Introduction to Decision Support, Data Warehousing, Business Intelligence, and Analytical Load Testing for all Databases

This guide gives you an introduction to conducting DSS (Decision Support System) testing based on the TPC-H specification on all databases supported with HammerDB workloads.

What is a Decision Support Workload?

The basis of Decision Support Systems is the ability to process complex ad-hoc queries on large volumes of data. In contrast to a transactional workload the focus is upon reading as opposed to modifying data and therefore requires a distinct approach. The ability of a database to process transactions gives limited information towards the ability of a database to support query based workloads and vice-versa, therefore both TPC-C and TPC-H based workloads complement each other in investigating the capabilities of a particular database. When reading large volumes of data to satisfy query workloads it should be apparent that if multiple CPU cores are available reading with a single processing thread is going to leave a significant amount of resources underutilized. Consequently the most effective Decision Support Systems employ a feature called Parallel Query to break down such queries into multiple sub tasks to complete the query more quickly. Additional features such as compression and partitioning can also be used to improve parallel query performance. Advances in server technologies in particular large numbers of CPU cores available with large memory configurations have popularised both in-memory and column store technologies as a means to enhance Parallel Query performance. Examples of databases supported by HammerDB that support some or all of these enhanced query technologies are the Oracle Database, TimesTen, SQL Server, and Greenplum, databases that do not support any of these technologies are MySQL and PostgreSQL and therefore single threaded query workloads cannot be expected to complete these workloads as quickly. As a NoSQL database Redis does not support Decision Support workloads. If you are unfamiliar with row-oriented and column-store technologies then it is beneficial to read one of the many guides explaining the differences and familiarising with the technologies available in the database that you have chosen to test. With commercial databases you should also ensure that your license includes the ability to run Parallel workloads as you may have a version of a database that supports single-threaded workloads only.

What is TPC-H?

To complement the usage of the TPC-C specification for OLTP workloads HammerDB also uses the relevant TPC specification for Decision Support Systems called TPC-H. Just as with the TPC-C specification the *"TPC benchmarks are industry standards. The TPC, at no charge, distributes its benchmark specifications to the public." Therefore* HammerDB includes an implementation of the specification of the TPC-H benchmark that can be run in any supported database environment. As with the load tests based upon TPC-C it is important to note that the implementation is not a full specification TPC-H benchmark and the query results cannot be compared with the official published benchmarks in any way. Instead the same approach has been taken to enable you to run an accurate and repeatable query based workload against your own database.

TPC-H in simple terms can be thought of as complementing the workload implemented in TPC-C related to the activities of a wholesale supplier. However whereas TPC-C simulates an online ordering system TPC-H represents the typical workload of business users inquiring about the performance of their business. To do this TPC-H is represented by a set of business focused ad-hoc queries (in addition to concurrent data updates and deletes) and is measured upon the time it takes to complete these queries. In particular the focus is upon highly complex queries that require the processing of large volumes of data.

Also in similarity to TPC-C the schema size is not fixed and is dependent upon a Scale Factor and there your schema your test schema can also be as small or large as you wish with a larger schema requiring a more powerful computer system to process the increased data volume for queries. However in contrast to TPC-C it is not valid to compare the test results of query load tests taken at different Scale Factors, Figure 2 shows the TPC-H schema.

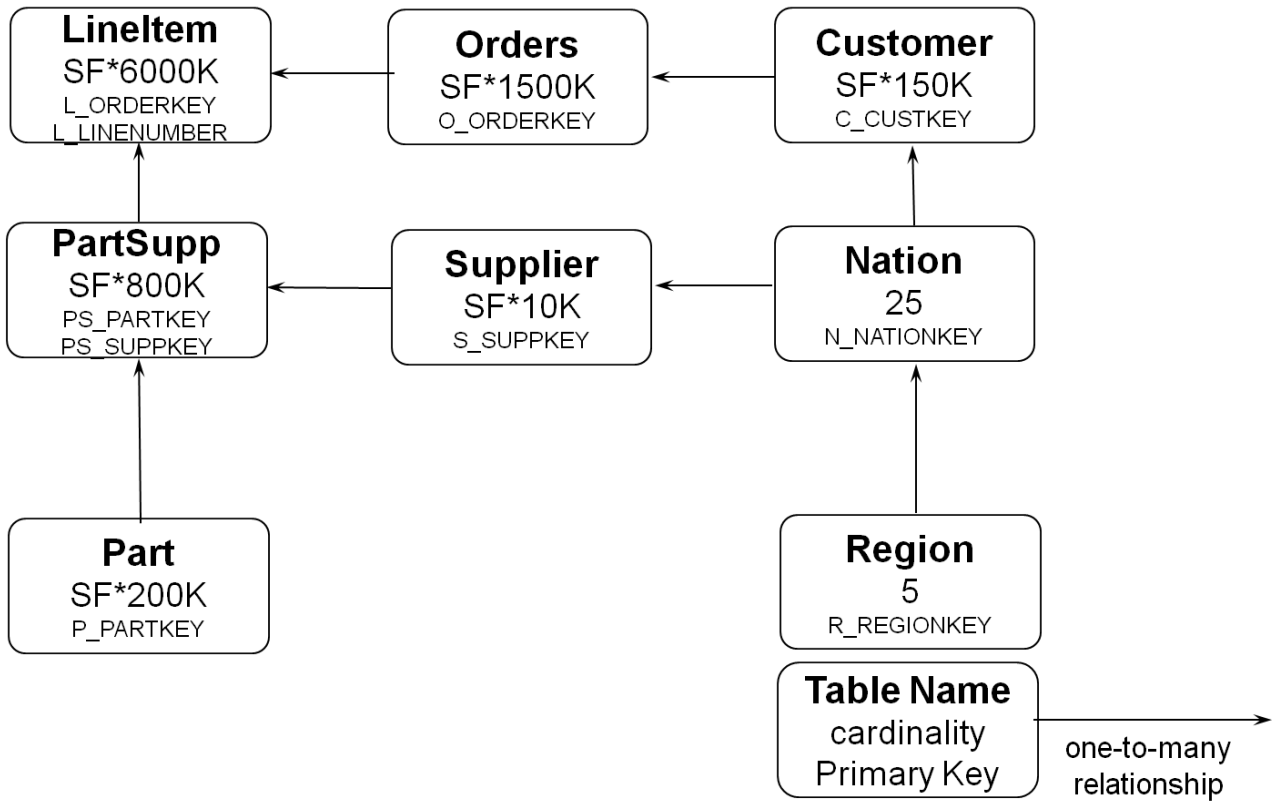


Figure 1 TPC-C Schema

The workload is represented by users executing a stream of 22 ad-hocs queries against the database with an example query as follows:

```
-- using 647655760 as a seed to the RNG
```

```
 select
        l_returnflag,
        l_linestatus,
        sum(l_quantity) as sum_qty,
        sum(l_extendedprice) as sum_base_price,
        sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
        sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as
sum_charge,
        avg(l_quantity) as avg_qty,
        avg(l_extendedprice) as avg_price,
        avg(l_discount) as avg_disc,
        count(*) as count_order
from
        lineitem
where
        l_shipdate <= date '1998-12-01' - interval '69' day (3)
group by
        l_returnflag,
        l_linestatus
order by
        l_returnflag,
        l_linestatus;
```

In measuring the results the key aspect is the time the queries take to complete , however if you wish to simulate a workload approaching the measurements used for an official TPC-H benchmark then the results are more complex:

There is a spreadsheet that can be downloaded from the HammerDB website documentation section that will calculate the QphH for you based on your results. You do not need to manually calculate the QphH yourself.

Generating Performance Profiles

As shown in figure 2 a typical performance profile is represented by the time it takes the system to process a query set from Q1 to Q22 (run in a pre-determined random order). At its most simplest you can take the Power Test and compare the timings required for a single virtual user across systems (whilst noting that with parallel query much more of the system will be utilised), or by measuring the Power and Throughput tests and calculating the QphH by utilising the provided spreadsheet.

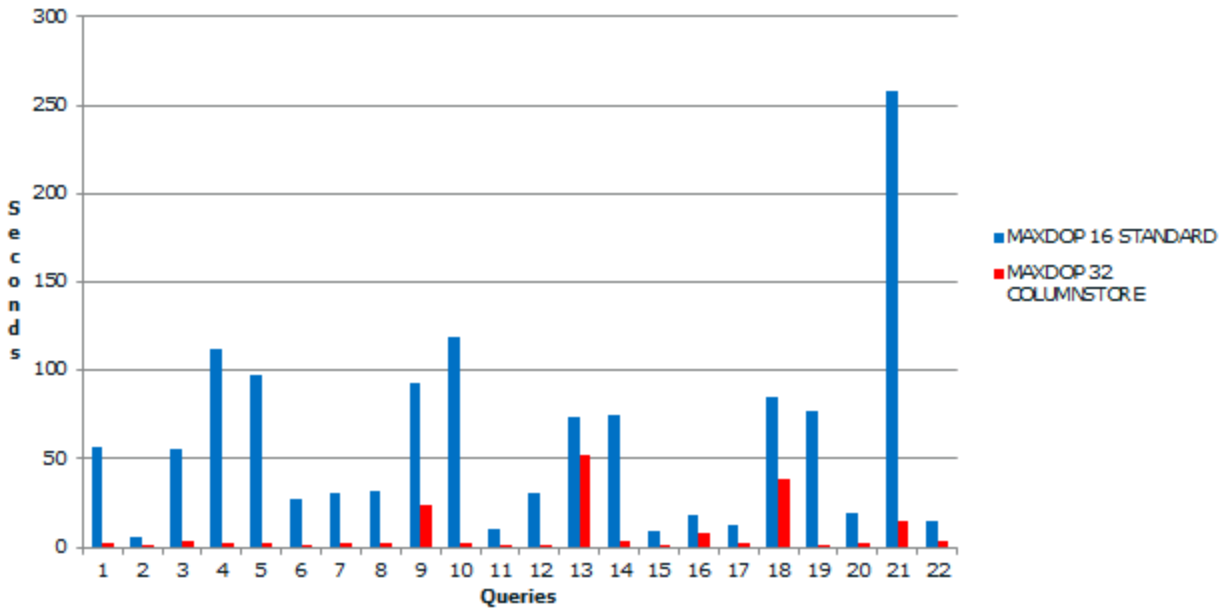


Figure 2 Performance Profile Example

Running the Power and Throughput Test and Calculating QphH

The Composite Query-per-Hour Performance Metric (QphH@Size) is calculated as follows:

$$QphH@Size = \sqrt{Power@Size \times Throughput@Size}$$

To calculate QphH it requires 3 aspects of the capability of the system to process queries:

1. Database size
2. Query processing power of queries in a single stream
3. Total query throughput of queries from multiple concurrent users

For the multiple concurrent user tests the throughput test always follows the power test and the number of users is based upon the following table:

Number of Query Streams fixed according to scale factor

SF (Scale Factor)	**S (Streams)**
100000	11
30000	10
10000	9
3000	8
1000	7
300	6
100	5

- 30 4
- 10 3
- 1 2

There is also a requirement for a simultaneous data refresh set. HammerDB provides full capabilities to run this refresh set both automatically as part of a Power test and concurrently with a Throughput test. Note however that once a refresh set is run the schema is required to be refreshed and it is prudent to backup and restore a HammerDB TPC-H based schema where running a refresh set is planned.

In order to calculate your QphH value you need to measure the following:

1. The Refresh Times for one set of the Refresh Function (run as part of a Power test).
2. The Query times for the Power test
3. The Query Time of the longest running virtual user for the throughput test.

Therefore once you have done your pre-testing run the tests detailed above and capture the values for the optimal configuration for your system ensuring that the configuration is the same for both the Power and Throughput tests.

It is important that you read the documentation for running a DSS workload for your chosen database as this explains in detail how with HammerDB you can simulate the running of a Power test including both the Refresh Function and Query test and how you should configure the throughput test also with the refresh function.

Calculating the QphH

The next stage is to prepare your calculations in order to be able to capture your query performance. In similarity to an OLTP workload you can we use a spreadsheet to do the query based calculations. As we have seen the calculation of QphH @ Size is taken as the Square Root of the Power @ Size metric multiplied by the Throughput @ Size metric. Consequently we must calculate the Power @ Size metric from the Power test and Throughput @ Size metric from the Throughput test.

Calculating Power @ Size

The Power @ Size metric is the inverse of the geometric mean of the timing intervals and shown as follows:

$$\text{TPC-H Power@Size} = \frac{3600 * SF}{\sqrt[24]{\prod_{i=1}^{i=22} QI(i,0) * \prod_{j=1}^{j=2} RI(j,0)}}$$

Here QI(i,0) is the timing interval, in seconds of the query stream and RI(j,0) is the timing interval, in seconds, of the refresh function and SF is the scale factor.

An alternative calculation is shown where ln(x) is the natural logarithm of x

$$\text{TPC-H Power@Size} = 3600 * \exp\left\{-\frac{1}{24}\left[\sum_{i=1}^{i=22} \ln(QI(i,0)) + \sum_{j=1}^{j=2} \ln(RI(j,0))\right]\right\} * SF$$

For strict adherence to TPC-H you should also be aware of the clause that specifies if the ratio between the longest running and shortest running query is greater than 1000 than all of the queries that took less than the maximum query time / 1000 should be increased to the maximum query time / 1000.

Using a spreadsheet to do the calculations for us makes the calculation of Power @ Size relatively straightforward requiring only the entry of the Scale Factor, Query Times and Refresh times.

Calculating Throughput @ Size

To calculate Throughput@Size the following function is used.

$$\text{TPC-H Throughput@Size} = (S*22*3600)/T_s * SF$$

Here SF is the scale factor and S the number of streams.And the timing interval T_s is the time taken between the first stream starting and the last stream completing. For simplicity because all of the query streams start at the same time this value can be taken as the longest time it takes for any of the query to complete .i.e. the time of the slowest query stream.

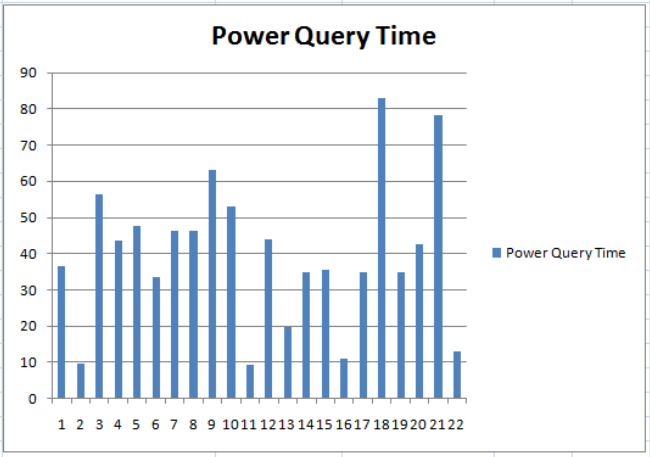
Calculating QphH @ Size

Given the Power @ Size Calculation and Throughput @ Size Calculation it is then simple to calculate the QphH as the Square of these two values multiplied.

$$\text{QphH@Size} = \sqrt{Power\ @\ Size * Throughput\ @\ Size}$$

Your completed spreadsheet will do this automatically for you resemble the following where the necessary input values are shown in red.

Power @ Size =	8830.53794		Throughput @ Size =	9902.4756		QphH @ Size =	9351.15964
Scale Factor	100		Streams	5			
	Power Query Time	Natural Logarithm	Throughput Maximum Query Stream Time				
Query 1	36.544	3.598517014	3999				
Query 2	9.646	2.266543322					
Query 3	56.345	4.031493505					
Query 4	43.643	3.776042903					
Query 5	47.821	3.867464874					
Query 6	33.444	3.509872398					
Query 7	46.469	3.838785424					
Query 8	46.375	3.836760521					
Query 9	63.072	4.144276931					
Query 10	53.078	3.97176253					
Query 11	9.227	2.222133969					
Query 12	43.953	3.783120881					
Query 13	19.749	2.983102857					
Query 14	35.022	3.555976435					
Query 15	35.619	3.572879203					
Query 16	10.942	2.392608596					
Query 17	34.889	3.552171593					
Query 18	82.864	4.41720071					
Query 19	34.815	3.550048328					
Query 20	42.694	3.754058395					
Query 21	78.218	4.3594998					
Query 22	13.046	2.568481573					
	Sum of Query Time	Sum of Natural Log					
	877.475	77.55280176					
Min	9.227						
Max	82.864						
Ratio	8.980600412						
Refresh1	616.8	6.424544822					
Refresh2	150.2	5.011967739					
	Sum of Refresh Time	Sum of Natural Log					
	767	11.43651256					



Hammerora TPC-H Calculations

Figure 3 Calculator Spreadsheet

And your QphH value in this case 9351 @ 100GB has been calculated for you. In the example spreadsheet note that the sample values are not intended to represent the performance of an actual system and are simply to demonstrate the calculation of QphH. You should also calculate your price/performance by dividing the QphH value by the total system cost.

Choosing a Database for running TPC-H workloads

As you have seen TPC-H workloads run complex queries scanning large volumes of data and therefore require the use of database features such as parallel query and in-memory column stores to maximise performance. Within the publically available HammerDB TPC-H based workloads the two databases that support these features are the Enterprise Editions of Oracle and SQL Server and therefore these databases will deliver the best experience for building and running TPC-H. There are additional TPC-H workloads supporting PostgreSQL/Greenplum and MySQL within HammerDB however due to the limitations of the databases themselves (and not HammerDB) these workloads should be viewed in an experimental manner as they will not result in the ability to generate a QphH value. Consequently until the time that these databases are able to fully run the TPC-H workload they remain undocumented for TPC-H.

Oracle and TimesTen

The Oracle database is fully featured for running TPC-H based workloads and is documented here Oracle DSS/Data Warehousing Testing Guide. You may also run TPC-H workloads against the TimesTen database using the same guide by selecting the option shown in Figure 4 however should be aware that TimesTen does not currently support parallel query or in-memory column stores and therefore the Oracle database will be more suitable for TPC-H based testing.

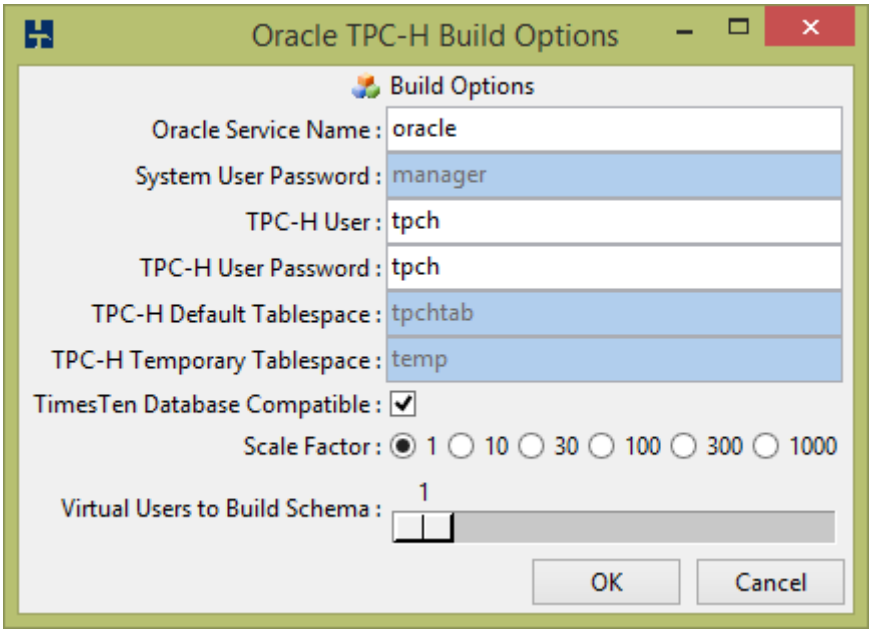


Figure 4 TimesTen compatible

Microsoft SQL Server

The Microsoft SQL Server database is fully featured for running TPC-H based workloads and is documented here Microsoft SQL Server DSS/Data Warehousing Testing Guide.

PostgreSQL and Greenplum

A TPC-H workload is available within HammerDB for MySQL however it is important to be aware that PostgreSQL has neither parallel query or column store features and therefore does not currently provide competitive performance for a TPC-H workload. Additionally the optimizer experiences challenges with some TPC-H queries in particular Queries 17 and 20 and therefore there is an option within the HammerDB driver script to skip the running of Queries 17 and 20 to ensure that the Query Set completes as shown in Figure 5.

```
711 #Queries 17 and 20 are long running on PostgreSQL
712 set SKIP_QUERY_17_20 "false"
```

Figure 5 Skip Query 17 and 20

Due to the limitations of PostgreSQL with query based workloads it remains undocumented within HammerDB and PostgreSQL TPC-H/Query workloads should be considered experimental until improvements in query processing and optimization are implemented within PostgreSQL. As an alternative to PostgreSQL, the HammerDB PostgreSQL workload can also run against the Greenplum database by selected the Greenplum Database Compatible option as shown in Figure 6.

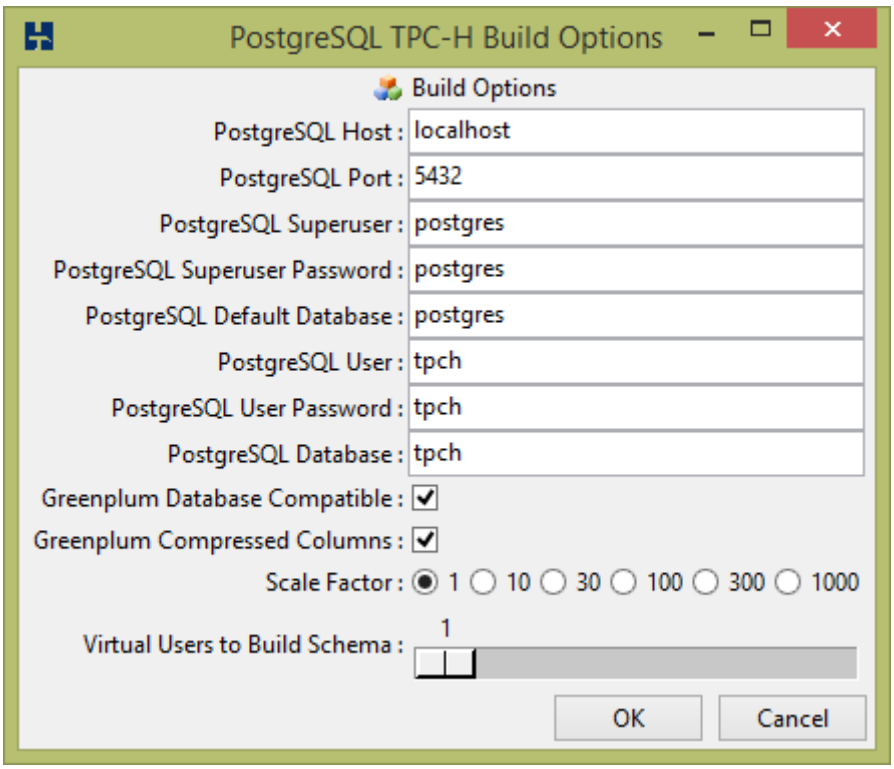


Figure 6 Greenplum Compatible

Nevertheless it is important to be aware that because of the Greenplum MPP architecture there is significant overhead in processing INSERT operations. This means that INSERT based workloads such as the HammerDB data load can take considerable lengths of time and become impractical for larger data sets. This limitation is not restricted to HammerDB inserts and is a limitation instead of the Greenplem architecture. Consequently Greenplum prefers bulk data load operations and therefore a best known method is to load the TPC-H schema into PostgreSQL and then bulk load from PostgreSQL into Greenplum.

MySQL

A TPC-H workload is available within HammerDB for MySQL however it is important to be aware that in similarity to PostgreSQL, MySQL has neither parallel query or column store features and therefore does not currently provide competitive performance for a TPC-H workload. Additionally the optimizer experiences challenges with some TPC-H queries in particular Query 18 and therefore there is an option within the HammerDB driver script to skip the running of Query 18 to ensure that the Query Set completes as shown in Figure 7.

```
676 #Query 18 is long running on MySQL
677 set SKIP_QUERY_18 "false"
```

Figure 7 Skip Query 18

Due to the limitations of MySQL with query based workloads it remains undocumented within HammerDB and MySQL TPC-H/Query workloads should be considered experimental until improvements in query processing and optimization are implemented within MySQL.

Redis

Redis is a single threaded NoSQL key-value database and therefore unable to process a TPC-H query workload. For this reason as shown in Figure 8 there is no option available to select TPC-H for Redis within HammerDB.

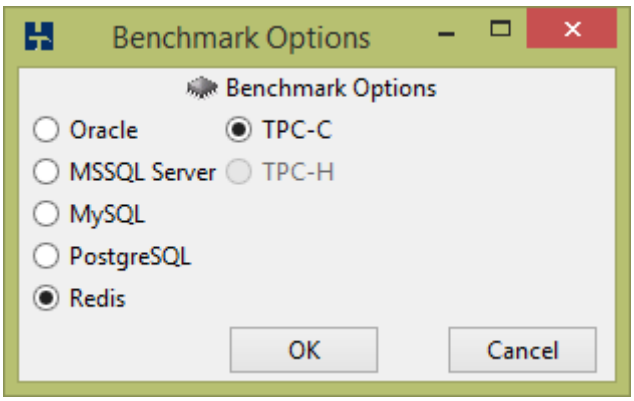


Figure 8 Redis TPC-H disabled

Comparing HammerDB results

In similarity to the the OLTP tests based on the TPC-C specification HammerDB implements a workload based on the TPC-H specification however does **NOT** implement a full specification TPC-H benchmark and the QphH results from HammerDB cannot be compared with the official published TPC-H benchmarks in any manner.

Publishing database performance results

Commercial software typically includes a clause (commonly known as a DeWitt clause) in the license agreement that prohibits the user from publishing program benchmarking results that have not been approved by the software vendor. This clause may exist in your database software license and/or your operating system and virtualization software system license. Typically this clause does not exist in the open source databases supported by HammerDB, namely MySQL, PostgreSQL and Redis. Consequently as a general guideline you are free to publish performance data generated by HammerDB against the open source databases in any domain, for the commercial databases you are recommended to consult the legal guidelines in your region. HammerDB is released under the GPL license and cannot accept responsibility for published data or offer legal advice on doing so.

Support and Questions

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