100 million rows should take approximately half an hour or less to create and insert on an up to date 2 socket Linux server.

The TPC-C schema creation script is a standard Hammerora script like any other so you can save it, modify it and re-run it just like any other Hammerora 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 sut.example.com 3306 10000 root mysql tpcc innodb 8

Similarly change any other value to modify your script. If you have made a mistake simply close the application and run the following SQL to undo the database you have created.

```
SQL>drop database tpcc;
```

When you have created your schema you can verify the contents with SQL or your favourite admin tool as the newly created user.

```
mysql> use tpcc;
Database changed
mysql> show tables;
+-----------------+
| Tables_in_tpcc |
+----------------+
| customer       |
| district       |
| history        |
| item           |
| new_order      |
| order_line     |
| orders         |
| stock          |
| warehouse      |
+----------------+
9 rows in set (0.00 sec)

mysql> select * from warehouse limit 1 \G
*************************** 1. row ***************************
      w_id: 1
     w_ytd: 3000000.00
     w_tax: 0.1300
    w_name: mBr6dkgK
w_street_1: FH0SO5CUEREo
w_street_2: cBcStSxKcIIs4IAUUsJy
    w_city: FKaak9ZBgtJr3Tr6gESW
   w_state: Tt
     w_zip: 432611111
1 row in set (0.00 sec)

mysql> show indexes from warehouse \G
*************************** 1. row ***************************
        Table: warehouse
   Non_unique: 0
     Key_name: PRIMARY
 Seq_in_index: 1
  Column_name: w_id
    Collation: A
  Cardinality: 10
     Sub_part: NULL
       Packed: NULL
         Null:
   Index_type: BTREE
      Comment:
Index_comment:
1 row in set (0.00 sec)

mysql> select routine_name from information_schema.routines where routine_schema
 = 'TPCC';
+--------------+
| routine_name |
+--------------+
| DELIVERY     |
| NEWORD       |
| OSTAT        |
| PAYMENT      |
| SLEV         |
+--------------+
5 rows in set (0.03 sec)
```

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 tablespace space the results of tests against different sizes are comparable.

Pre-Testing and Planning

After database 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 MySQL 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 Hammerora 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 and select the same Schema Options you selected before the schema build as shown in Figure 22.

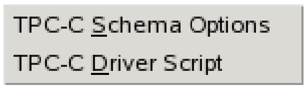
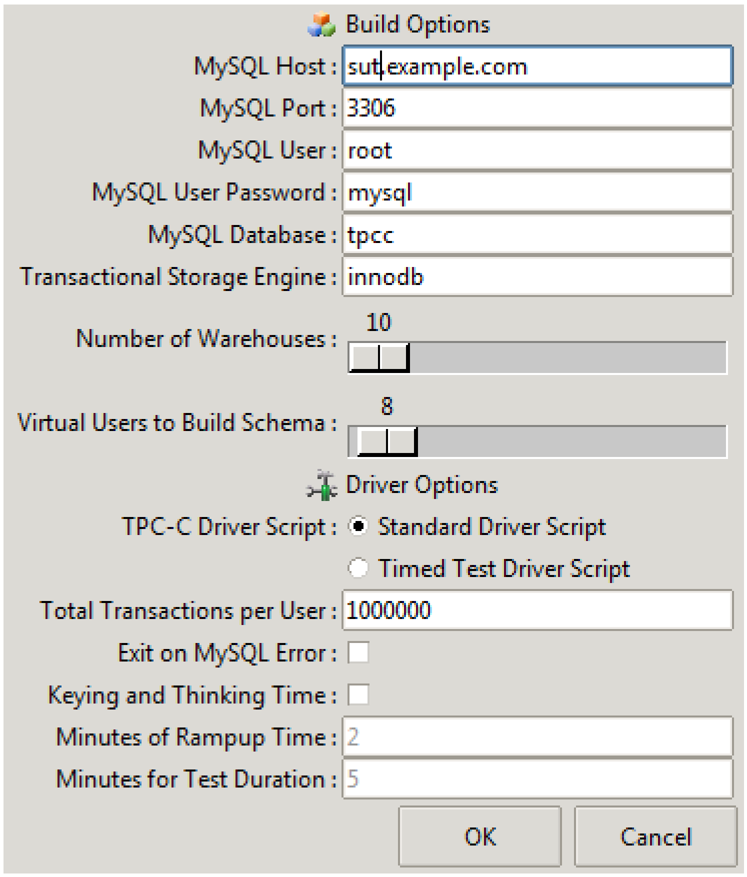


Figure 22 Select Schema Options

At this stage your focus is now on the options given under the section Driver Options as shown in Figure 23.



Build Options

MySQL Host :	sut.example.com
MySQL Port :	3306
MySQL User :	root
MySQL User Password :	mysql
MySQL Database :	tpcc
Transactional Storage Engine :	innodb
Number of Warehouses :	10
Virtual Users to Build Schema :	8

Driver Options

TPC-C Driver Script :	[✓ Standard Driver Script] Timed Test Driver Script
Total Transactions per User :	1000000
Exit on MySQL Error :	☐
Keying and Thinking Time :	☐
Minutes of Rampup Time :	2
Minutes for Test Duration :	5

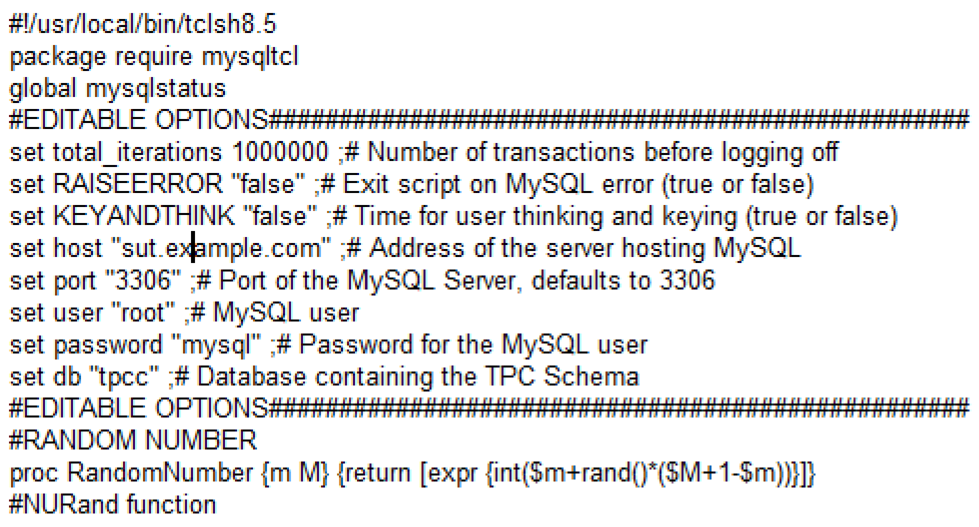
Figure 23 Driver Options

Driver Options

Under the Driver Options section you have the following choices:

TPC-C Driver Script

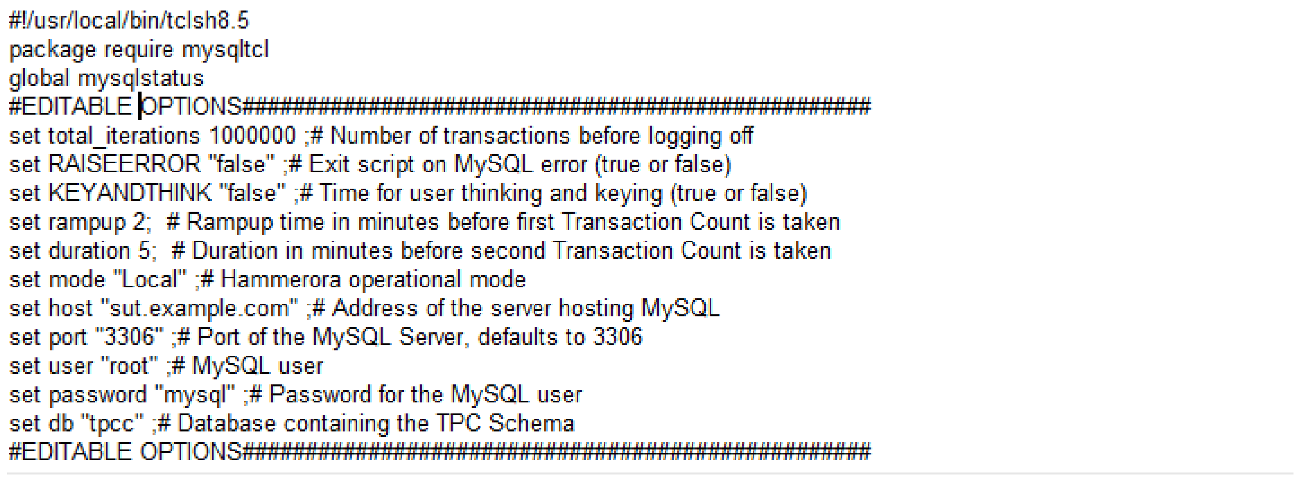
Under TPC-C Driver script you have the option of choosing either the Standard Driver Script or the Timed Test Driver Script. This choice will dynamically change the Driver Script that is loaded when the TPC-C Driver Script menu option is chosen. The Standard Driver Script as shown in figure 24 is a script 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 users 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.



```
#!/usr/local/bin/tclsh8.5
package require mysqltcl
global mysqlstatus
#EDITABLE OPTIONS#####################################################
set total_iterations 1000000 ;# Number of transactions before logging off
set RAISEERROR "false" ;# Exit script on MySQL error (true or false)
set KEYANDTHINK "false" ;# Time for user thinking and keying (true or false)
set host "sut.example.com" ;# Address of the server hosting MySQL
set port "3306" ;# Port of the MySQL Server, defaults to 3306
set user "root" ;# MySQL user
set password "mysql" ;# Password for the MySQL user
set db "tpcc" ;# Database containing the TPC Schema
#EDITABLE OPTIONS#####################################################
#RANDOM NUMBER
proc RandomNumber {m M} {return [expr {int($m+rand()*($M+1-$m))}]}
#NURand function
```

Figure 24 Standard Driver Script

Instead of the Standard Driver Script you can select the Timed Test Driver Script. As shown in Figure 25 this produces a number of additional options. You should select the Timed Test Driver Script when you wish to run timed tests and have Hammerora 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.



```
#!/usr/local/bin/tclsh8.5
package require mysqltcl
global mysqlstatus
#EDITABLE OPTIONS##################################################
set total_iterations 1000000 ;# Number of transactions before logging off
set RAISEERROR "false" ;# Exit script on MySQL error (true or false)
set KEYANDTHINK "false" ;# Time for user thinking and keying (true or false)
set rampup 2;  # Rampup time in minutes before first Transaction Count is taken
set duration 5;  # Duration in minutes before second Transaction Count is taken
set mode "Local" ;# Hammerora operational mode
set host "sut.example.com" ;# Address of the server hosting MySQL
set port "3306" ;# Port of the MySQL Server, defaults to 3306
set user "root" ;# MySQL user
set password "mysql" ;# Password for the MySQL user
set db "tpcc" ;# Database containing the TPC Schema
#EDITABLE OPTIONS##################################################
```

Figure 25 AWR Snapshot Driver Script

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. 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 entireity. The total_iterations value is internal to the TPC-C driver script and determines the number of times the internal loop is iterated ie

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 Timed Test Driver Script as the test is timed you should ensure that the number of transactions is set to a suitably high vale 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 (now the default value from Hammerora 2.5) 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 MySQL Error

Exit on MySQL Error is shown as the parameter RAISEERROR in the Driver Script. RAISEERROR impacts the behaviour of an individual virtual user on detecting a MySQL error. If set to TRUE on detecting a MySQL error the user will report the error into the Hammerora 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. Without keying and thinking time you are likely to see peak performance at or around the number of cores/Hyper Threads on your Database Server.

Keying and thinking time is an integral part of an offical 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 offical 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 of vritual 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 exhbit a massive 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 Ramup 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 snapshot and timing the test. The rampup time should be sufficiently long enough for a workload to reach a steady transaction rate before the first snapshot 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 snapshot 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 snapshots, if set to Slave no snapshots are taken. This is useful if multiple instances of Hammerora are running in Master and Slave mode to ensure that only one instance takes the snapshots.

The values of the MySQL Host, MySQL Port, MySQL User and MySQL Password detailed under the Build Options of the Schema Options page is shown as the host, port, user and password values in the Driver Script. This value defines the connectivity details for the user who owns the TPC-C schema. As long as you can connect to MySQL using exactly the same connection details then Hammerora will also be able to connect. If having difficulty connecting with Hammerora then troubleshoot with MySQL tools to resolve the connectivity issues.

MySQL Database

The Database detailed under the Build Options of the Schema Options page is shown as the db value in the Driver Script. The Database defines the database where the schema was created.

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

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 as shown in Figure 25.

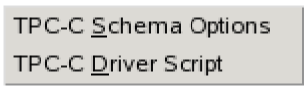
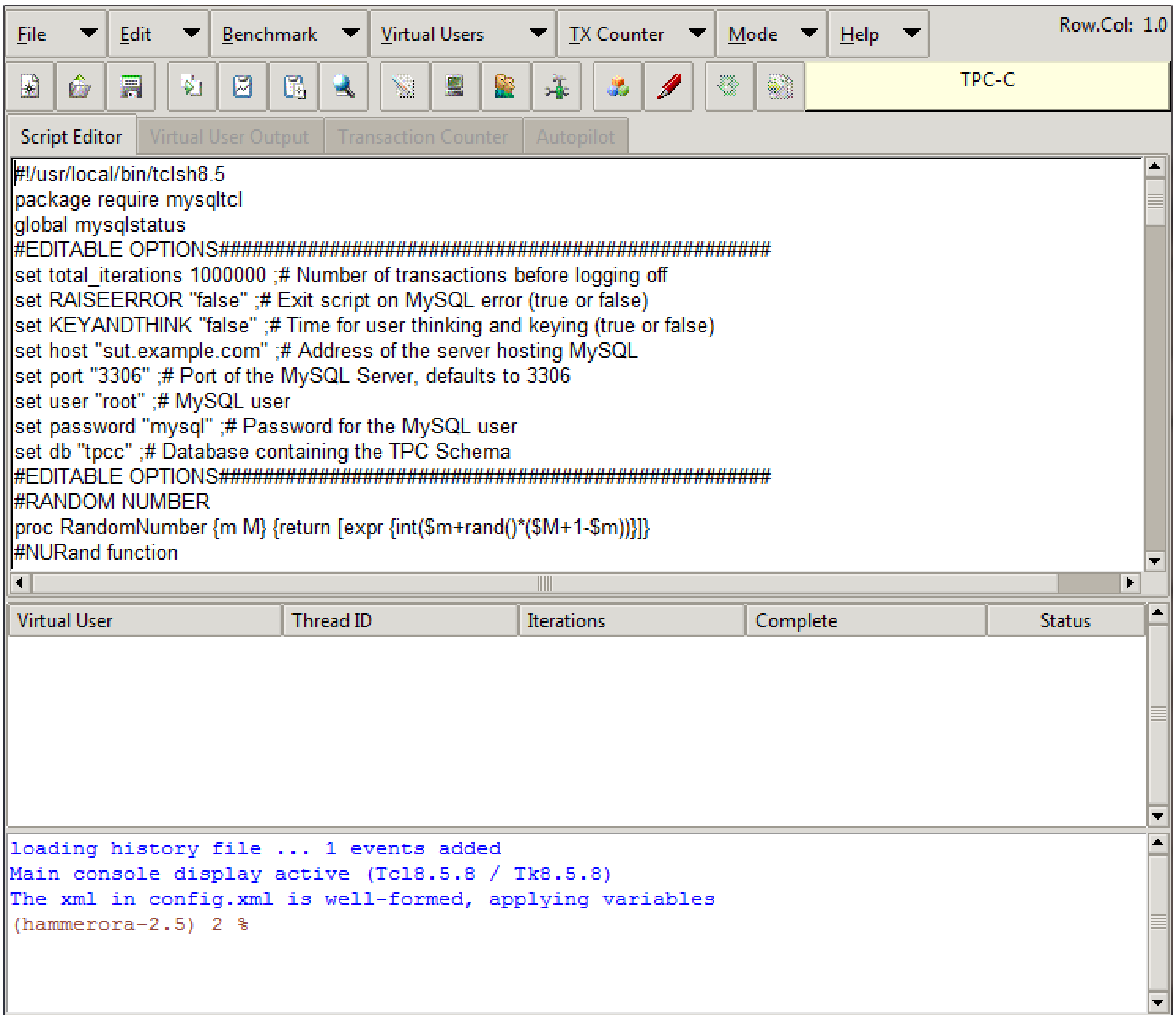


Figure 26 Select Driver Script

This will populate the Script Editor window with the driver script shown in Figure 24 or 25 according to whether the standard or AWR 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 Hammerora 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 27 shows a successfully loaded Standard Driver Script which provides a useful first test against a newly created TPC-C Schema.

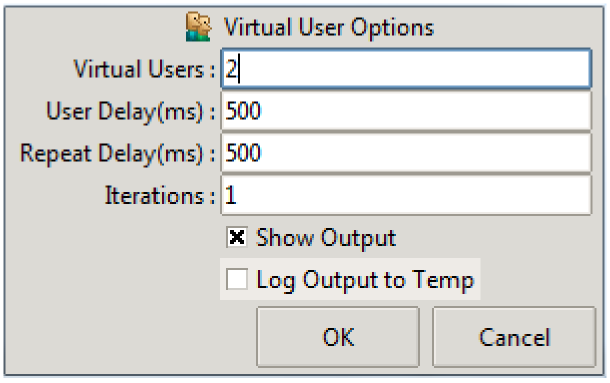


```
#!/usr/local/bin/tclsh8.5
package require mysqltcl
global mysqlstatus
#EDITABLE OPTIONS##################################################
set total_iterations 1000000 ;# Number of transactions before logging off
set RAISEERROR "false" ;# Exit script on MySQL error (true or false)
set KEYANDTHINK "false" ;# Time for user thinking and keying (true or false)
set host "sut.example.com" ;# Address of the server hosting MySQL
set port "3306" ;# Port of the MySQL Server, defaults to 3306
set user "root" ;# MySQL user
set password "mysql" ;# Password for the MySQL user
set db "tpcc" ;# Database containing the TPC Schema
#EDITABLE OPTIONS##################################################
#RANDOM NUMBER
proc RandomNumber {m M} {return [expr {int($m+rand()*($M+1-$m))}]}
#NURand function
```

```
loading history file ... 1 events added
Main console display active (Tcl8.5.8 / Tk8.5.8)
The xml in config.xml is well-formed, applying variables
(hammerora-2.5) 2 %
```

Figure 27 TPC-C Driver Script

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 Virtual Users menu as shown in Figure 28 select Vuser 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 Hammerora 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.


Virtual User Options

Virtual Users : 2

User Delay(ms) : 500

Repeat Delay(ms) : 500

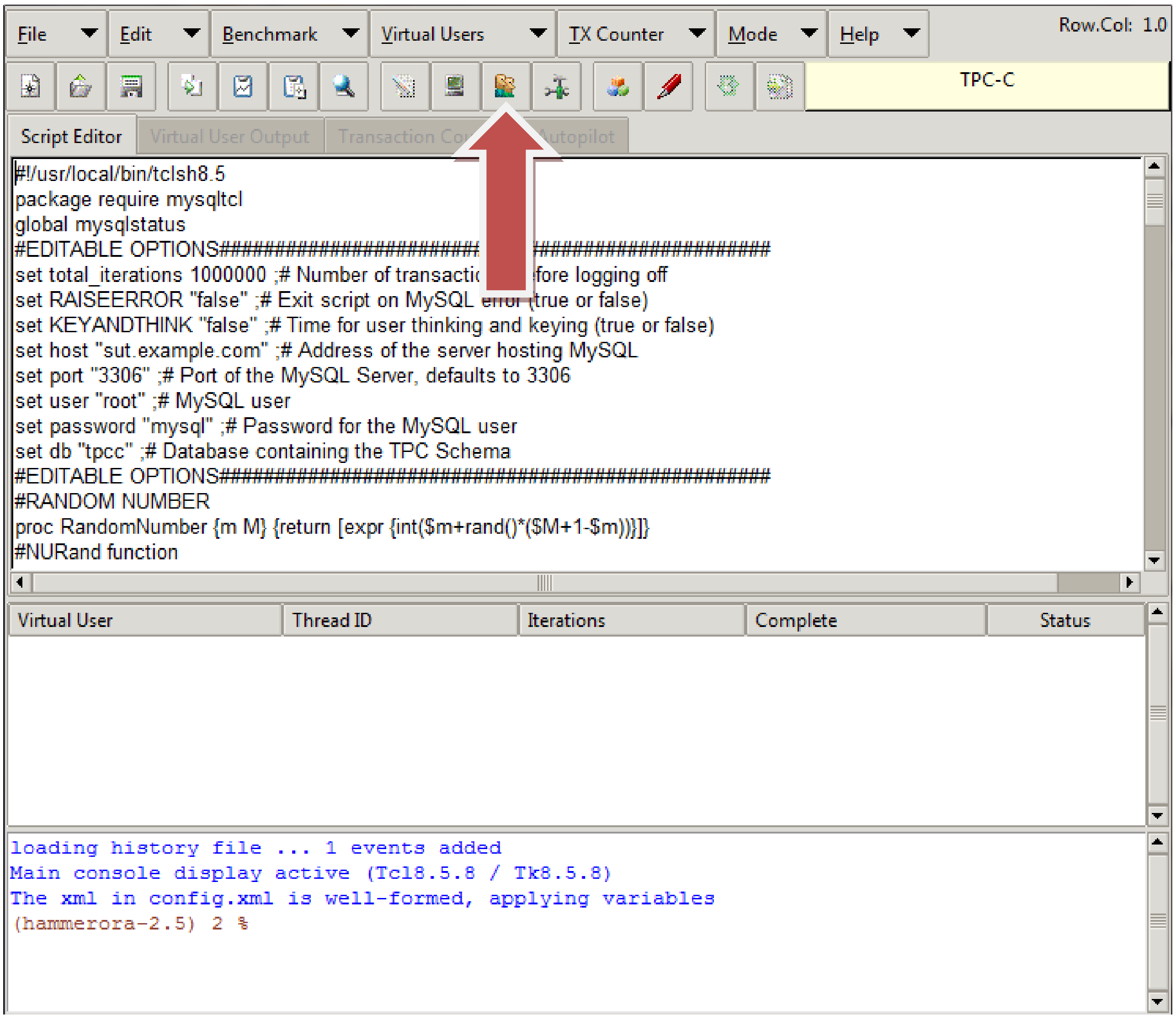
Iterations : 1

[✓] Show Output

[ ] Log Output to Temp

Figure 28 Select Virtual Users

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 Hammerora 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 as shown in Figure 29 to create the virtual users, they will be created but not start running yet.

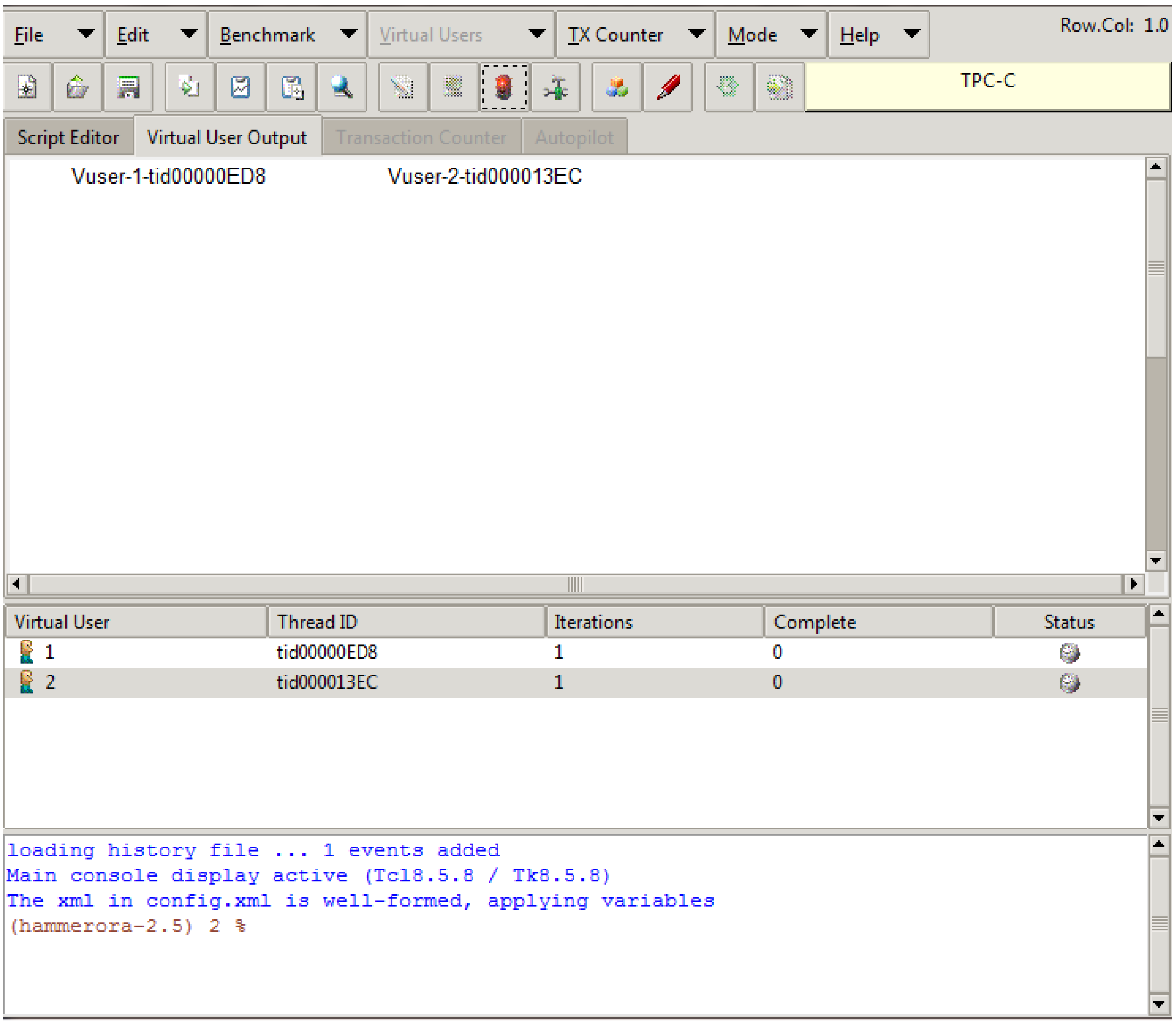


```
#!/usr/local/bin/tclsh8.5
package require mysqltcl
global mysqlstatus
#EDITABLE OPTIONS########################        ######################
set total_iterations 1000000 ;# Number of transactic        fore logging off
set RAISEERROR "false" ;# Exit script on MySQL error (true or false)
set KEYANDTHINK "false" ;# Time for user thinking and keying (true or false)
set host "sut.example.com" ;# Address of the server hosting MySQL
set port "3306" ;# Port of the MySQL Server, defaults to 3306
set user "root" ;# MySQL user
set password "mysql" ;# Password for the MySQL user
set db "tpcc" ;# Database containing the TPC Schema
#EDITABLE OPTIONS##################################################
#RANDOM NUMBER
proc RandomNumber {m M} {return [expr {int($m+rand()*($M+1-$m))}]}
#NURand function
```

```
loading history file ... 1 events added
Main console display active (Tcl8.5.8 / Tk8.5.8)
The xml in config.xml is well-formed, applying variables
(hammerora-2.5) 2 %
```

Figure 29 Create Virtual Users

You can observe as shown in Figure 30 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.

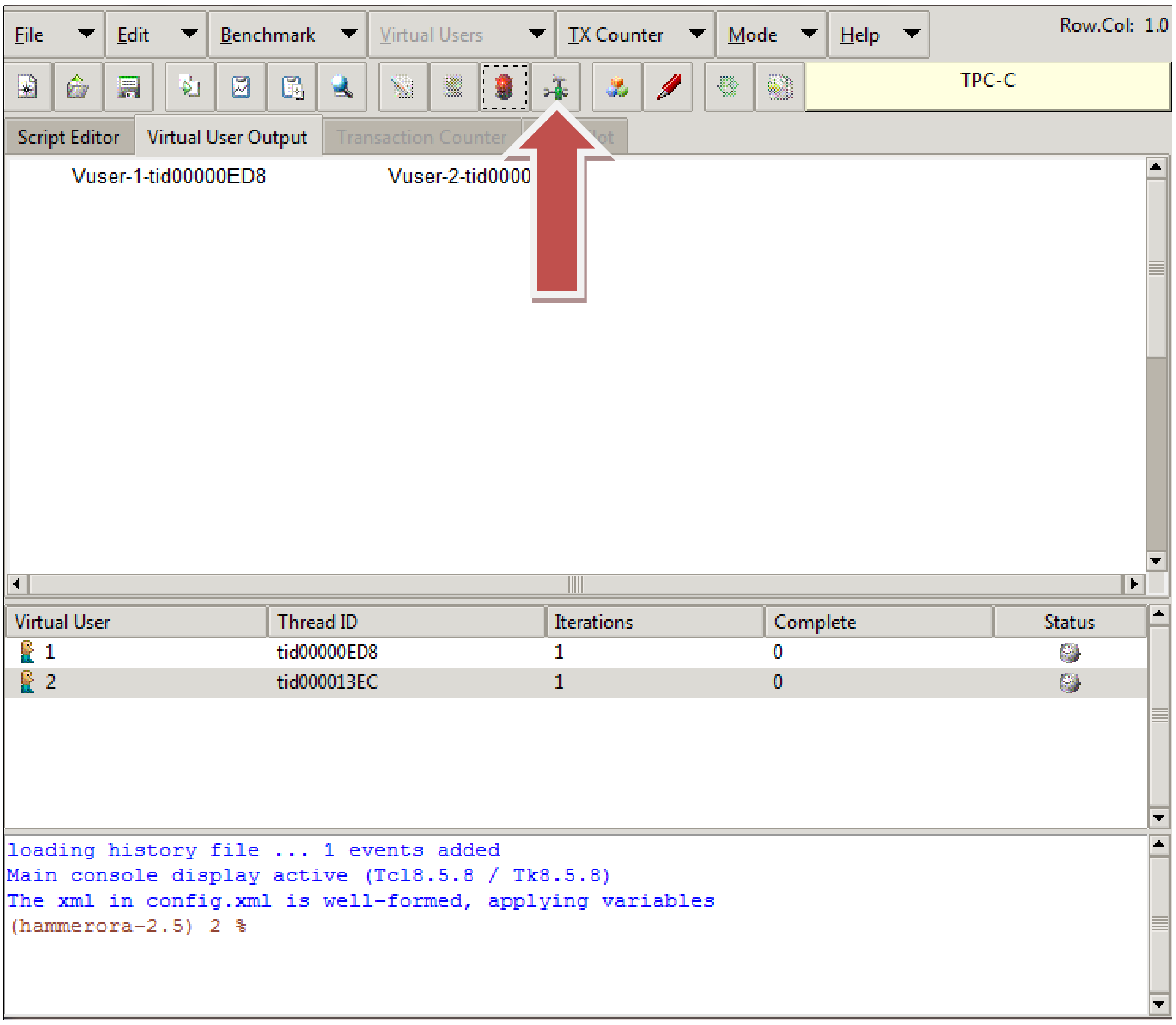


Virtual User	Thread ID	Iterations	Complete	Status
1	tid00000ED8	1	0	
2	tid000013EC	1	0	

```
loading history file ... 1 events added
Main console display active (Tcl8.5.8 / Tk8.5.8)
The xml in config.xml is well-formed, applying variables
(hammerora-2.5) 2 %
```

Figure 30 Virtual Users Created

To begin the test press the button Run Hammerora Load Test as shown in Figure 31, the name of the button will appear in the information pane.



```
Vuser-1-tid00000ED8          Vuser-2-tid0000
```

Virtual User	Thread ID	Iterations	Complete	Status
1	tid00000ED8	1	0	
2	tid000013EC	1	0	

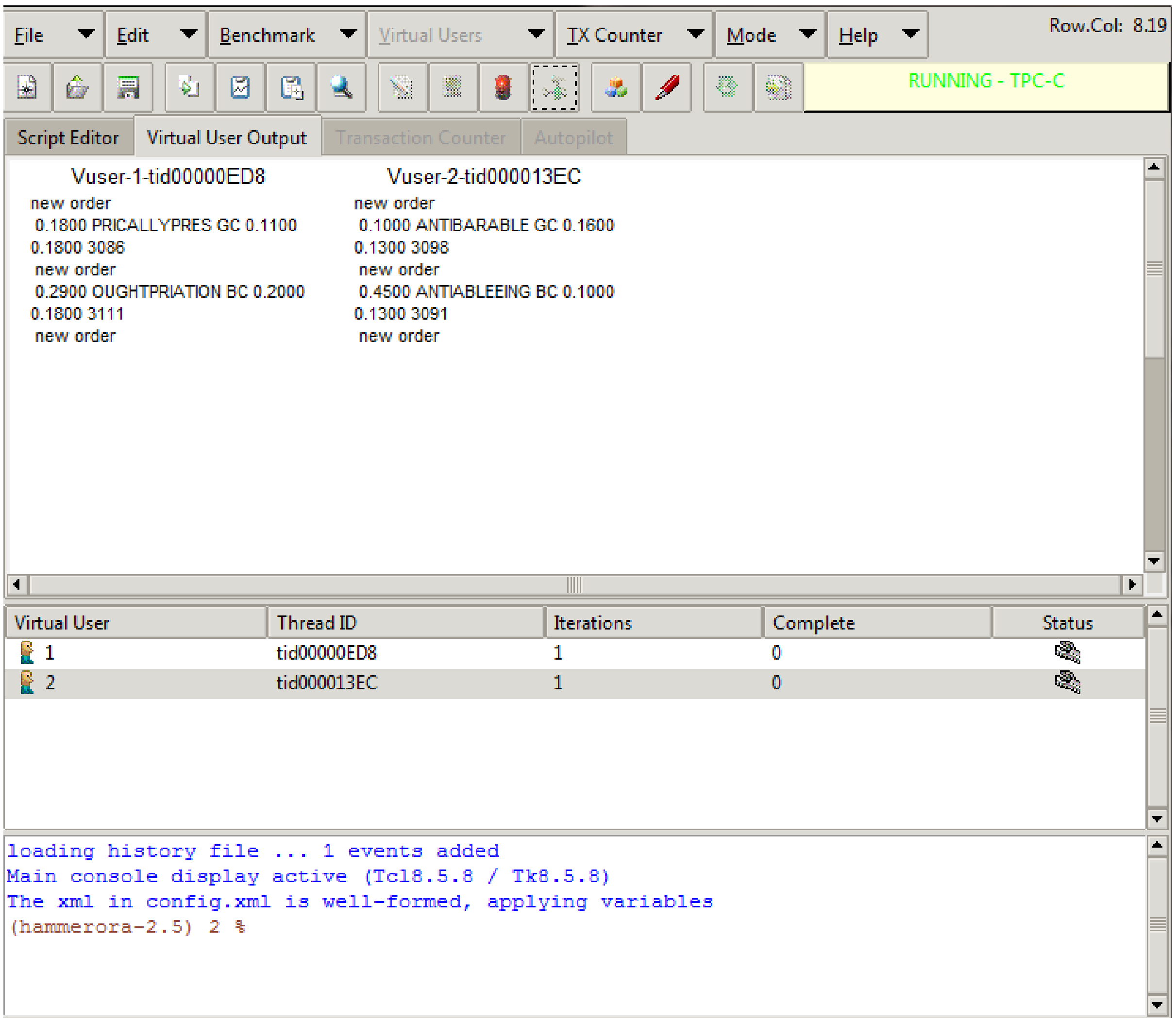
```
loading history file ... 1 events added
Main console display active (Tcl8.5.8 / Tk8.5.8)
The xml in config.xml is well-formed, applying variables
(hammerora-2.5) 2 %
```

Figure 31 Run Hammerora Load Test

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 process list for example

```
mysql> show processlist \G
*************************** 1. row ***************************
     Id: 3
   User: root
   Host: localhost:55069
     db: tpcc
Command: Query
   Time: 0
  State: NULL
   Info: show processlist
*************************** 2. row ***************************
     Id: 22
   User: root
   Host: test.example.com:53573
     db: tpcc
Command: Query
   Time: 0
  State: Updating
  Info: UPDATE warehouse SET w_ytd = w_ytd +  NAME_CONST('p_h_amount',287.00)
WHERE w_id =  NAME_CONST('p_w_
*************************** 3. row ***************************
     Id: 23
   User: root
   Host: test.example.com:53574
     db: tpcc
Command: Query
   Time: 0
  State: statistics
   Info: SELECT s_quantity, s_data, s_dist_01, s_dist_02, s_dist_03, s_dist_04,
s_dist_05, s_dist_06, s_dist_
3 rows in set (0.00 sec)
```

and are running transactions as can be observed in the Virtual User Output as shown in Figure 32.



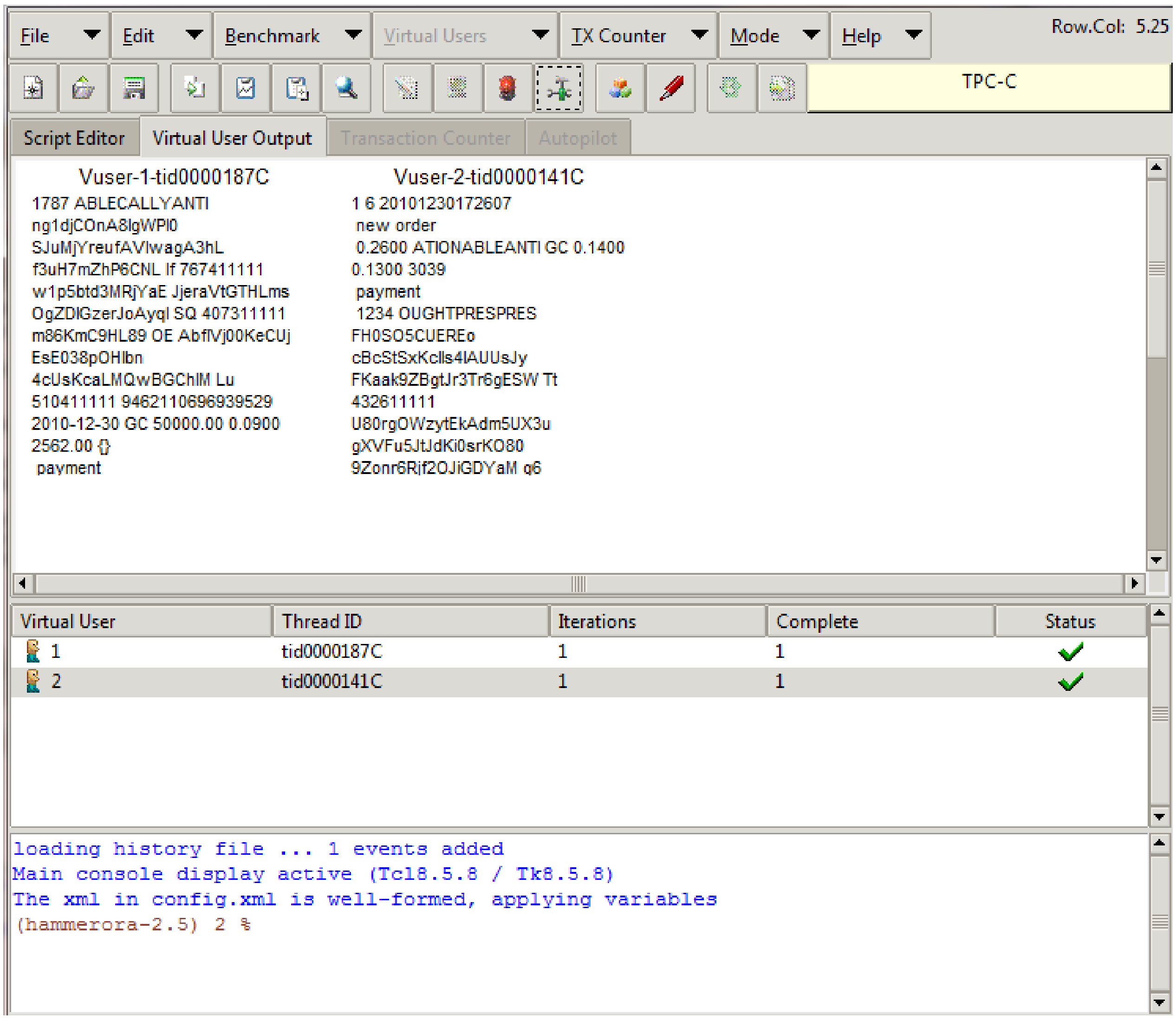
```
Vuser-1-tid00000ED8                     Vuser-2-tid000013EC
new order                               new order
 0.1800 PRICALLYPRES GC 0.1100           0.1000 ANTIBARABLE GC 0.1600
0.1800 3086                             0.1300 3098
 new order                               new order
 0.2900 OUGHTPRIATION BC 0.2000          0.4500 ANTIABLEEING BC 0.1000
0.1800 3111                             0.1300 3091
 new order                               new order
```

Virtual User	Thread ID	Iterations	Complete	Status
1	tid00000ED8	1	0	
2	tid000013EC	1	0	

```
loading history file ... 1 events added
Main console display active (Tcl8.5.8 / Tk8.5.8)
The xml in config.xml is well-formed, applying variables
(hammerora-2.5) 2 %
```

Figure 32 Load Testing Running

When the Virtual Users have completed all of their designated transactions they will exit showing a positive status as shown in Figure 33. Once the Virtual User is displaying this positive status it has logged off the database and will not be seen in V$SESSION. 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.



```
Vuser-1-tid0000187C
1787 ABLECALLYANTI
ng1djCOnA8lgWPl0
SJuMjYreufAVlwagA3hL
f3uH7mZhP6CNL lf 767411111
w1p5btd3MRjYaE JjeraVtGTHLms
OgZDlGzerJoAyql SQ 407311111
m86KmC9HL89 OE AbflVj00KeCUj
EsE038pOHlbn
4cUsKcaLMQwBGChlM Lu
510411111 9462110696939529
2010-12-30 GC 50000.00 0.0900
2562.00 {}
 payment
```

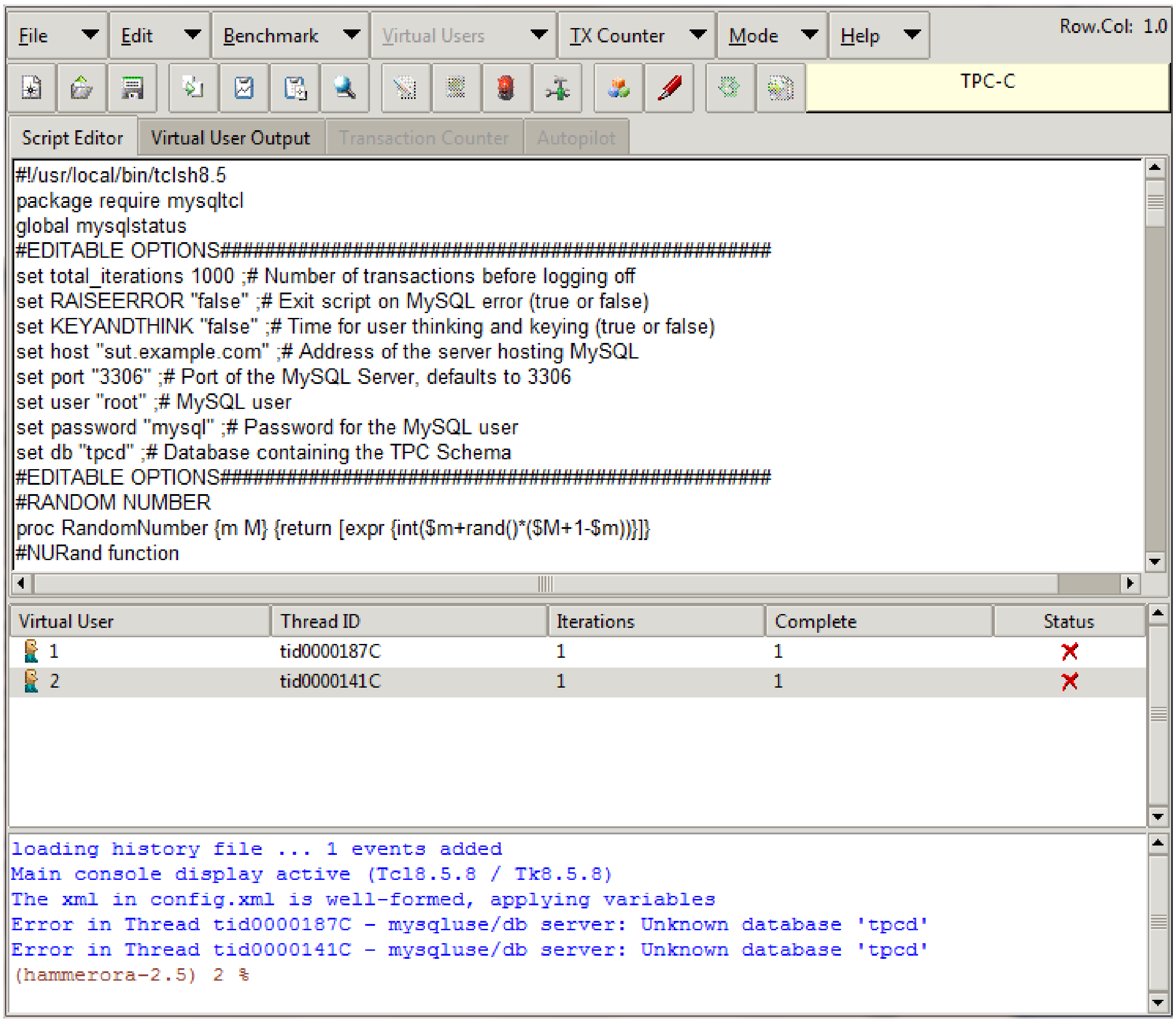
```
Vuser-2-tid0000141C
1 6 20101230172607
 new order
 0.2600 ATIONABLEANTI GC 0.1400
0.1300 3039
 payment
 1234 OUGHTPRESPRES
FH0SO5CUEREo
cBcStSxKclls4lAUUsJy
FKaak9ZBgtJr3Tr6gESW Tt
432611111
U80rgOWzytEkAdm5UX3u
gXVFu5JtJdKi0srKO80
9Zonr6Rjf2OJiGDYaM q6
```

Virtual User	Thread ID	Iterations	Complete	Status
1	tid0000187C	1	1	✓
2	tid0000141C	1	1	✓

```
loading history file ... 1 events added
Main console display active (Tcl8.5.8 / Tk8.5.8)
The xml in config.xml is well-formed, applying variables
(hammerora-2.5) 2 %
```

Figure 33 Virtual Users Complete

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 34 shows an example of an error, in this case it is an MySQL 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.



```
#!/usr/local/bin/tclsh8.5
package require mysqltcl
global mysqlstatus
#EDITABLE OPTIONS##################################################
set total_iterations 1000 ;# Number of transactions before logging off
set RAISEERROR "false" ;# Exit script on MySQL error (true or false)
set KEYANDTHINK "false" ;# Time for user thinking and keying (true or false)
set host "sut.example.com" ;# Address of the server hosting MySQL
set port "3306" ;# Port of the MySQL Server, defaults to 3306
set user "root" ;# MySQL user
set password "mysql" ;# Password for the MySQL user
set db "tpcd" ;# Database containing the TPC Schema
#EDITABLE OPTIONS##################################################
#RANDOM NUMBER
proc RandomNumber {m M} {return [expr {int($m+rand()*($M+1-$m))}]}
#NURand function
```

Virtual User	Thread ID	Iterations	Complete	Status
1	tid0000187C	1	1	✗
2	tid0000141C	1	1	✗

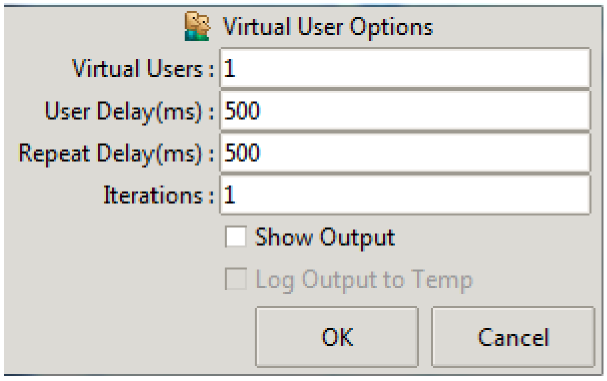
```
loading history file ... 1 events added
Main console display active (Tcl8.5.8 / Tk8.5.8)
The xml in config.xml is well-formed, applying variables
Error in Thread tid0000187C - mysqluse/db server: Unknown database 'tpcd'
Error in Thread tid0000141C - mysqluse/db server: Unknown database 'tpcd'
(hammerora-2.5) 2 %
```

Figure 34 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.

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 35. Configure One Virtual user without selecting the Show Output check box (The reason for suppressing output is described under Pre-Test 1).


Virtual User Options

Virtual Users : 1

User Delay(ms) : 500

Repeat Delay(ms) : 500

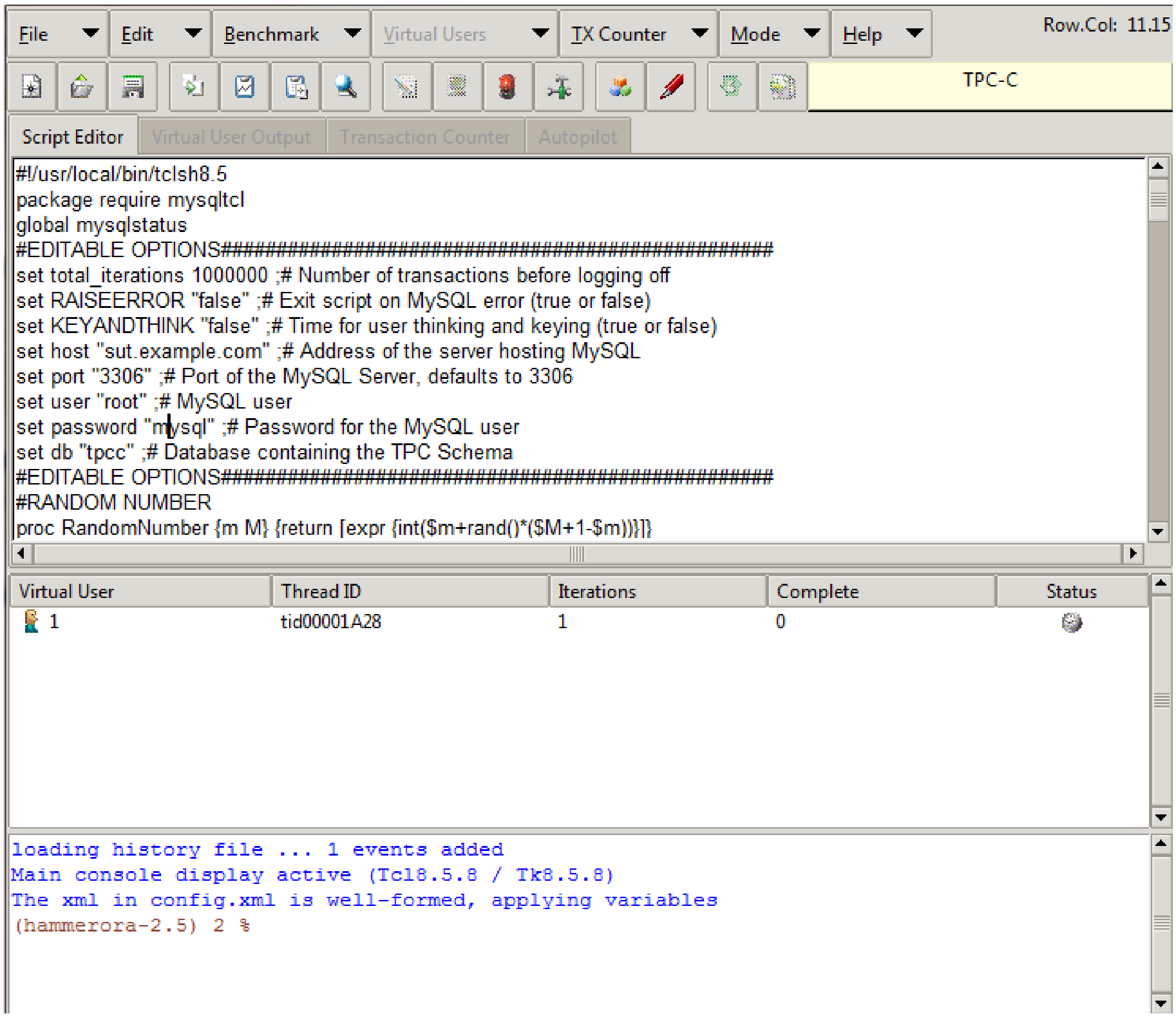
Iterations : 1

☐ Show Output

☐ Log Output to Temp

Figure 35 One Virtual User

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 as shown in Figure 29 will give you this single Virtual Configuration without specifically configuring the Virtual Users as shown in Figure 35. Figure 36 shows the single Virtual User created and the Standard Driver script loaded.



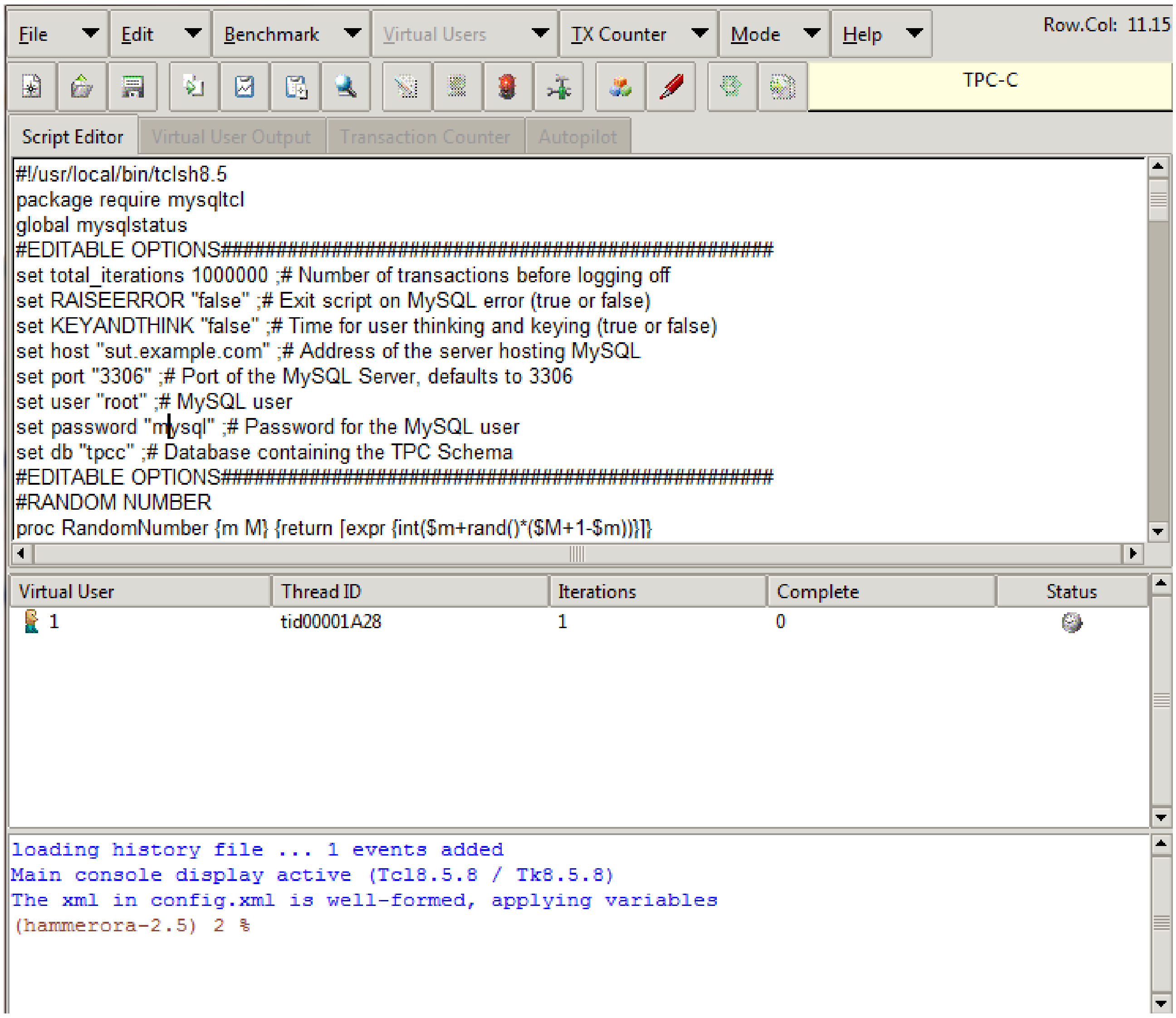
```
#!/usr/local/bin/tclsh8.5
package require mysqltcl
global mysqlstatus
#EDITABLE OPTIONS##################################################
set total_iterations 1000000 ;# Number of transactions before logging off
set RAISEERROR "false" ;# Exit script on MySQL error (true or false)
set KEYANDTHINK "false" ;# Time for user thinking and keying (true or false)
set host "sut.example.com" ;# Address of the server hosting MySQL
set port "3306" ;# Port of the MySQL Server, defaults to 3306
set user "root" ;# MySQL user
set password "mysql" ;# Password for the MySQL user
set db "tpcc" ;# Database containing the TPC Schema
#EDITABLE OPTIONS##################################################
#RANDOM NUMBER
proc RandomNumber {m M} {return [expr {int($m+rand()*($M+1-$m))}]}
```

Virtual User	Thread ID	Iterations	Complete	Status
1	tid00001A28	1	0	

```
loading history file ... 1 events added
Main console display active (Tcl8.5.8 / Tk8.5.8)
The xml in config.xml is well-formed, applying variables
(hammerora-2.5) 2 %
```

Figure 36 Single Virtual User

Press the Run Hammerora Load Test button as shown previously in Figure 31 to begin generating the Single User Throughput test. As shown in figure 37 the Virtual User icon has been updated to signify that the workload is running.



```
#!/usr/local/bin/tclsh8.5
package require mysqltcl
global mysqlstatus
#EDITABLE OPTIONS##################################################
set total_iterations 1000000 ;# Number of transactions before logging off
set RAISEERROR "false" ;# Exit script on MySQL error (true or false)
set KEYANDTHINK "false" ;# Time for user thinking and keying (true or false)
set host "sut.example.com" ;# Address of the server hosting MySQL
set port "3306" ;# Port of the MySQL Server, defaults to 3306
set user "root" ;# MySQL user
set password "mysql" ;# Password for the MySQL user
set db "tpcc" ;# Database containing the TPC Schema
#EDITABLE OPTIONS##################################################
#RANDOM NUMBER
proc RandomNumber {m M} {return [expr {int($m+rand()*($M+1-$m))}]}
```

Virtual User	Thread ID	Iterations	Complete	Status
1	tid00001A28	1	0	

```
loading history file ... 1 events added
Main console display active (Tcl8.5.8 / Tk8.5.8)
The xml in config.xml is well-formed, applying variables
(hammerora-2.5) 2 %
```

Figure 37 Single User Throughput Test

To observe performance during the test you can use the Transaction Counter. The Transaction Counter options can be selected from the TX Counter Menu as shown in Figure 38.

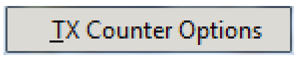
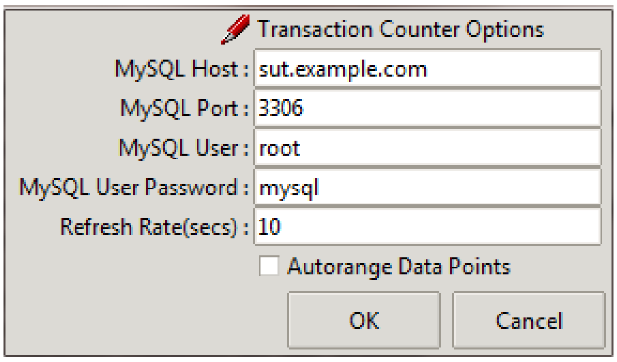


Figure 38 TX Counter Options

This displays the Transaction Counter Options as shown in Figure 39.



MySQL Host :	sut.example.com
MySQL Port :	3306
MySQL User :	root
MySQL User Password :	mysql
Refresh Rate(secs) :	10
☐	Autorange Data Points

Figure 39 Transaction Counter Options

Transaction Counter Options

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

MySQL Host/MySQL Port/MySQL User/MySQL User Password

The Connection details must be for a User with permission to run the “show global status command”, you can validate by logging on with this user and running the following command.

```
mysql> show global status where Variable_name = 'Handler_commit' or
Variable_name =  'Handler_rollback';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| Handler_commit   | 28689 |
| Handler_rollback | 4     |
+------------------+-------+
2 rows in set (0.00 sec)
```

Note that if the values shown are both 0 and continue to remain so whilst running a test then the transaction counter will not display the transaction rate during a test. This is most likely to occur if using a non-standard or experimental storage engine that fails to update its statistics correctly. In this eventuality please report the issue to the storage engine developers.

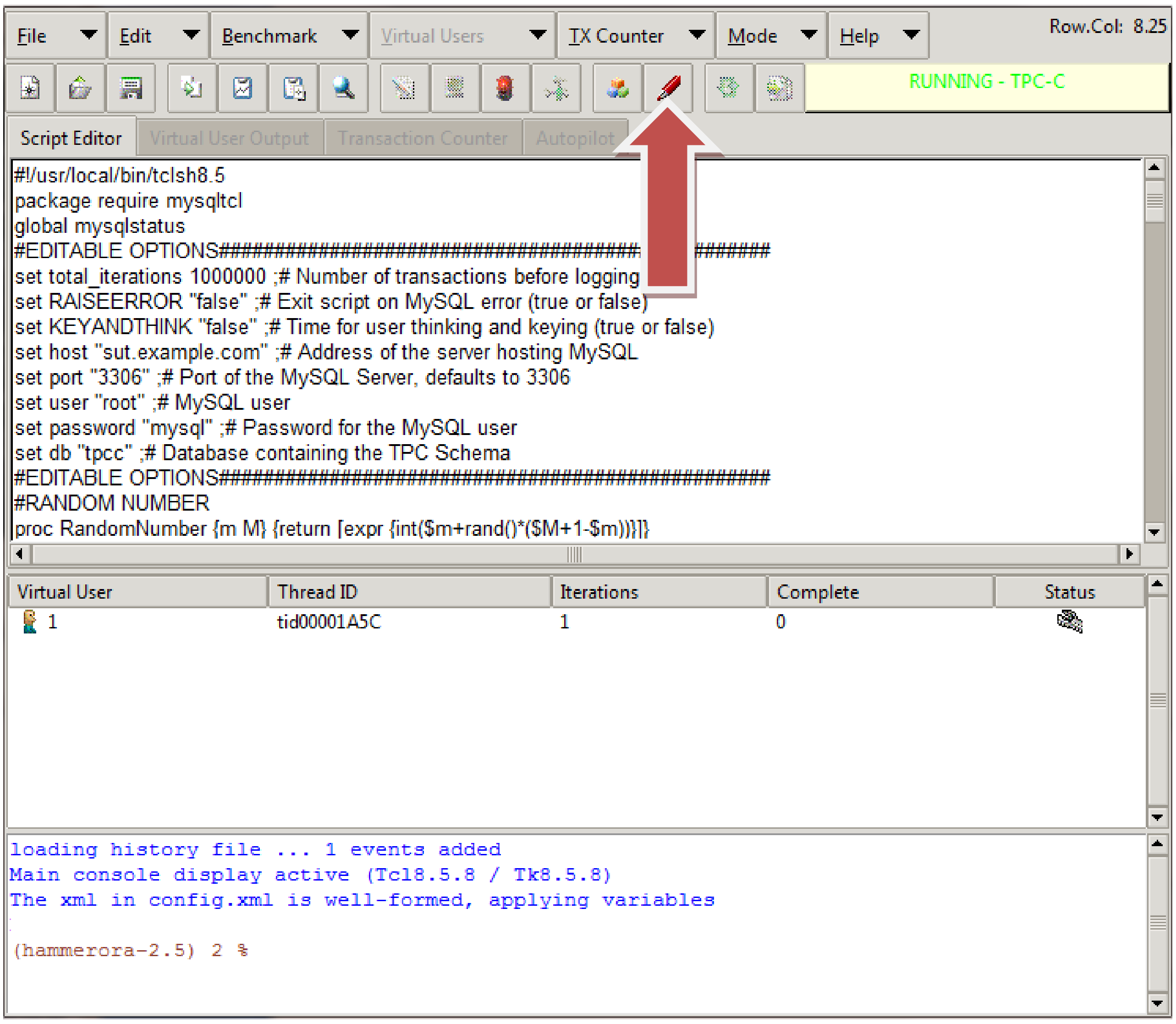
Refresh Rate

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 MySQL 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 40 to begin observing the transaction rate.



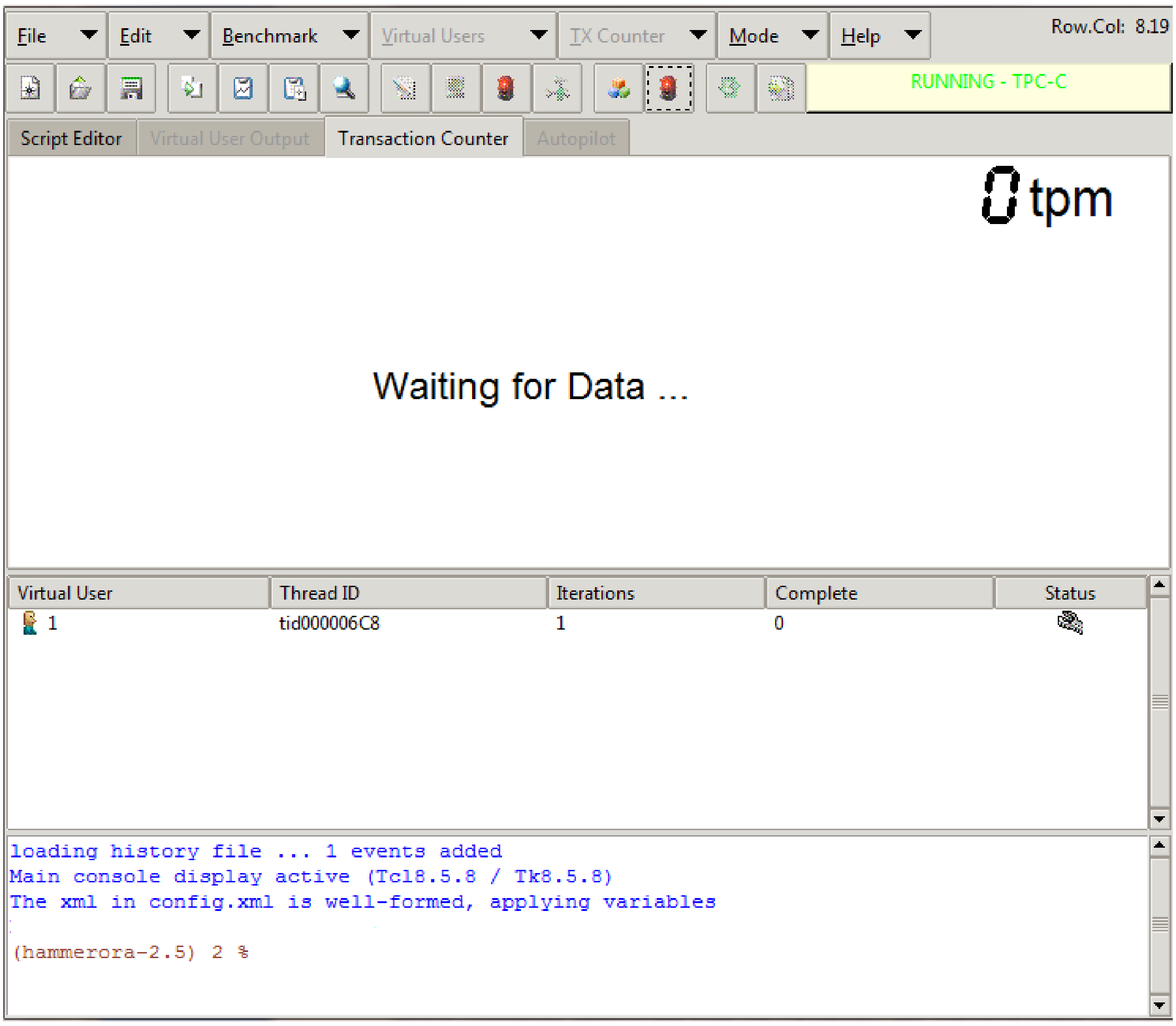
```
#!/usr/local/bin/tclsh8.5
package require mysqltcl
global mysqlstatus
#EDITABLE OPTIONS##################################################
set total_iterations 1000000 ;# Number of transactions before logging
set RAISEERROR "false" ;# Exit script on MySQL error (true or false)
set KEYANDTHINK "false" ;# Time for user thinking and keying (true or false)
set host "sut.example.com" ;# Address of the server hosting MySQL
set port "3306" ;# Port of the MySQL Server, defaults to 3306
set user "root" ;# MySQL user
set password "mysql" ;# Password for the MySQL user
set db "tpcc" ;# Database containing the TPC Schema
#EDITABLE OPTIONS##################################################
#RANDOM NUMBER
proc RandomNumber {m M} {return [expr {int($m+rand()*($M+1-$m))}]}
```

Virtual User	Thread ID	Iterations	Complete	Status
1	tid00001A5C	1	0	

```
loading history file ... 1 events added
Main console display active (Tcl8.5.8 / Tk8.5.8)
The xml in config.xml is well-formed, applying variables

(hammerora-2.5) 2 %
```

Figure 40 Start Transaction Counter

The transaction Counter will become active and start collecting throughput data as shown in figure 41.

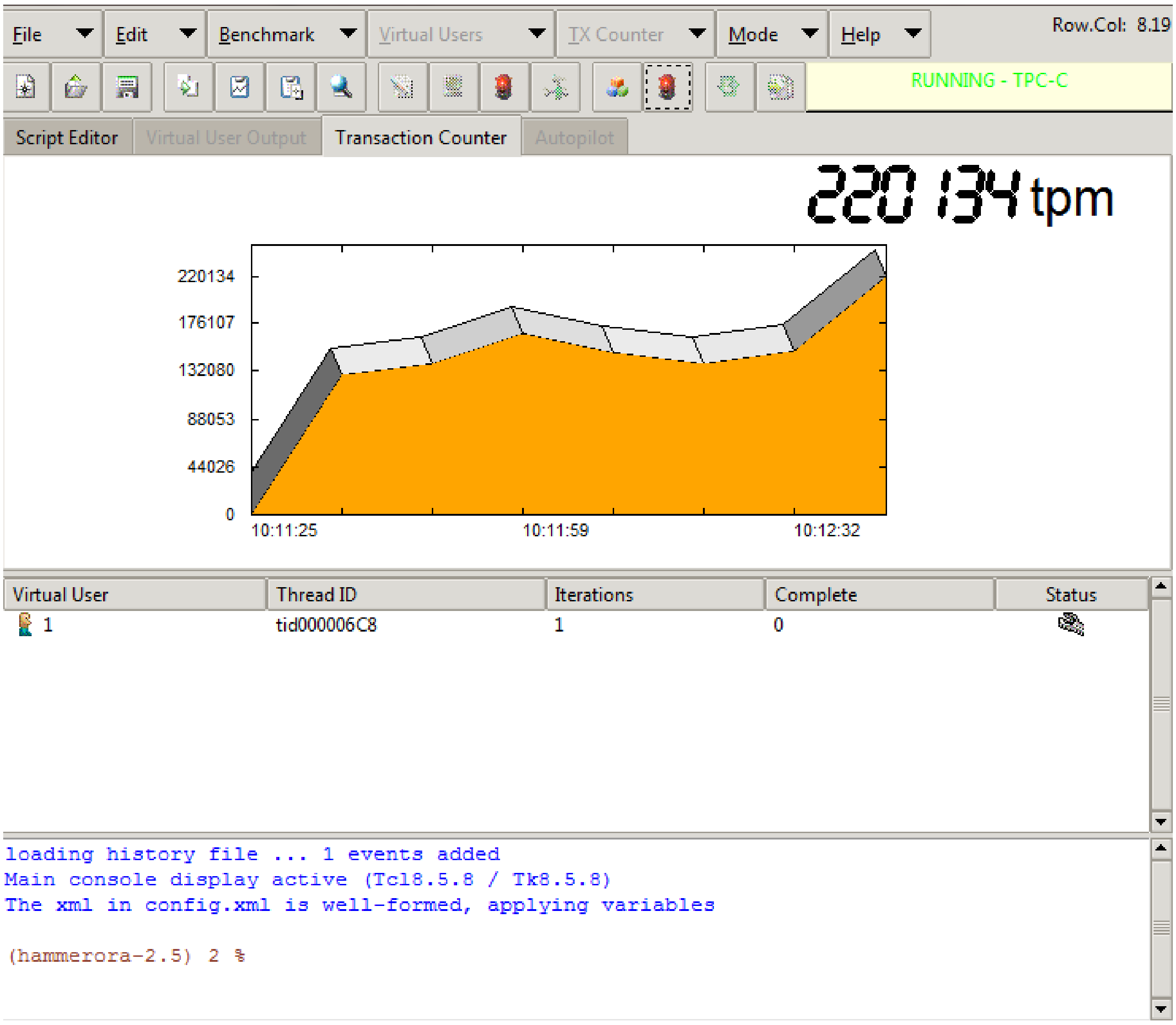


Virtual User	Thread ID	Iterations	Complete	Status
1	tid000006C8	1	0	

```
loading history file ... 1 events added
Main console display active (Tcl8.5.8 / Tk8.5.8)
The xml in config.xml is well-formed, applying variables

(hammerora-2.5) 2 %
```

Figure 41Waiting 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, however typically you should be looking for values in the hundreds of thousands. 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.

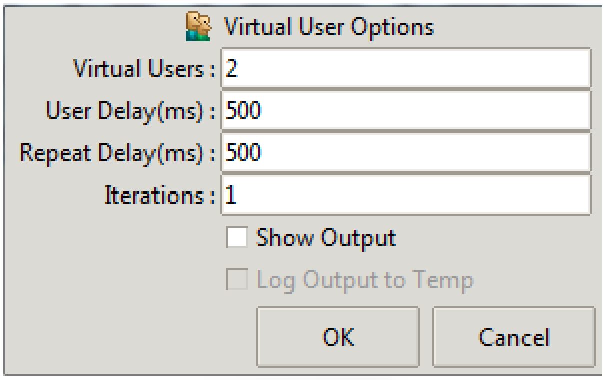


Virtual User	Thread ID	Iterations	Complete	Status
1	tid000006C8	1	0	

```
loading history file ... 1 events added
Main console display active (Tcl8.5.8 / Tk8.5.8)
The xml in config.xml is well-formed, applying variables

(hammerora-2.5) 2 %
```

Figure 42 Virtual user 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 43.

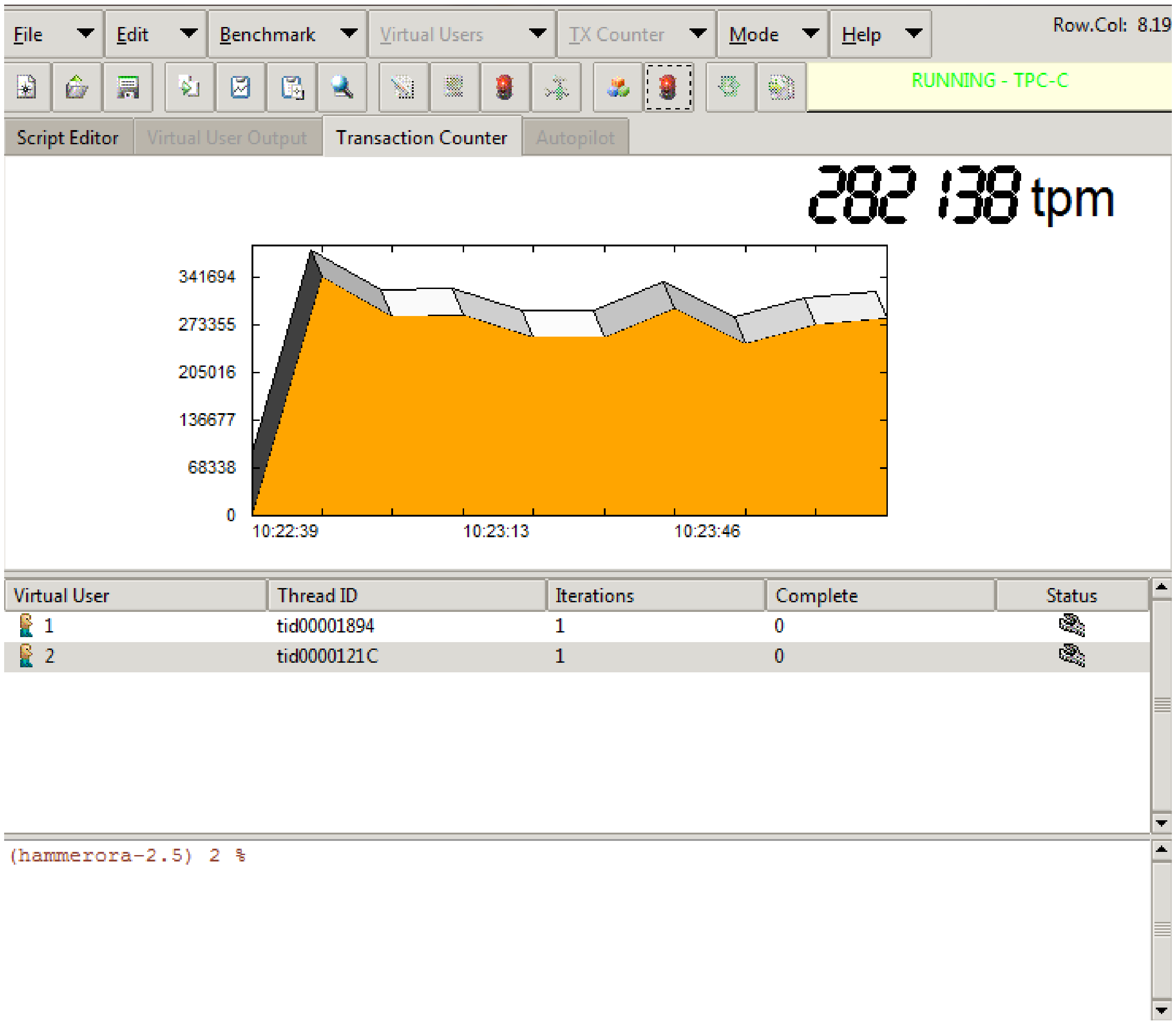

Virtual User Options

Virtual Users :	2
User Delay(ms) :	500
Repeat Delay(ms) :	500
Iterations :	1

☐ Show Output
☐ Log Output to Temp

Figure 43 Configuring Multiple Virtual Users

Similarly monitor the throughput for a higher number of Virtual Users as shown in Figure 44.



Virtual User	Thread ID	Iterations	Complete	Status
1	tid00001894	1	0	
2	tid0000121C	1	0	

```
(hammerora-2.5) 2 %
```

Figure 44 Running Multiple Virtual Users

Recommended Multiple Virtual Users for throughput testing are on an exponential scale from the single Virtual User test ie. 2,4,8,16,32 Virtual Users should be tested up to double the number of Cores or Hyper-Threads on the SUT Database Server. You should also not just test on an increasing scale. It is useful for example to run a single or two Virtual User test after running a test with Multiple Virtual Users to observe the importance of the caching of the data in the database Buffer Cache. This will be particularly noticeable with larger schemas. Figure 45 shows a typical performance profile of data being cached with a rising transaction rate through to a steady state.

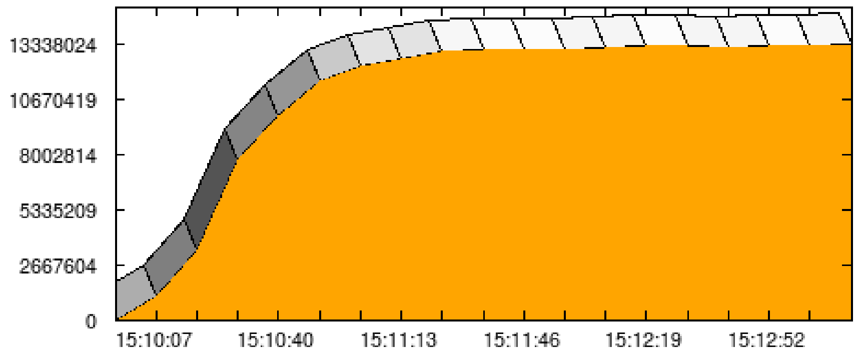


Figure 45 Memory Performance Plateau

Pre-testing is your opportunity to modify you configuration, tune your database and operating system and maximise the database throughput. Your aim should be ensure a consistent performance profile over a period of time such as that shown in Figure 46.

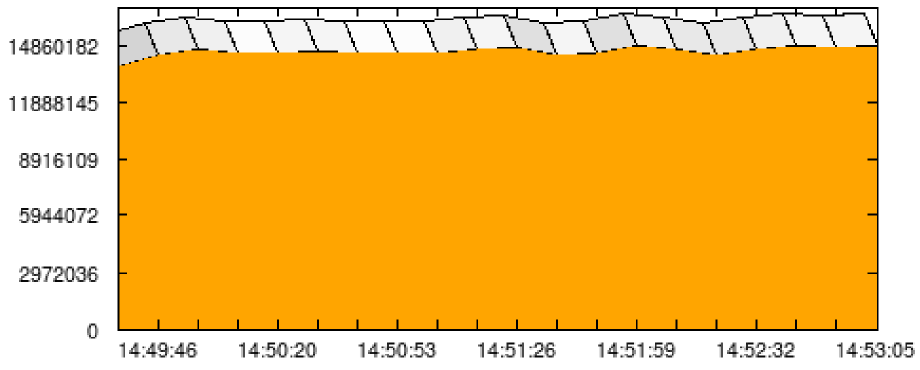


Figure 46 Steady Transaction State

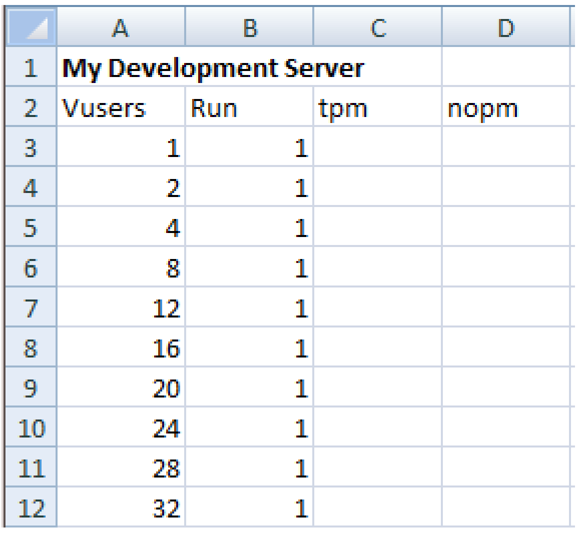
If the observed transaction rate has numerous peaks and troughs or the consistent throughput is lower than expected you should examine the system configuration to diagnose the reasons why performance is limited. To do this you can use MySQL provided or third party tools. You should also not neglect the relevant log files such as the Linux operating system logs and the MySQL error log. Once you have completed your pre-testing and are satisfied with your configuration you should move to planning and preparing to run a series of measured tests. You do not have to restart the database or rebuild the schema to conduct your performance tests. In fact having run a series of pre-tests and to have data resident in the buffer cache is the ideal starting point for conducting measured tests.

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 MySQL version and my.cnf 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 different storage engines or comparing drop in replacement databases with the MySQL upstream release.

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 MySQL 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 47.



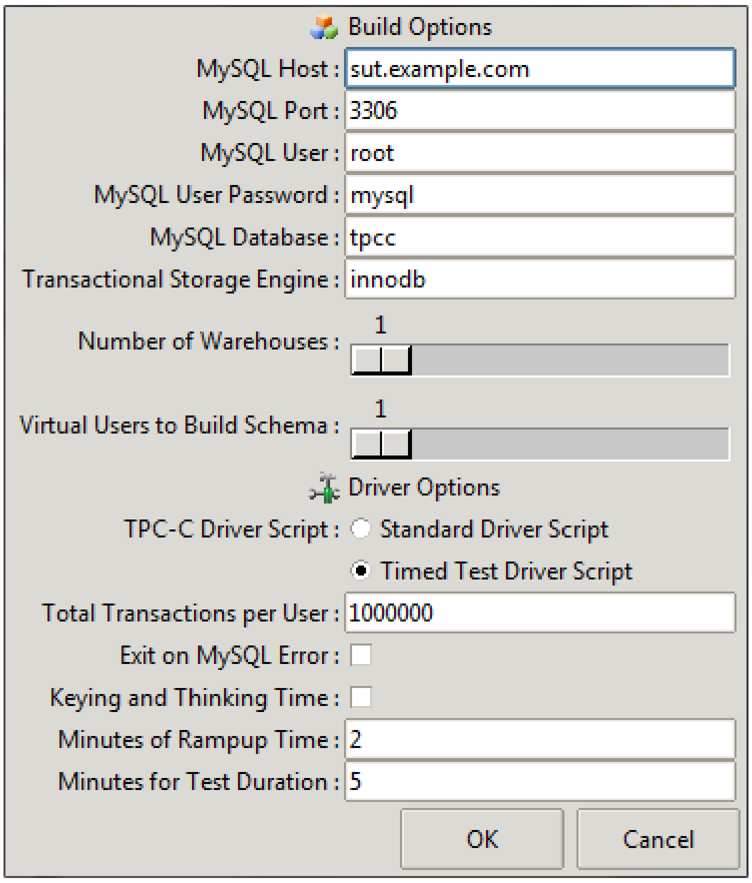
	A	B	C	D
1	**My Development Server**			
2	Vusers	Run	tpm	nopm
3	1	1		
4	2	1		
5	4	1		
6	8	1		
7	12	1		
8	16	1		
9	20	1		
10	24	1		
11	28	1		
12	32	1		

Figure 47 Planning Spreadsheet

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 Schema Options Window as described previously in this guide. Your focus is upon the Driver Options in this Window, and 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 48, select the Timed Test Driver Script radio button.



Build Options

MySQL Host :	sut.example.com
MySQL Port :	3306
MySQL User :	root
MySQL User Password :	mysql
MySQL Database :	tpcc
Transactional Storage Engine :	innodb
Number of Warehouses :	1
Virtual Users to Build Schema :	1

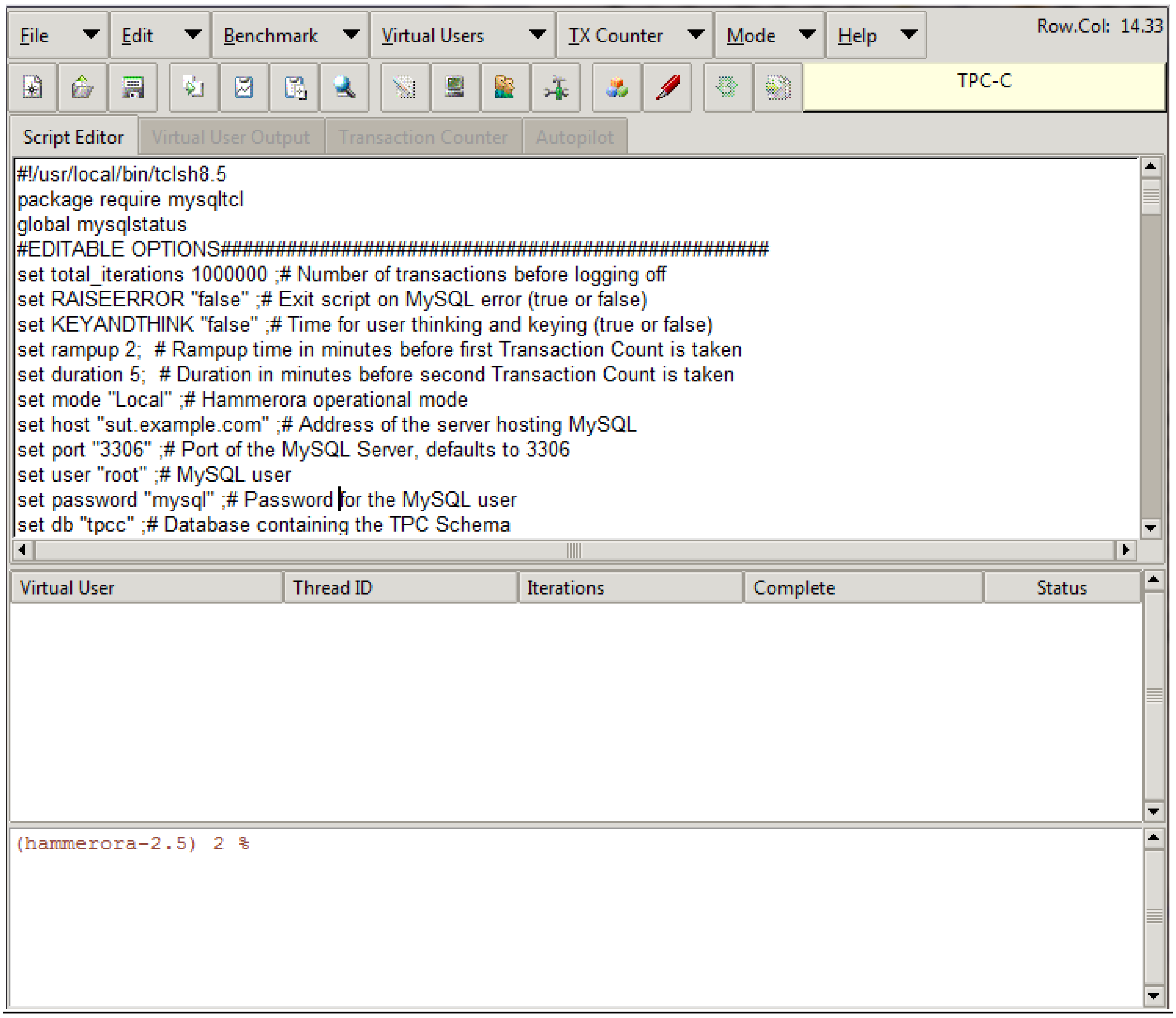
Driver Options

TPC-C Driver Script :	Standard Driver Script [✓ Timed Test Driver Script]
Total Transactions per User :	1000000
Exit on MySQL Error :	☐
Keying and Thinking Time :	☐
Minutes of Rampup Time :	2
Minutes for Test Duration :	5

Figure 48 Timed Test Driver Options

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 49 with the Timed Test Driver Script configured with your chosen options.

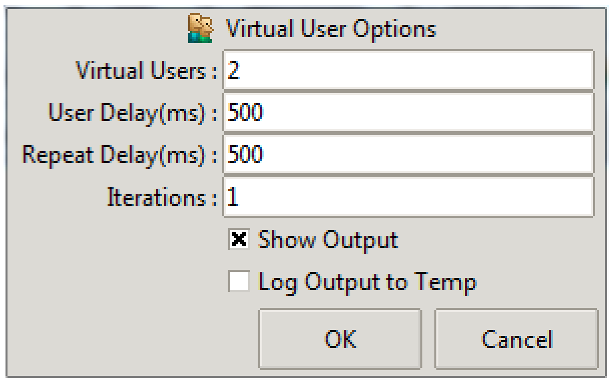


```
#!/usr/local/bin/tclsh8.5
package require mysqltcl
global mysqlstatus
#EDITABLE OPTIONS##################################################
set total_iterations 1000000 ;# Number of transactions before logging off
set RAISEERROR "false" ;# Exit script on MySQL error (true or false)
set KEYANDTHINK "false" ;# Time for user thinking and keying (true or false)
set rampup 2;  # Rampup time in minutes before first Transaction Count is taken
set duration 5;  # Duration in minutes before second Transaction Count is taken
set mode "Local" ;# Hammerora operational mode
set host "sut.example.com" ;# Address of the server hosting MySQL
set port "3306" ;# Port of the MySQL Server, defaults to 3306
set user "root" ;# MySQL user
set password "mysql" ;# Password for the MySQL user
set db "tpcc" ;# Database containing the TPC Schema
```

```
(hammerora-2.5) 2 %
```

Figure 49 AWR Snapshot Driver Script

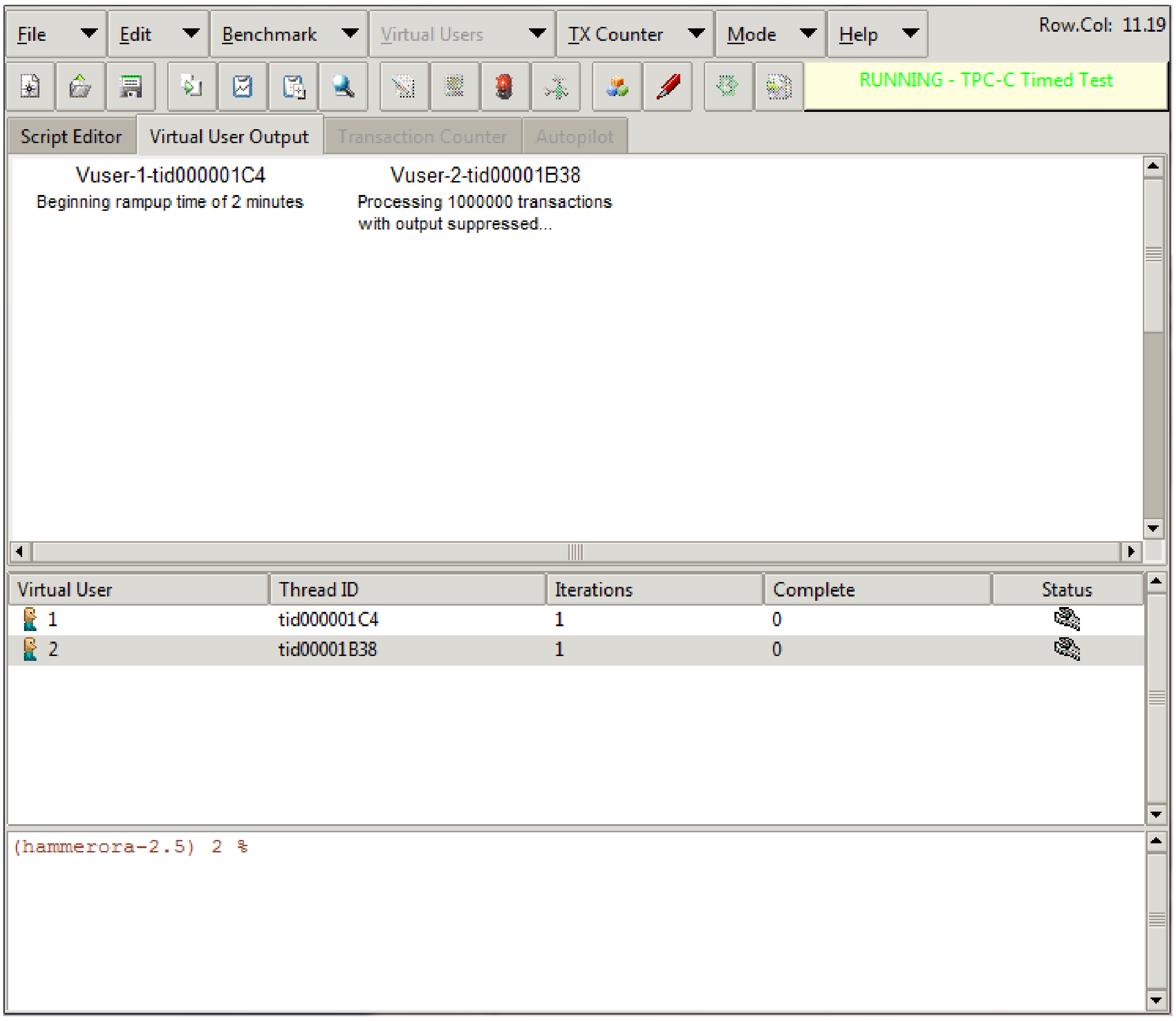
To change these options you can either change them in the Schema Options window and reload the driver script or more advanced users can also change them directly in the Driver Script itself. To run the Timed Test Driver Script you must configure the Virtual Users as you did with the Standard Driver Script 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 50.



Virtual Users : 2
User Delay(ms) : 500
Repeat Delay(ms) : 500
Iterations : 1
[✓ Show Output]
Log Output to Temp

Figure 50 AWR Snapshot Virtual Users

Click OK to save the configuration. Click the Create Virtual Users button as shown previously in this guide in Figure 29 to create the virtual users as shown in Figure 30 and Start the Virtual Users running as shown in Figure 32. Note that the Virtual User output is now different as shown in Figure 51.



```
Vuser-1-tid000001C4
Beginning rampup time of 2 minutes

Vuser-2-tid00001B38
Processing 1000000 transactions
with output suppressed...
```

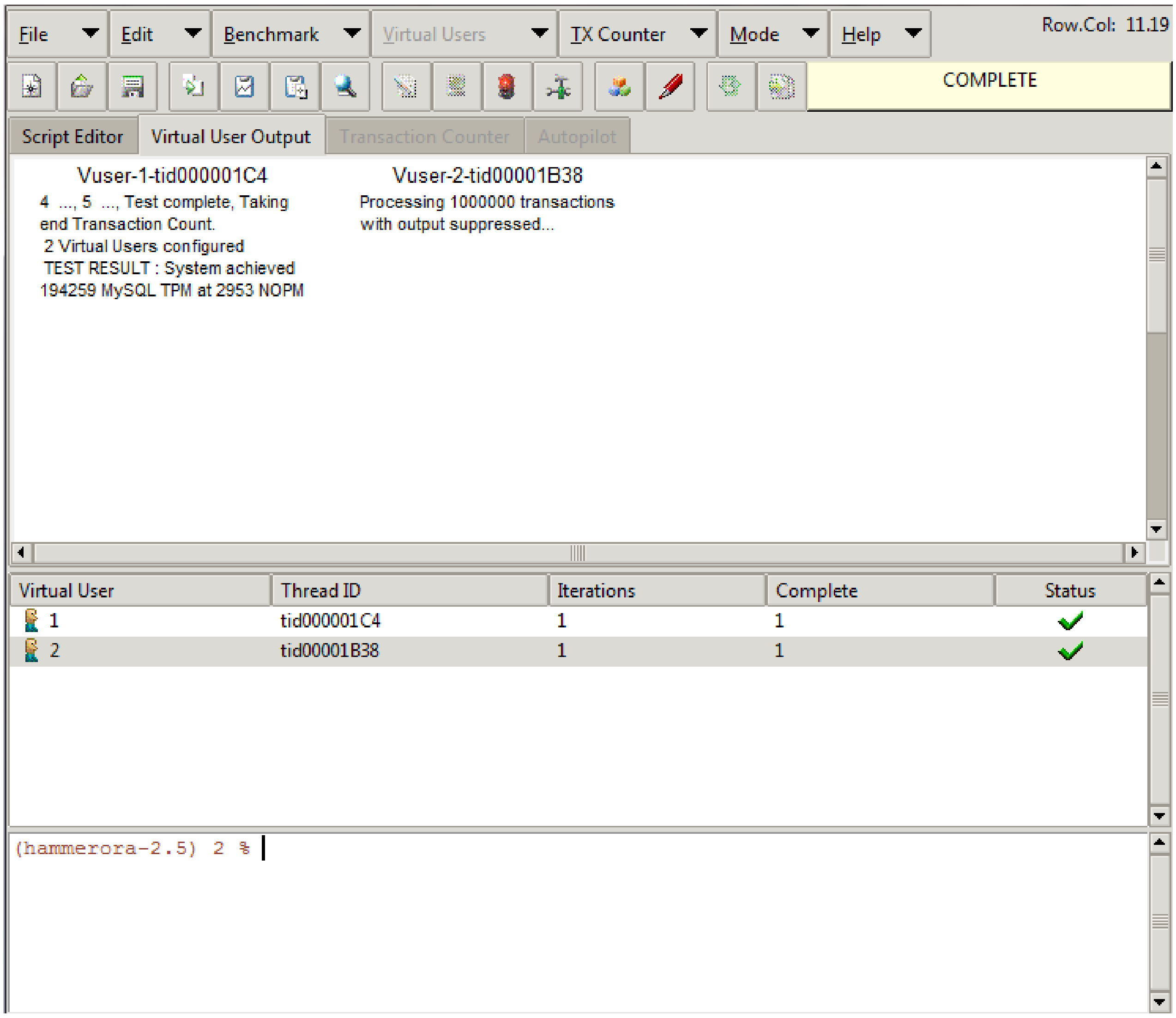
Virtual User	Thread ID	Iterations	Complete	Status
1	tid000001C4	1	0	
2	tid00001B38	1	0	

```
(hammerora-2.5) 2 %
```

Figure 51 AWR Driver Script Running

The output shows that rather than reporting the outcome of every transaction the worker Virtual User in this example Vuser-2-tid000001B38 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 User Delay(ms) value in the virtual user options can on some systems prevent a "login storm" and have all users logged on and processing tranasctions more quickly. 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 Vuser-1-tid000001C4 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 52.



```
Vuser-1-tid000001C4
4 ..., 5 ..., Test complete, Taking
end Transaction Count.
 2 Virtual Users configured
 TEST RESULT : System achieved
194259 MySQL TPM at 2953 NOPM
```

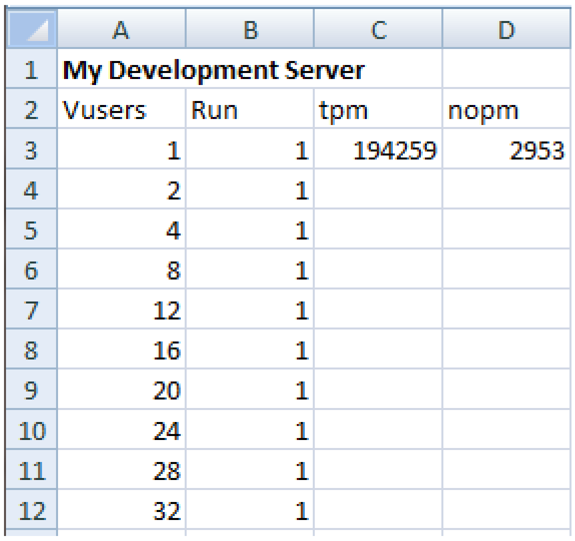
```
Vuser-2-tid00001B38
Processing 1000000 transactions
with output suppressed...
```

Virtual User	Thread ID	Iterations	Complete	Status
1	tid000001C4	1	1	✓
2	tid00001B38	1	1	✓

```
(hammerora-2.5) 2 %
```

Figure 52AWR Snapshot Result

The monitoring user reports the TEST RESULT of TPM and NOPM. TPM measures the number of MySQL 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 MySQL with an Oracle Database but NOPM can. When you have stopped the test enter your data into your reporting spreadsheet as shown in Figure 53.



	A	B	C	D
1	**My Development Server**			
2	Vusers	Run	tpm	nopm
3	1	1	194259	2953
4	2	1		
5	4	1		
6	8	1		
7	12	1		
8	16	1		
9	20	1		
10	24	1		
11	28	1		
12	32	1		

Figure 53 Updated Reporting Spreadsheet

Note that the Run column enables you to record multiple tests for the same number of Virtual Users. You should run two or three tests at the same number of Virtual Users and take the average value of all of the tests as your final value for the workload. 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 with the AWR Snapshot Driver Script up to the steps illustrated in Figures 49 and 50. 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 Mode menu as shown in Figure 54.

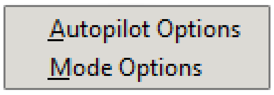
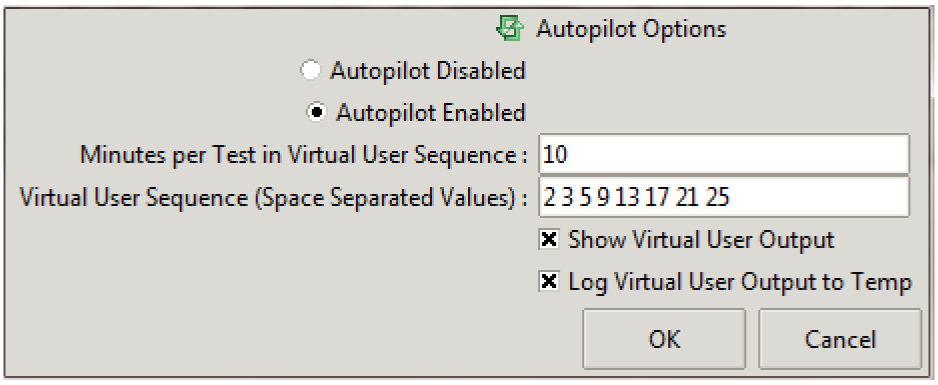


Figure 54 Autopilot menu

This shows the Autopilot Options menu as shown in Figure 55.



Autopilot Options

( ) Autopilot Disabled
(●) Autopilot Enabled

Minutes per Test in Virtual User Sequence : 10
Virtual User Sequence (Space Separated Values) : 2 3 5 9 13 17 21 25

[✓] Show Virtual User Output
[✓] Log Virtual User Output to Temp

Figure 55 Autopilot Options

shown in Figure 55. 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 given in the AWR Snapshot options shown in Figure 48. 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)

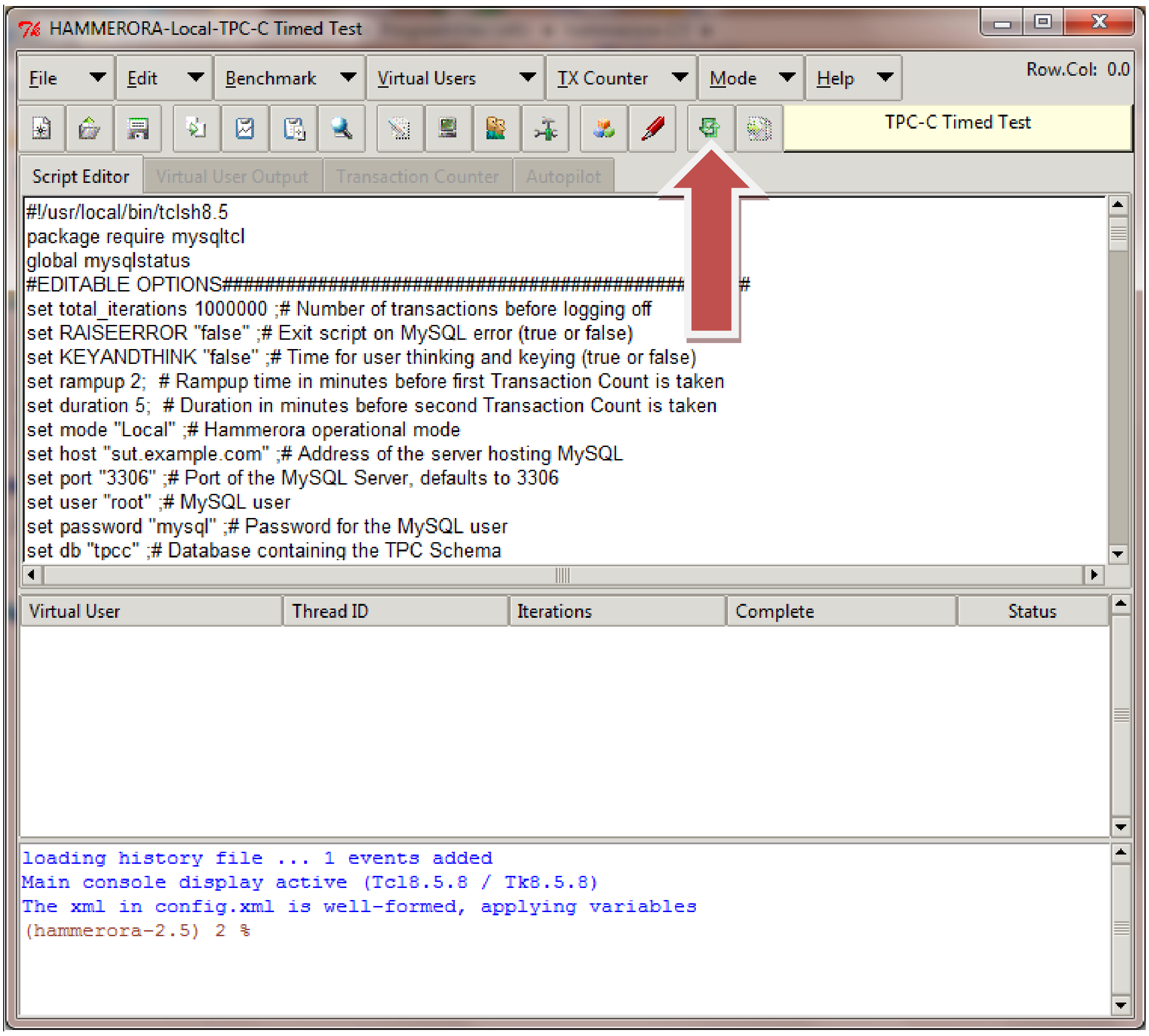
The Virtual User Sequence defines the number of Virtual Users to be configured in order for a sequence of tests separated by the Minutes for Test Duration. For example as shown in Figure 55, firstly a test with 2 Virtual Users will be run, then after 10 minutes a test with 3 Virtual Users will be run, 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 AWR Snapshot 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 record of the output of the tests that you run.

TIP: When running tests unattended with Autopilot Mode on MySQL you should always select the option Log Virtual User Output to Temp to ensure that you have a permanent record of the transaction results recorded by each test.

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 56.

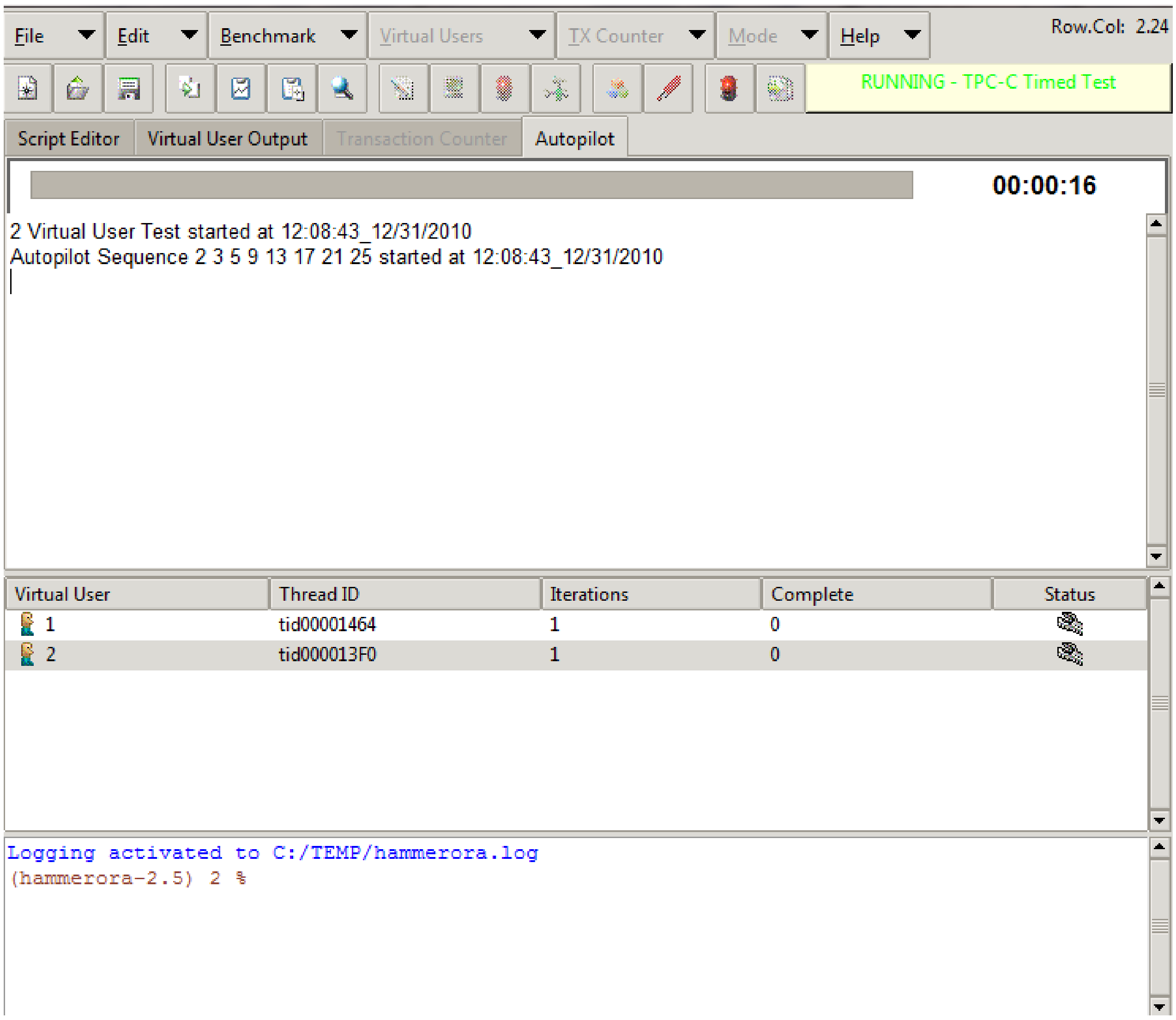


```
#!/usr/local/bin/tclsh8.5
package require mysqltcl
global mysqlstatus
#EDITABLE OPTIONS##################################################
set total_iterations 1000000 ;# Number of transactions before logging off
set RAISEERROR "false" ;# Exit script on MySQL error (true or false)
set KEYANDTHINK "false" ;# Time for user thinking and keying (true or false)
set rampup 2;  # Rampup time in minutes before first Transaction Count is taken
set duration 5;  # Duration in minutes before second Transaction Count is taken
set mode "Local" ;# Hammerora operational mode
set host "sut.example.com" ;# Address of the server hosting MySQL
set port "3306" ;# Port of the MySQL Server, defaults to 3306
set user "root" ;# MySQL user
set password "mysql" ;# Password for the MySQL user
set db "tpcc" ;# Database containing the TPC Schema
```

```
loading history file ... 1 events added
Main console display active (Tcl8.5.8 / Tk8.5.8)
The xml in config.xml is well-formed, applying variables
(hammerora-2.5) 2 %
```

Figure 56 Start Autopilot

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 57 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.



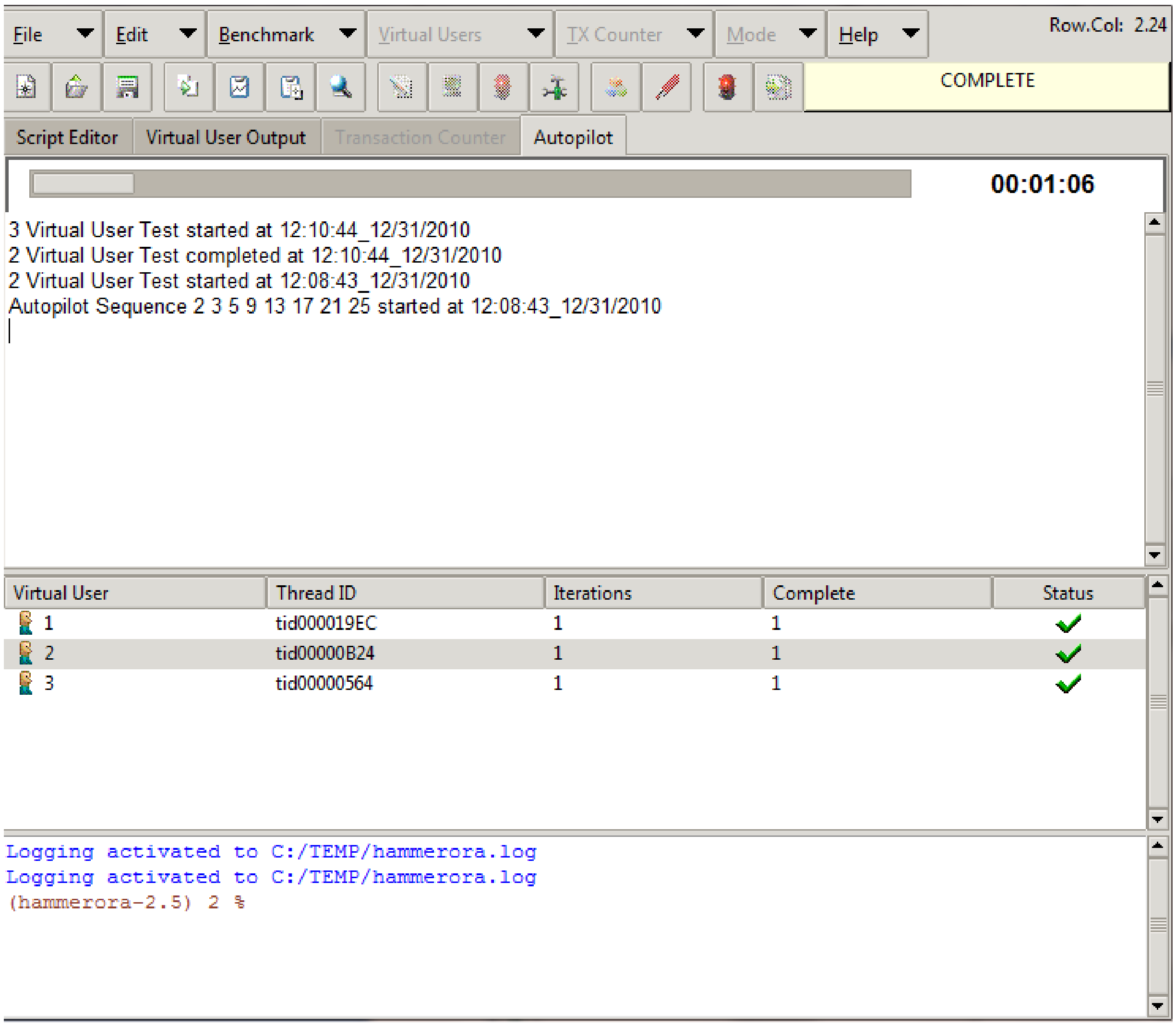
```
2 Virtual User Test started at 12:08:43_12/31/2010
Autopilot Sequence 2 3 5 9 13 17 21 25 started at 12:08:43_12/31/2010
```

Virtual User	Thread ID	Iterations	Complete	Status
1	tid00001464	1	0	
2	tid000013F0	1	0	

```
Logging activated to C:/TEMP/hammerora.log
(hammerora-2.5) 2 %
```

Figure 57 Autopilot Screen

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 58.



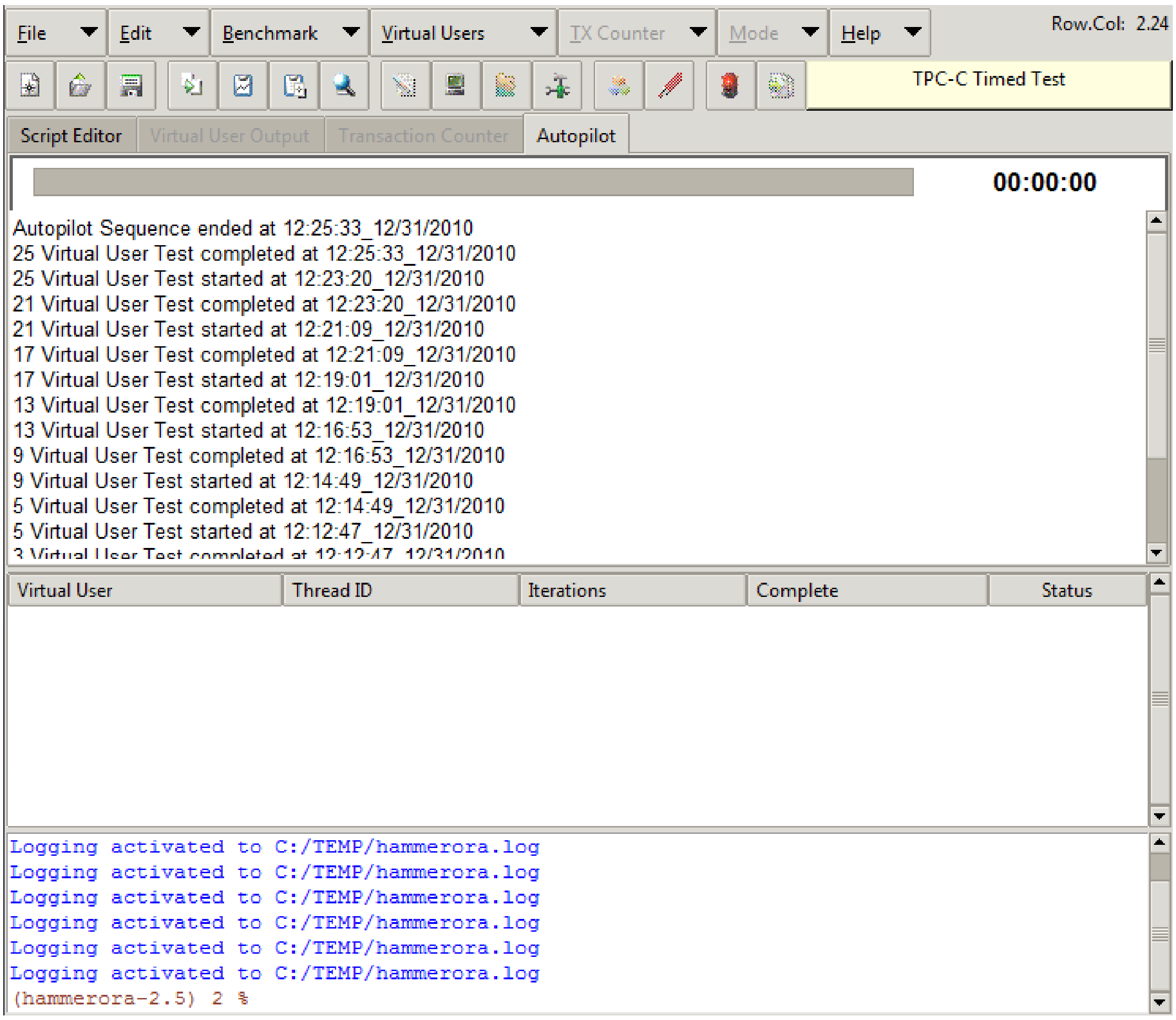
```
3 Virtual User Test started at 12:10:44_12/31/2010
2 Virtual User Test completed at 12:10:44_12/31/2010
2 Virtual User Test started at 12:08:43_12/31/2010
Autopilot Sequence 2 3 5 9 13 17 21 25 started at 12:08:43_12/31/2010
```

Virtual User	Thread ID	Iterations	Complete	Status
1	tid000019EC	1	1	✓
2	tid00000B24	1	1	✓
3	tid00000564	1	1	✓

```
Logging activated to C:/TEMP/hammerora.log
Logging activated to C:/TEMP/hammerora.log
(hammerora-2.5) 2 %
```

Figure 58 Autopilot Continuing

When your tests has completed as shown in Figure 59 you may retrieve your results from the logfile.



```
Autopilot Sequence ended at 12:25:33_12/31/2010
25 Virtual User Test completed at 12:25:33_12/31/2010
25 Virtual User Test started at 12:23:20_12/31/2010
21 Virtual User Test completed at 12:23:20_12/31/2010
21 Virtual User Test started at 12:21:09_12/31/2010
17 Virtual User Test completed at 12:21:09_12/31/2010
17 Virtual User Test started at 12:19:01_12/31/2010
13 Virtual User Test completed at 12:19:01_12/31/2010
13 Virtual User Test started at 12:16:53_12/31/2010
9 Virtual User Test completed at 12:16:53_12/31/2010
9 Virtual User Test started at 12:14:49_12/31/2010
5 Virtual User Test completed at 12:14:49_12/31/2010
5 Virtual User Test started at 12:12:47_12/31/2010
3 Virtual User Test completed at 12:12:47_12/31/2010
```

```
Logging activated to C:/TEMP/hammerora.log
Logging activated to C:/TEMP/hammerora.log
Logging activated to C:/TEMP/hammerora.log
Logging activated to C:/TEMP/hammerora.log
Logging activated to C:/TEMP/hammerora.log
Logging activated to C:/TEMP/hammerora.log
(hammerora-2.5) 2 %
```

Figure 59 Autopilot Complete

The Hammerora log file you will have a record of all the data of your sequence of tests. For example the following truncated and consolidated listing shows the lines from the report highlighting the number of Virtual Users configured for a test (remember that this value includes the monitoring user) and the test results.

```
…
tid0x434a6940:2 Virtual Users configured
tid0x434a6940:TEST RESULT : System achieved 617073 MySQL TPM at 9436 NOPM
tid0x42aa5940:3 Virtual Users configured
tid0x42aa5940:TEST RESULT : System achieved 1336528 MySQL TPM at 20459 NOPM
tid0x434a6940:5 Virtual Users configured
tid0x434a6940:TEST RESULT : System achieved 2766546 MySQL TPM at 42312 NOPM
tid0x43ea7940:9 Virtual Users configured
tid0x43ea7940:TEST RESULT : System achieved 5676422 MySQL TPM at 86747 NOPM
tid0x466ab940:13 Virtual Users configured
tid0x466ab940:TEST RESULT : System achieved 7790467 MySQL TPM at 119145 NOPM
…
```

When you have finished your test sequence press the traffic light icon to end Autopilot Mode.

Performance Analysis

When you have finished your sequence of tests you will be equipped with a spreadsheet containing a number of data points corresponding to an increasing data load.

Performance Comparisons

Use your spreadsheet to generate a graph of a performance profile with the TPM value for the y axis and the number of Virtual Users for the x axis.

The performance profile should resemble the figure as shown in Figure 60 with an increasing level of transactions as the number of Virtual Users increases up to a maximum point of system utilisation.

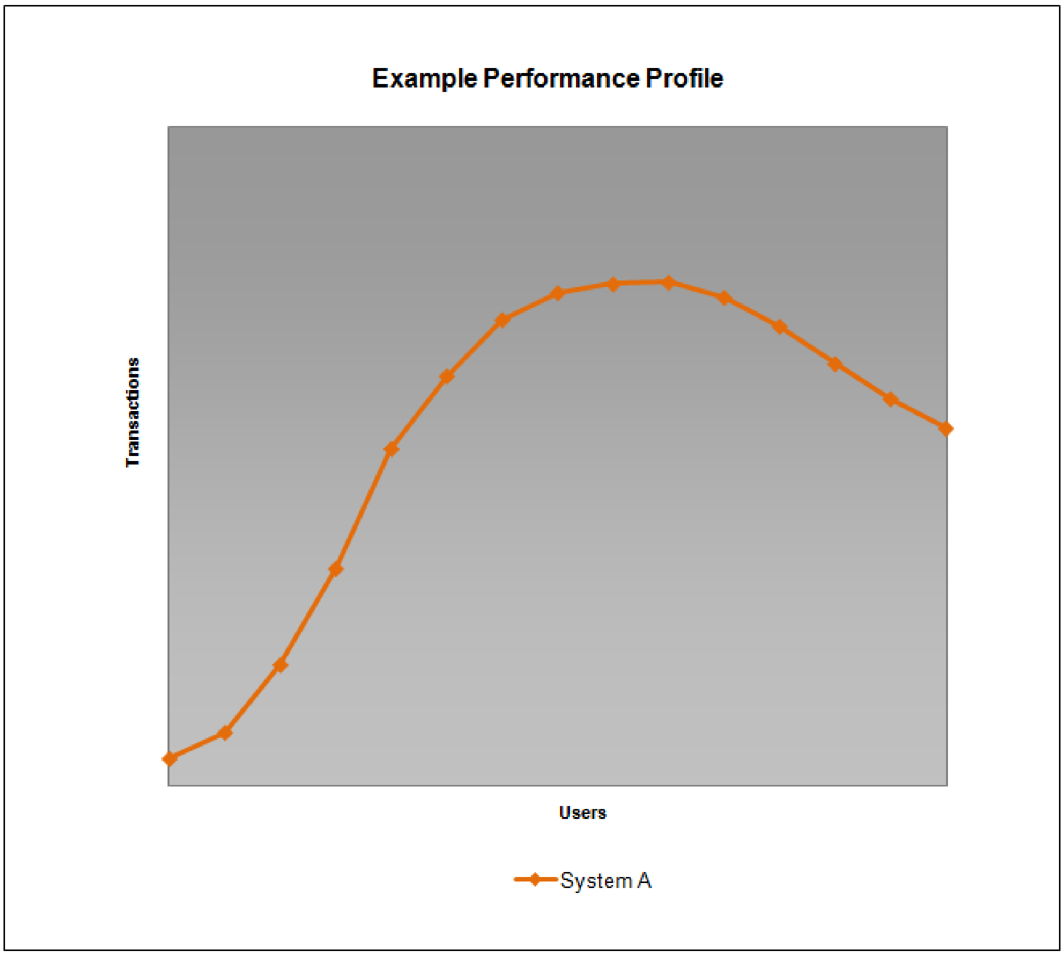


Figure 60 System A performance profile

TIP: The data produced for an official TPC-C benchmark is the logical equivalent of discarding all of the data points and keeping the top value only. Here you have produced significantly more data regarding the performance of the system across all levels of utilisation.

When you have data for multiple systems you can now add the performance data to the same spreadsheet and use the resultant graph to show a comparison. If you wish to compare the performance of both Oracle and MySQL on the same server then you should use the NOPM value as opposed to the TPM value for the y axis of the graph. This comparison provides a fair representation of the performance of both databases.

Performance Analysis

Performance Analysis on MySQL during Hammerora testing can be performed with a number of MySQL or Third-Party tools. With MySQL 5.5 the performance schema can used http://dev.mysql.com/doc/refman/5.5/en/performance-schema.html . Additionally viewing the storage engine status can provide further diagnosis information, for example:

```
MariaDB [(none)]> show engine innodb status \G;
*************************** 1. row ***************************
  Type: InnoDB
  Name:
Status:
=====================================
101214  4:29:11 INNODB MONITOR OUTPUT
=====================================
Per second averages calculated from the last 6 seconds
-----------------
BACKGROUND THREAD
-----------------
srv_master_thread loops: 1011 1_second, 1010 sleeps, 100 10_second, 427
background, 427 flush
srv_master_thread log flush and writes: 1072
----------
SEMAPHORES
----------
OS WAIT ARRAY INFO: reservation count 2970238, signal count 9148839
Mutex spin waits 35802598, rounds 187922292, OS waits 357177
RW-shared spins 3669916, OS waits 1058724; RW-excl spins 9468071, OS
waits 1226383
Spin rounds per wait: 5.25 mutex, 13.35 RW-shared, 8.57 RW-excl
…
```

Conclusion

You should now be equipped to perform fully scalable transactional workloads against a MySQL Database environment. An experienced MySQL Database tester will be able to go from bare metal to test in one working day to install a system, create the schema and start the automated test collecting the results the following working day making a 2 day project time possible to determine a full system performance profile. Do not forget that Hammerora is Open Source and you have the ability to modify the test scripts to fit whatever purpose meets your needs for testing MySQL transactional performance and comparing and contrasting MySQL with Oracle.

Support and Discussion

Need help? Try the Hammerora Sourceforge forum here :
http://sourceforge.net/projects/hammerora/forums/forum/292313

Want to discuss your results or have tips on tuning and configuration? open a Discussion topic here:

http://sourceforge.net/projects/hammerora/forums/forum/292312/topic/3848751