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? ................................................................. 1
What is TPC-H? ....................................................................................................... 1
Generating Performance Profiles ....................................................................... 3
Running the Power and Throughput Test and Calculating QphH ...................... 4
Calculating the QphH ............................................................................................ 5
Comparing HammerDB results ............................................................................. 6
Publishing database performance results ............................................................. 6
Support and Questions ......................................................................................... 7

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](image)

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

```sql
-- 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_linenumber
order by l_returnflag, l_linenumber;

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](image-url)
Running the Power and Throughput Test and Calculating QphH

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

\[
QphH @ Size = \sqrt{\text{Power@Size} \times \text{Throughput@Size}}
\]

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:

<table>
<thead>
<tr>
<th>Scale Factor (SF)</th>
<th>Streams (S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100000</td>
<td>11</td>
</tr>
<tr>
<td>30000</td>
<td>10</td>
</tr>
<tr>
<td>10000</td>
<td>9</td>
</tr>
<tr>
<td>3000</td>
<td>8</td>
</tr>
<tr>
<td>1000</td>
<td>7</td>
</tr>
<tr>
<td>300</td>
<td>6</td>
</tr>
<tr>
<td>100</td>
<td>5</td>
</tr>
<tr>
<td>30</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

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 Refresh Function (if desired)
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 simply 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.
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 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:

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

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 \( x \)

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

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.

\[
\text{TPC-H Throughput@Size} = \frac{(S \times 22 \times 3600)}{T_s} \times SF
\]

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.

\[ \text{QphH} \times \text{Size} = \sqrt{\text{Power} \times \text{Throughput} \times \text{Size}} \]

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

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.

---

**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.