

SQL Server Instance-Level Benchmarks with HammerDB

TPC-C is an older standard for performing synthetic benchmarks against an OLTP database engine. The HammerDB tool is an open-sourced tool that can run these benchmarking tests against SQL Server, Oracle, MySQL, and PostgreSQL installations. It is open-sourced and freely available at <http://hammerdb.com>. It generates a benchmark values in the form of 'transactions placed per minute' and 'orders placed per minute' per test cycle, and tests the performance of the instance configuration and the infrastructure underneath it. You create a synthetic workload dataset with an included tool, or automate it with an add-on utility called AutoHammer.

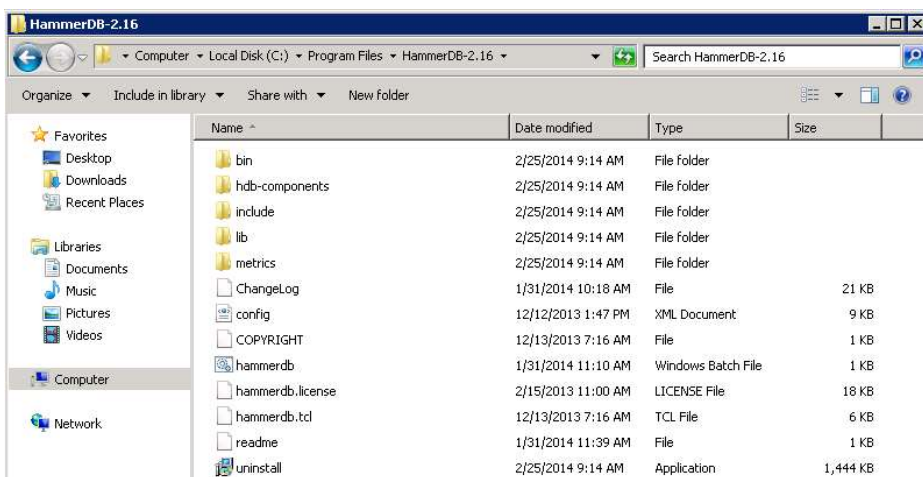
This tool is used quite frequently by Heraflux to compare the relative performance between two SQL Server instances. Load the same test data into two different instances, run the same test on each server, and compare the results. Raw server performance is compared, and differences in the server performance can be isolated.

An official quickstart guide to HammerDB is located for your reference at http://hammerdb.sourceforge.net/hammerdb_quickstart_v2.10.pdf.

Pre-Requisites

First, download the two pre-requisite test bundles. The core HammerDB release is located at SourceForge at <https://sourceforge.net/projects/hammerdb/files/HammerDB/HammerDB-2.16/HammerDB-2.16-Win-x86-64-Setup.exe/download>. The automation engine AutoHammer is located at https://sourceforge.net/projects/hammerdb/files/HammerDB/Autohammer%20Extension/autohammer_extension_5.zip/download. Download them both.

First, install the HammerDB runtime installer. The 64-bit installer will create a folder in the C:\Program Files\HammerDB-X.XX folder (where X.XX is the version number). Create a shortcut in a location of your choosing to the hammerdb.bat file inside this folder, as it does not create a shortcut on the Start menu.



Database Creation

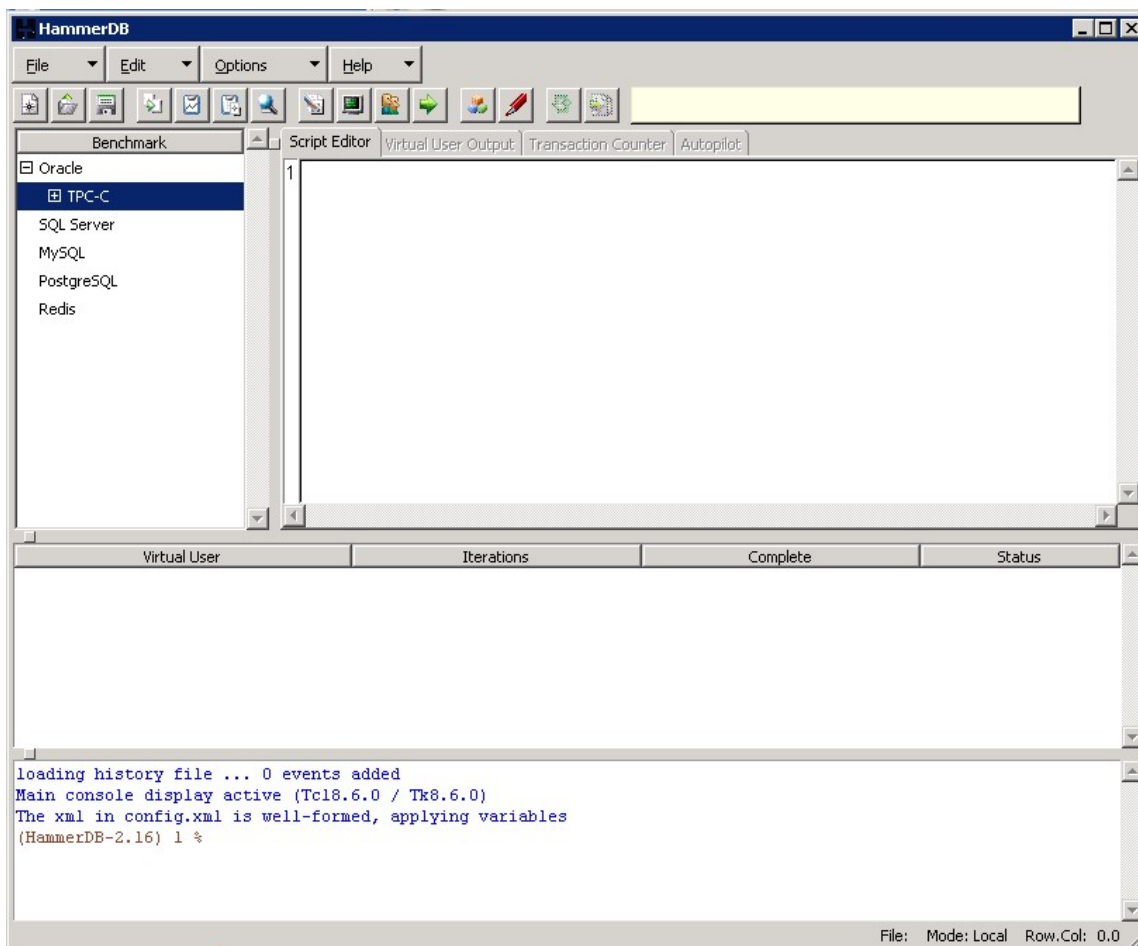
First, we need to create a placeholder database for this benchmark. The default creation process is not the most efficient in that it uses options that create inefficiencies, such as high VLF counts.

Modify the following script to create a new database with appropriately sized data and log files, the correct autogrowth sizes, and SIMPLE mode. Adjust the script as needed. This script was originally constructed for SQL Server 2012.

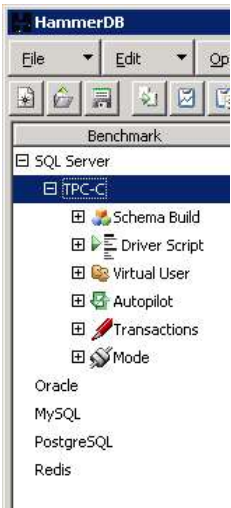
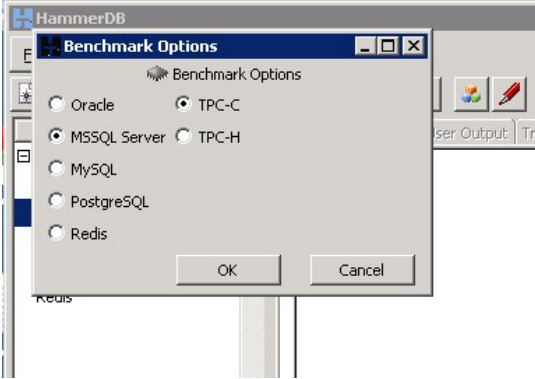
```
USE master
GO

CREATE DATABASE [tpcc]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'tpcc', FILENAME = N'E:\DATA\tpcc.mdf',
  SIZE = 10485760KB, FILEGROWTH = 524288KB )
LOG ON
( NAME = N'tpcc_log', FILENAME = N'F:\LOG\tpcc_log.ldf',
  SIZE = 1048576KB, FILEGROWTH = 524288KB )
GO
ALTER DATABASE [tpcc] SET RECOVERY SIMPLE
GO
```

Next, open the shortcut to hammerdb.bat. The default view opens to the Oracle database test.



Double click on the SQL Server benchmark. Select the MSSQL Server TPC-C option, and click OK.

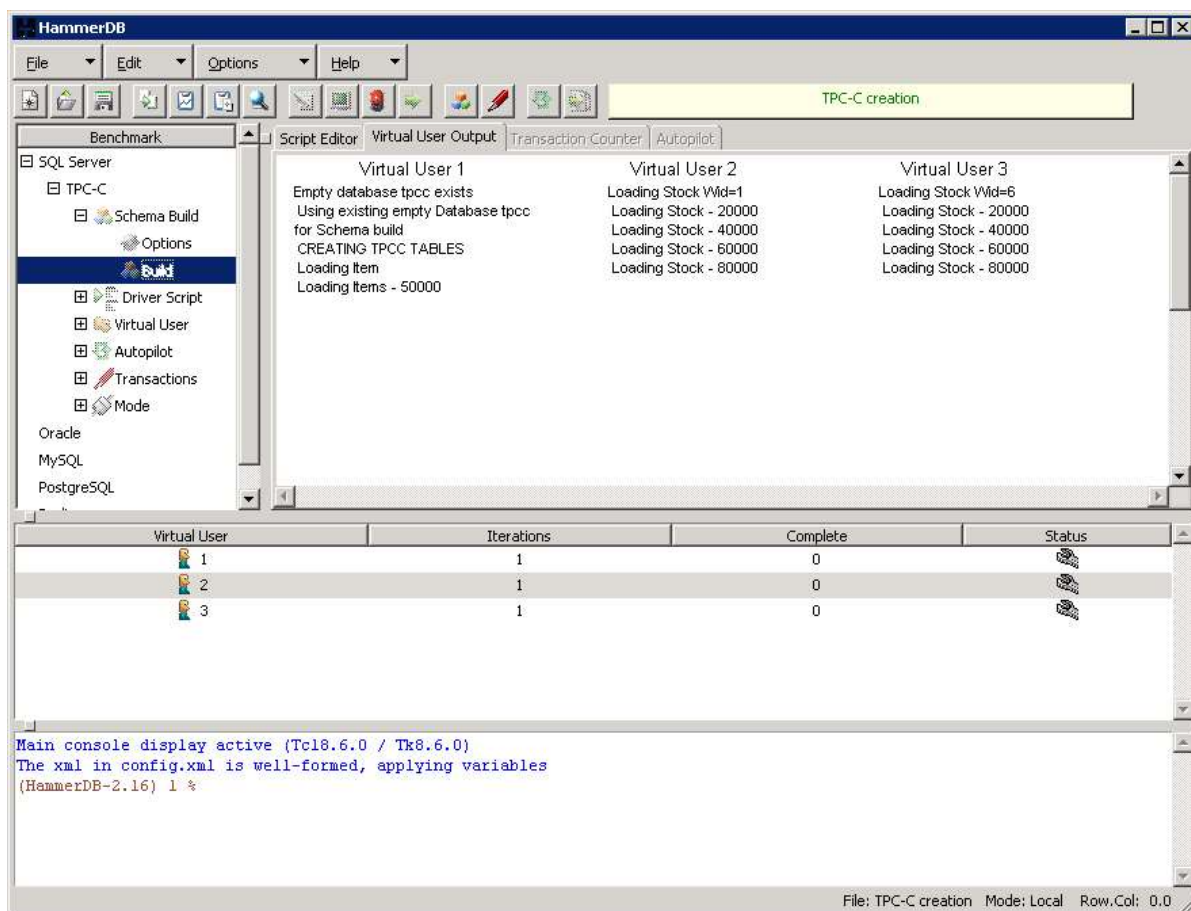


Expand 'Schema Build' and double-click Options. Name the database the same as the one you created within SQL Server. Select the number of warehouses and virtual users. This helps select the intensity of the OLTP workload. To help size the workload, the following items are examples of the workloads created by these options.

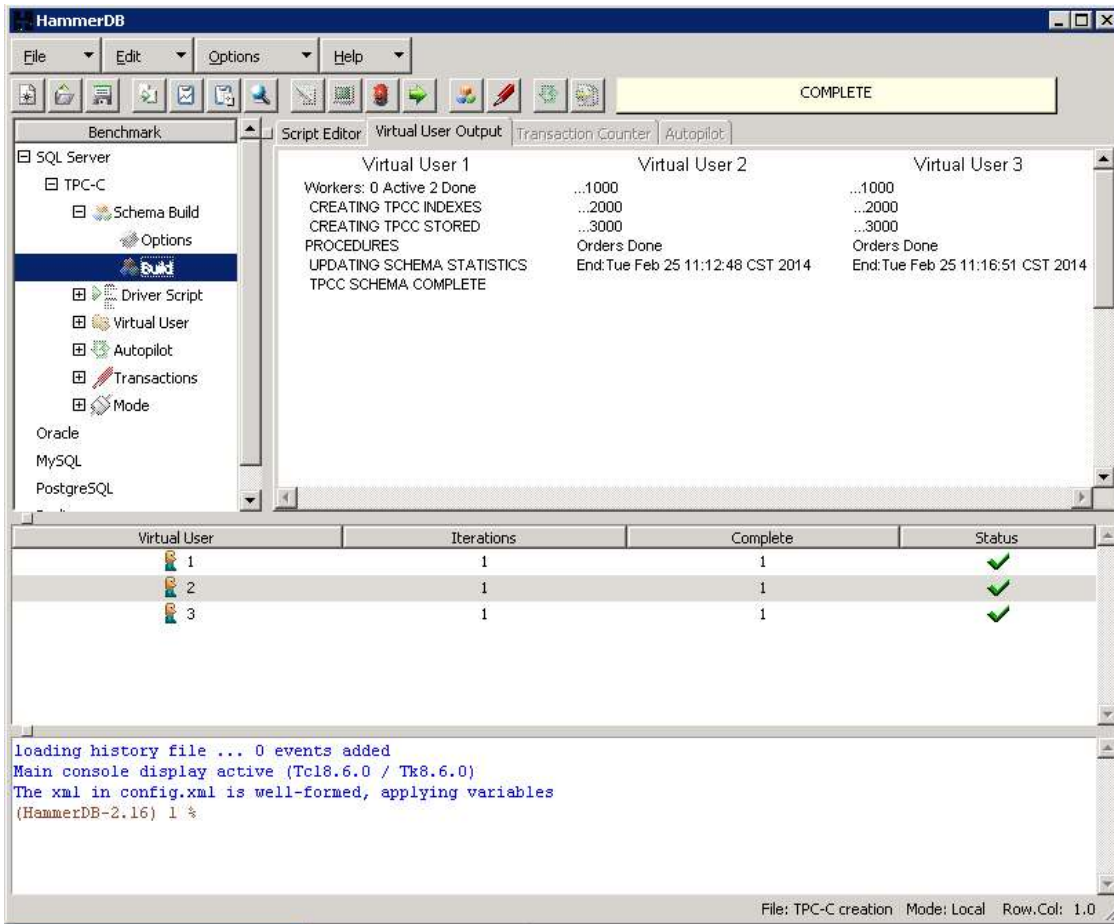
Number of Warehouses	Database Size	Compressed Backup Size
5	500MB	240MB
100	10GB	4.6GB
500	50GB	23GB

Select the number of warehouses that will create a database which is indicative of your average production workload.

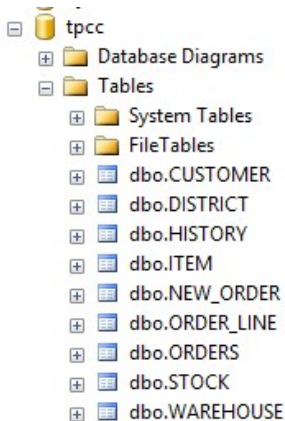
Next, double click Build, and the system will populate the database with sample data which it will use to benchmark the system.



Once complete, the header will change to 'COMPLETE'. Click the red stoplight icon in the header to end the process.



You will now see that the database has tables populated with data.

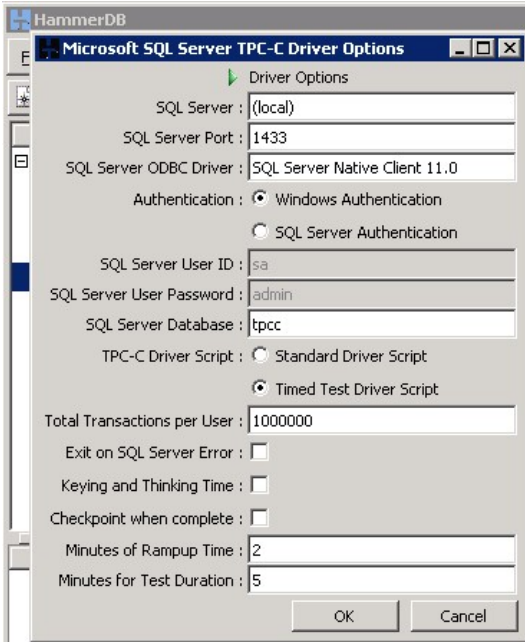


It is recommended to back up the new database for future restoration purposes, as this database population process is time consuming and the backup copy gives us an exact dataset to perform comparison testing on other systems with.

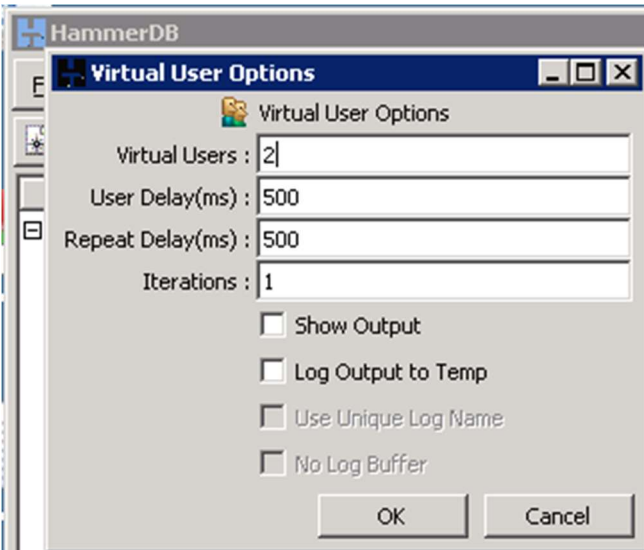
Load Testing

Open the shortcut to hammerdb.bat if the program is not already open and select SQL Server.

Expand 'Driver Script' and select Options. Select 'Timed Test Driver Script'. Leave the default rampup and test duration values unless you have a specific need to do otherwise.



Open 'Virtual User' then Options. Select the number of virtual users.



Make sure not to check 'Show Output', as it can slow the actual benchmark results considerably!

Select Create under 'Virtual User' to create the virtual users for the test.

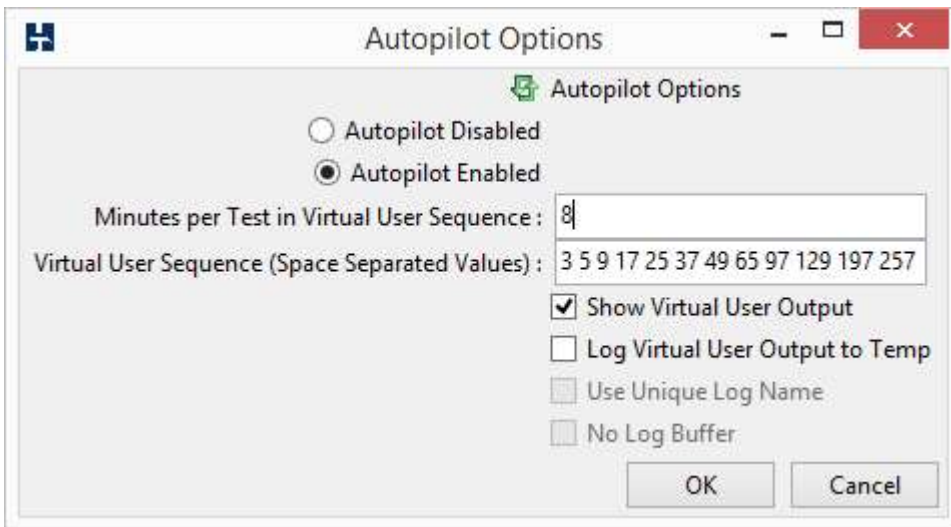
Finally, select the 'Run virtual users' green arrow icon in the header to start the test.


Autopilot Testing

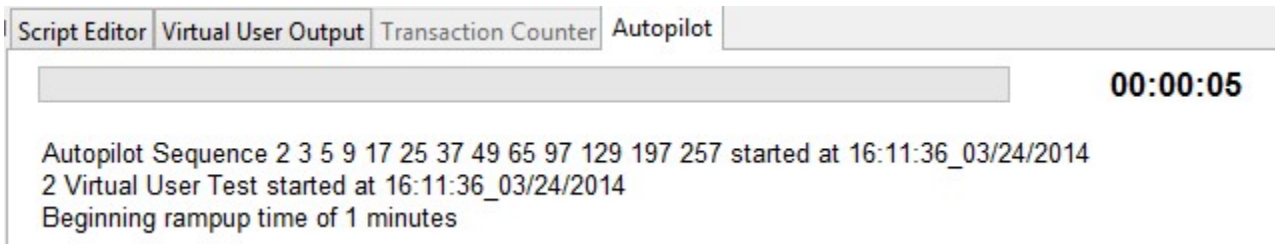
To perform a stress test with varying number of virtual users, load the Driver Script options as outlined above. Next, expand Autopilot. Double click Options.

Click the 'Autopilot Enabled' option, and reset the minutes per test field to eight minutes. At five minutes per test, plus two minutes for ramp-up, the eight minute window is more than enough time to complete each test.

Next, set the appropriate virtual user sequence values. These numbers are to be one greater than the number of actual users executed against the database, as the first user is a controlling user. In this example, we will be ramping up the users from two to 256.



Click OK to save these values. Next, click the green circular arrow icon () in the menu row to begin the automatic test bundle. In the Autopilot window, the test output resembles the following output.



Wait as the test bundle completes. Record the two values per test – TPM (Transactions Per Minute) and OPM (Orders Placed Per Minute) for the user value.

Interpreting Output

The final step to the HammerDB load testing is to interpret the output. Two metrics are produced by the testing:

- Transactions per minute
- Orders placed per minute

These two metrics can be used, along with the Perfmon sample collection during the tests, to show relative comparison of multiple machines.