Oracle Decision Support (DSS) Load Testing

This guide gives you an introduction to conducting Decision Support or analytical workloads on the Oracle Database. This guide will equip you with the essentials for assessing the ability of any system that runs the Oracle Database for processing query based workloads, such workloads are also known as Data Warehousing, Business Intelligence and Analytical workloads. On completion of this guide you will be able to run detailed and comprehensive query based Oracle load tests. If you have not already done so you should read the Introduction to Decision Support, Data Warehousing, Business Intelligence, and Analytical Load Testing for all Databases before proceeding with this guide.

You should ensure that your version of Oracle supports the Parallel Query and for Oracle 12c the In-Memory Option. Oracle Enterprise Edition does, however Oracle Standard Edition and Oracle Express do not and are therefore not suitable for running Query based workloads. This DSS HammerDB workload is the ideal workload for testing the features of In-Memory Column Stores.

Introduction ................................................................................................................. 1
SUT Database Server Configuration ............................................................................... 2
Installation and Configuration ....................................................................................... 3
SUT Database Server Installation .................................................................................. 3
Creating the Test Schema .............................................................................................. 4
Build Options .................................................................................................................. 7
Starting the Schema Build ............................................................................................. 8
Pre-Testing and Planning ............................................................................................... 16
Driver Options ................................................................................................................ 18
Loading the Driver Script ............................................................................................. 19
Pre-Test 1 Running a Power Test Query ...................................................................... 21
Pre-Test 2 Optimal DOP and In-memory for Power and Throughput Tests .................. 27
Running the DSS Tests ................................................................................................... 32
Power Test ...................................................................................................................... 32
Throughput Test ............................................................................................................. 36
Calculating QphH .......................................................................................................... 39
Support and Questions ................................................................................................. 40

Introduction

The basis of Analytic or Decision Support Systems is the ability to process complex ad-hoc queries on large volumes of data. Processing this amount of data within a single process or thread on traditional row-oriented database is time consuming. Consequently Oracle employs Parallel Execution to break down such queries into multiple sub tasks to complex the query more quickly. Additional features such as compression and partitioning are also used with Parallel Execution to improve parallel query performance. From Oracle version 12.1.0.2 there is also the option of utilising in-memory columnar storage. In-memory column stores can offer significant performance benefits and reduce query times for analytic workloads. Note that the Oracle in-memory columnar storage option is not the same as the in-memory parallel execution feature introduced with 11gR2 which applied to row based parallel execution, however in-memory columnar storage is also used in conjunction with parallel execution as shown below.
As a consequence when planning analytic workloads for optimal performance you should consider both your in-memory (from 12.1.0.2) and parallel execution configuration. In similarity to the HammerDB OLTP workload, HammerDB implements a fair usage of a TPC workload however the results should not be compared to official published TPC-H results in any way.

**SUT Database Server Configuration**

For query based workloads there is no requirement for a load testing client although you may use one if you wish. It is entirely acceptable to run HammerDB directly on the SUT (System Under Test) Database system if you wish, the client workload is minimal compared to an OLTP workload. In the DSS workload the client sends long running queries to the SUT and awaits a response therefore requiring minimal resources on the client side. As with an OLTP configuration however the database server architecture to be tested must meet the standard requirements for an Oracle Database Server. Similarly Oracle can be installed on any supported operating system, there is no restriction on the version of Oracle that is required.

Before running a HammerDB DSS test depending on your configuration you should focus on memory and I/O (disk performance). Also in turn the number and type of multi-core and multi-threaded processors installed will have a significant impact on parallel performance to drive the workload. When using in-memory column store features processors that support SIMD/AVX instructions sets are also required for the vectorisation of column scans.

HammerDB by default provides TPC-H schemas at Scale Factors 1,10,30,100,300 and 1000 (larger can be configured if required). The Scale Factors correspond to the schema size in Gigabytes. As with the official TPC-H tests the results at one schema size should not be compared with the results derived with another schema size. As the DSS workload utilizes parallel query it is possible for a single virtual user to use all of the CPU resources on the SUT at any schema size. Nevertheless there is still a relation with all of the hardware resources available including memory and I/O and a larger system will benefit from tests run a larger schema size. The actual sizing of hardware resources of hardware resources is beyond the scope of this document however at the basic level in contrast to an OLTP workload a traditional parallel query workload is typically characterized by direct path (directly from disk as opposed to from the buffer cache in the SGA) reads into the sessions’ PGA. Consequently the emphasis is more on PGA sizing as opposed to SGA for memory and high bandwidth read performance on the storage. With this traditional parallel execution and modern CPU capabilities I/O read performance is typically the constraining factor. Note that also in contrast to an OLTP workload high throughput redo log write performance is not a requirement, however in similarity to the OLTP workload storage based on SSD disks will usually offer significant improvements in performance over standard hard disks although in this case it is the benefits of read bandwidth as opposed to the IOPs benefits of SSDs for OLTP.

When using the in-memory column store memory capacity and bandwidth feature and if fully cached in memory storage performance is not directly a factor for query performance. Nevertheless data loads are an important consideration for in-memory data and therefore I/O and SSD read performance remain important for loading the data into memory to be available for scans.
Installation and Configuration

This section describes the procedure to install and configure the Load Generation Server (if one is used) and the SUT Database Server.

**SUT Database Server Installation**

Installation and configuration of the Oracle Database on your chosen operating system is beyond the scope of this document. You should have the Oracle database software installed, a test database created and running and be aware of the database features you wish to configure for query performance. During the installation make a note of your system user password, you will need it for the test schema creation. You may at your discretion use an existing database however please note that HammerDB load testing can drive your system utilization to maximum levels and therefore testing an active production system is not recommended. In particular parallel execution workloads can use significant resources for low numbers of users and result in table level checkpoints and therefore locking (TM enqueues) and caution is advised before running on an instance shared with other workloads.

When your database server is installed you should create a tablespace into which the test data will be installed allowing disk space according to the guide previously given in this document.

```
SQL> create tablespace tpch datafile size 2g;
Tablespace created.
```

When using parallel query ensure that the instance is configured for parallel execution, noting in particular the value for `parallel_max_servers`.

```
SQL> show parameter parallel

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>parallel_max_servers</td>
<td>integer</td>
<td>160</td>
</tr>
<tr>
<td>parallel_min_servers</td>
<td>integer</td>
<td>16</td>
</tr>
<tr>
<td>parallel_servers_target</td>
<td>integer</td>
<td>64</td>
</tr>
<tr>
<td>parallel_threads_per_cpu</td>
<td>integer</td>
<td>2</td>
</tr>
</tbody>
</table>
```

For testing purposes you can disable parallel execution in a particular environment by setting `parallel_max_servers` to a value of zero.

An additional parameter that can provide significant benefit to the performance of parallel query workloads is `optimizer_dynamic_sampling`. By default this value is set to 2. Increasing this value to 4 has been shown to benefit query performance however testing the impact of changing this parameter should always be done during pre-testing as it may change between Oracle releases.

```
SQL> alter system set optimizer_dynamic_sampling=4;
System altered.
```

```
SQL> show parameter optimizer_dynamic

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimizer_dynamic_sampling</td>
<td>integer</td>
<td>4</td>
</tr>
</tbody>
</table>
```
If using the 12c In-Memory option ensure that the parameter `inmemory_size` has been configured and the database restarted.

```
SQL> show parameter inmemory

NAME          TYPE       VALUE
----------------------------------------
inmemory_clause_default string     DEFAULT
inmemory_force         string     ENABLE
inmemory_max_populate_servers integer   2
inmemory_query         string     ENABLE
inmemory_size          big integer 1500M
inmemory_trickle_repopulate_servers_percent integer   1
optimizer_inmemory_aware boolean    TRUE
```

Then alter the new tablespace containing the schema to be in-memory.

```
SQL> alter tablespace TPCHTAB default inmemory;
Tablespace altered.
```

As shown the objects created within the tablespace will now be configured to be in-memory.

```
SQL> select tablespace_name, def_inmemory from dba_tablespaces;

TABLESPACE_NAME     DEF_INMEM
--------------- -------
SYSTEM               DISABLED
SYSAUX               DISABLED
TEMP                 DISABLED
USERS                DISABLED
TPCCTAB              DISABLED
TPCHTAB              ENABLED
```

For larger schemas both partitioning and compression settings (both standard and in-memory) should be considered for query tuning.

---

**Creating the Test Schema**

To create the DSS test schema based on the TPC-H 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 Oracle and TPC-H and press OK as shown in Figure 1.
Figure 1 Select Benchmark

Confirm the benchmark by pressing OK as shown in Figure 2.

Figure 2 Confirm Options

HammerDB will now be ready to create and run a TPC-H based workload as shown in Figure 3.
As with the OLTP workloads if selected from the top menu the schema options window is divided into two sections. The “Build Options” section details the general login information and where the schema will be built and the “Driver Options” for the Driver Script to run after the schema is built. “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 schema every time you change the “Driver Options”, once the schema has been built only these “Driver Options” may need to be modified and therefore you can also select to view the Build Options only from the treeview as shown in Figure 4.
Figure 4 Build Options

For the “Build Options” fill in the values according to the database where the schema will be built.

Figure 5 Schema Options

Build Options

The Build Option values have the following meanings.
Oracle Service Name
The Oracle Service Name is the service name that your load generation server will use to connect to the database running on the SUT database server.

System User Password
The system user password is the password for the “system” user you entered during database creation. The system user already exists in all Oracle databases and has the necessary permissions to create the TPC-H user.

TPC-H User
The TPC-H user is the name of a user to be created that will own the TPC-H schema. This user can have any name you choose but must not already exist and adhere to the standard rules for naming Oracle users. You may if you wish run the schema creation multiple times and have multiple TPC-H schemas created with ownership under a different user you create each time.

TPC-H User Password
The TPC-H user password is the password to be used for the TPC-H user you create and must adhere to the standard rules for Oracle user password. You will need to remember the TPC-H user name and password for running the TPC-H driver script after the schema is built.

TPC-H Default Tablespace
The TPC-H default tablespace is the tablespace that will be the default for the TPC-H user and therefore the tablespace to be used for the schema creation. The tablespace must have sufficient free space for the schema to be created. If using In-memory features the tablespace should also be enabled as described previously to use this feature.

TPC-H Temporary Tablespace
The TPC-H temporary tablespace is the temporary tablespace that already exists in the database to be used by the TPC-H User.

TimesTen Database Compatible
This option is to be selected when running the test against an Oracle TimesTen database as opposed to the Oracle Database.

Scale Factor
The Scale Factor is selected by a radio button with a choice of scale factors of 1,10,30,100,300 and 1000 corresponding to 1GB, 10GB, 30GB, 100GB and 1000GB respectively (larger schema sizes can also be created) although the required space will be larger than these values due to the indexes required.

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 the number of cores/Hyper-Threads on your Load Generation Server or SUT if HammerDB is running there.

Starting the Schema Build
When you have completed your Build Options click OK to store the values you have entered. For a permanent record the values can be entered directly into the config.xml file. On starting HammerDB the schema options will already contain the values you have entered in the corresponding fields, for example:
To begin the schema creation at the buttons in the top level window click the "Create TPC Schema" button. as shown in Figure 6.

Figure 6 Create Schema
On clicking this button a dialogue box such as the one shown in Figure 7 appears.
Figure 7 Confirm Schema

When you click Yes HammerDB will login to your chosen service name with a monitor thread as the system user and create the user with the password you have chosen. It will then log out and log in again as your chosen user, create the tables and then load the region and nation table data before waiting and monitoring the other threads. The worker threads will wait for the monitor thread to complete its initial work. Subsequently the worker threads will create and insert the data for their assigned warehouses as shown in Figure 8. There are no intermediate data files or manual builds required, HammerDB will both create and load your requested data dynamically. Data is inserted in a batch format for optimal performance.

Figure 8 Schema Building

When the workers are complete the monitor thread will create the indexes, stored procedures and gather the statistics. When complete Virtual User 1 will display the message TPCH SCHEMA COMPLETE and all virtual users will show that they completed their action successfully, press the button to destroy the virtual
users as shown in Figure 9.

Figure 9 Schema Build Complete

And clear the script editor as shown in Figure 10.
The schema build is now complete with the following tables created and populated. Note that in example below the tables have inherited the tablespace’s in-memory configuration without additional settings. If required the inmemory_priority can also be set at this point in time.

```
1* select table_name, num_rows, inmemory, inmemory_priority from user_tables
SQL> /  
```

<table>
<thead>
<tr>
<th>TABLE_NAME</th>
<th>NUM_ROWS</th>
<th>INMEMORY</th>
<th>INMEMORY_PRIORITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORDERS</td>
<td>1500000</td>
<td>ENABLED</td>
<td>NONE</td>
</tr>
<tr>
<td>PARTSUPP</td>
<td>800000</td>
<td>ENABLED</td>
<td>NONE</td>
</tr>
<tr>
<td>CUSTOMER</td>
<td>1500000</td>
<td>ENABLED</td>
<td>NONE</td>
</tr>
<tr>
<td>PART</td>
<td>200000</td>
<td>ENABLED</td>
<td>NONE</td>
</tr>
<tr>
<td>SUPPLIER</td>
<td>10000</td>
<td>ENABLED</td>
<td>NONE</td>
</tr>
<tr>
<td>NATION</td>
<td>25</td>
<td>ENABLED</td>
<td>NONE</td>
</tr>
<tr>
<td>REGION</td>
<td>5</td>
<td>ENABLED</td>
<td>NONE</td>
</tr>
<tr>
<td>LINEITEM</td>
<td>6003632</td>
<td>ENABLED</td>
<td>NONE</td>
</tr>
</tbody>
</table>

The TPC-H schema creation script is a standard HammerDB script like any other so you can save it, modify it and re-run it just like any other HammerDB script. For example if you wish to create more than a scale factor 1000 schema 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 larger scale factor, for example the following will create a scale factor 10000 schema.
do_tpch oracle pdb1 10000 tpch tpch tpchtab temp false 4

Similarly change any other value to modify your script. If you have made a mistake simply delete the schema as follows:

SQL> drop user tpch cascade;

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

SQL> select tname, tabtype from tab;

<table>
<thead>
<tr>
<th>TNAME</th>
<th>TABTYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUSTOMER</td>
<td>TABLE</td>
</tr>
<tr>
<td>LINEITEM</td>
<td>TABLE</td>
</tr>
<tr>
<td>NATION</td>
<td>TABLE</td>
</tr>
<tr>
<td>ORDERS</td>
<td>TABLE</td>
</tr>
<tr>
<td>PART</td>
<td>TABLE</td>
</tr>
<tr>
<td>PARTSUPP</td>
<td>TABLE</td>
</tr>
<tr>
<td>REGION</td>
<td>TABLE</td>
</tr>
<tr>
<td>SUPPLIER</td>
<td>TABLE</td>
</tr>
</tbody>
</table>

8 rows selected.

SQL> select * from customer where rownum = 1;

<table>
<thead>
<tr>
<th>C_CUSTKEY</th>
<th>C_MKTSEGME</th>
<th>C_NATIONKEY</th>
<th>C_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C_ADDRESS</td>
<td>C_PHONE</td>
<td>C_ACCTBAL</td>
<td>C_COMMENT</td>
</tr>
</tbody>
</table>
| v,QXkbT8yhhyQYXjX4Ag3iFPQq0gbfZNo7 | 776-160-1375 | 5010.19 | carefully pending instructions detect slyly-- pending deposits acco

SQL> select index_name, index_type from ind;

<table>
<thead>
<tr>
<th>INDEX_NAME</th>
<th>INDEX_TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>REGION_PK</td>
<td>NORMAL</td>
</tr>
<tr>
<td>NATION_PK</td>
<td>NORMAL</td>
</tr>
<tr>
<td>SUPPLIER_PK</td>
<td>NORMAL</td>
</tr>
<tr>
<td>PARTSUPP_PK</td>
<td>NORMAL</td>
</tr>
<tr>
<td>PART_PK</td>
<td>NORMAL</td>
</tr>
<tr>
<td>ORDERS_PK</td>
<td>NORMAL</td>
</tr>
<tr>
<td>LINEITEM_PK</td>
<td>NORMAL</td>
</tr>
<tr>
<td>CUSTOMER_PK</td>
<td>NORMAL</td>
</tr>
</tbody>
</table>

8 rows selected.

You can monitor the amount of space you have used in your schema with a statement such as follows:

SQL> select sum(bytes)/1024/1024 as MB from user_segments;

<table>
<thead>
<tr>
<th>MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1351.25</td>
</tr>
</tbody>
</table>

At this point the data creation is complete and you are ready to start running a performance test. Before
doing so note that as this is a query based workload you have the potential to run multiple tests and it will return the same results as the data is not modified during tests, however there is one exception, under the Driver Options the option to choose a Refresh Function. Further details will be given on the refresh function in the next section however at this point it is sufficient to note that the refresh function when enabled will modify data and no two same refresh functions can be run on the same data set. This means if you choose to include a refresh function and then attempt to re-run the test you will receive an error as the data has been modified. This means you have a number of options. Firstly (and recommended) you can use datapump to backup and restore your schema. To do this create a directory as follows to reference a file system folder you have already created (or use the pre-existing DATA_PUMP_DIR)

```
SQL> create directory dump_dir1 as '/u02/app/oracle/dumpdir';
Directory created.

Then use datapump to export your schema to this directory before you have run any workloads with a refresh function:

[oracle@MERLIN oracle]$ expdp "sys/oracle@pdb1 as sysdba" schemas=tpch content=all directory=DUMP_DIR1 logfile=dp.log
```

```
Export: Release 12.1.0.2.0 - Production on Wed Sep 17 11:23:32 2014
Copyright (c) 1982, 2014, Oracle and/or its affiliates. All rights reserved.

Connected to: Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options
Starting "SYS"."SYS_EXPORT_SCHEMA_01": "sys/******@pdb1 AS SYSDBA"
schemas=tpch content=all directory=DUMP_DIR1 logfile=dp.log
Estimate in progress using BLOCKS method...
Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA
Total estimation using BLOCKS method: 1.159 GB
Processing object type SCHEMA_EXPORT/User
Processing object type SCHEMA_EXPORT/SYSTEM_GRANT
Processing object type SCHEMA_EXPORT/ROLE_GRANT
Processing object type SCHEMA_EXPORT/DEFAULT_ROLE
Processing object type SCHEMA_EXPORT/TABLESPACE_QUOTA
Processing object type SCHEMA_EXPORT/PRE_SCHEMA/PROCACT_SCHEMA
Processing object type SCHEMA_EXPORT/TABLE/TABLE
Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
Processing object type SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
Master table "SYS"."SYS_EXPORT_SCHEMA_01" successfully loaded/unloaded
******************************************************************************
Dump file set for SYS.SYS_EXPORT_SCHEMA_01 is: /u02/app/oracle/dumpdir/expdat.dmp
```

14
Job "SYS"."SYS_EXPORT_SCHEMA_01" successfully completed at Wed Sep 17 11:24:10 2014 elapsed 0 00:00:36

After you have run a workload with a refresh function drop the TPCH user as follows:

SQL> drop user tpch cascade;

User dropped.

Then re-import the export file you took prior to running the refresh function:

[oracle@MERLIN oracle]$ impdp \"sys/oracle@pdb1 as sysdba\" schemas=tpch content=all directory=DUMP_DIR1 logfile=dp1.log

Import: Release 12.1.0.2.0 - Production on Wed Sep 17 11:37:54 2014
Copyright (c) 1982, 2014, Oracle and/or its affiliates. All rights reserved.

Connected to: Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options
Master table "SYS"."SYS_IMPORT_SCHEMA_04" successfully loaded/unloaded
Starting "SYS"."SYS_IMPORT_SCHEMA_04": "sys/********@pdb1 AS SYSDBA" schemas=tpch content=all directory=DUMP_DIR1 logfile=dp1.log

Processing object type SCHEMA_EXPORT/USER
Processing object type SCHEMA_EXPORT/SYSTEM_GRANT
Processing object type SCHEMA_EXPORT/ROLE_GRANT
Processing object type SCHEMA_EXPORT/DEFAULT_ROLE
Processing object type SCHEMA_EXPORT/TABLESPACE_QUOTA
Processing object type SCHEMA_EXPORT/PROCEDURE_SCHEMA
Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/REF_CONSTRAINT
Processing object type SCHEMA_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Processing object type SCHEMA_EXPORT/STATISTICS/MARKER

Job "SYS"."SYS_IMPORT_SCHEMA_04" successfully completed at Wed Sep 17 11:38:40 2014 elapsed 0 00:00:44

You only need to export once and can then re-import as many times as you wish to run the successfully refresh function.

Secondly another option you have is to use dbms_metadata to capture the table definitions and then use SQL*Loader to export and import the data.

SQL> l1* select dbms_metadata.get_ddl('TABLE','LINEITEM') from dual
SQL> /

DBMS_METADATA.GET_DDL('TABLE','LINEITEM')
CREATE TABLE "TPCH"."LINEITEM"
(
   "L_SHIPDATE" DATE,
   "L_ORDERKEY" NUMBER NOT NULL ENABLE,
   "L_DISCOUNT" NUMBER NOT NULL ENABLE,
   "L_EXTENDEDPRICE" NUMBER NOT NULL ENABLE,
   "L_SUPPKEY" NUMBER NOT NULL ENABLE,
   "L_QUANTITY" NUMBER NOT NULL ENABLE,
   "L_RETURNFLAG" CHAR(1),
   "L_PARTKEY" NUMBER NOT NULL ENABLE,
   "L_LINENUMBER" NUMBER NOT NULL ENABLE,
   "L_SHIPINSTRUCT" CHAR(25),
   "L_COMMENT" VARCHAR2(44),
   CONSTRAINT "LINEITEM_PK" PRIMARY KEY ("L_LINENUMBER", "L_ORDERKEY")
USING INDEX PCTFREE 2 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "TPCHTAB" ENABLE,
CONSTRAINT "LINEITEM_PARTSUPP_FK" FOREIGN KEY ("L_PARTKEY",
   "L_SUPPKEY")
   REFERENCES "TPCH"."PARTSUPP" ("PS_PARTKEY", "PS_SUPPKEY") ENABLE,
CONSTRAINT "LINEITEM_ORDER_FK" FOREIGN KEY ("L_ORDERKEY")
   REFERENCES "TPCH"."ORDERS" ("O_ORDERKEY") ENABLE
) SEGMENT CREATION IMMEDIATE
PCTFREE 2 PCTUSED 98 INITRANS 8 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "TPCHTAB"
INMEMORY PRIORITY NONE MEMCOMPRESS FOR QUERY LOW
DISTRIBUTE AUTO NO DUPLICATE PARALLEL

Finally if you have the flashback table feature enabled you can note the time that you start running a test
with a refresh function and then flashback the LINEITEM and ORDERS table to their previous state before
the test, for example:
flashback table lineitem to timestamp TO_TIMESTAMP('17-SEP-14 11.41.00.00 AM')

Whichever method you use, ensure that if you wish to run the refresh function you are prepared to restore
your schema to the previous state before running subsequent tests.

---

**Pre-Testing and Planning**

For DSS tests as with OLTP tests, after schema creation but before you start running measured tests an
important phase is pre-testing and planning. Pre-testing is a phase also known as ‘testing the tests’, in this
phase you verify that you have the optimal system, operating system and Oracle configuration which you
then document and hold consistent for a series of tests. Pre-testing enables you to ensure that your
configuration is suitable for testing and the time invested will generate valid results. Pre-testing also enables you to gain familiarity with the HammerDB driver script settings.

For DSS testing a key focus is determining the optimal Degree of Parallelism (DOP) and in-memory Oracle configuration. 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 if you have not already done so read the Introduction to Decision Support, Data Warehousing, Business Intelligence, and Analytical Load Testing for all Databases and therefore familiar with the Power Test, Throughput Test and Refresh Function. You should also have downloaded the HammerDB TPC-H Calculator to enable automatic calculation of the results of your tests.

Your first pre-test should be based upon the Power test. From the Options menu or treeview select the Driver Options as shown in Figure 11.

![Figure 11 Driver Options.](image)

At this stage your focus is now on completing the settings under Driver Options as shown in Figure 12.
**Driver Options**

In addition to the connection settings and TimesTen option that are the same as those used for the Build Options, namely:

**Oracle Service Name/System User Password/TPC-H User/TPH-User Password/TimesTen Database Compatible**

The values of the Oracle Service Name, System User Password, TPC-H User, TPH-User Password and TimesTen Database compatible detailed under the Build Options of the Schema Options page is shown as the connect value in the Driver Script. This value defines the connect string for the user who owns the TPC-H schema. As long as you can connect to Oracle with SQL*PLUS using exactly the same connect string then HammerDB will also be able to connect. If having difficulty connecting with HammerDB then troubleshoot with the normal tools as described previously such as tnsping to resolve the connectivity issues.

**Scale Factor**

Although not visible under the Driver Options the Scale Factor value is also inherited from the Build Options and must be the set to the same value for running the Driver Script as was used for the Build. This is especially important if you have restarted HammerDB as you may need to set the Scale Factor in the Build Options again or manually in the script for correct functionality or set the value you are using in the config.xml file.

Under the Driver Options section you have the following choices:

**Total Query Sets per User**

As you have already seen from the introduction document a Query Set is a sequence of 22 queries. The Total number of query sets is the number of times after logging on that the virtual user completes an entire sequence of queries before logging off again. The difference between this and using iterations value in the Virtual User options is that the virtual user only logs on and off once and completes all of the query sets in between whereas with the iterations value the entire script is run multiple times.
Exit on Oracle Error

Exit on Oracle Error is shown as the parameter RAISEERROR in the Driver Script. RAISEERROR impacts the behaviour of an individual virtual user on detecting an Oracle error. If set to TRUE on detecting an Oracle error the user will report the error into the HammerDB console and then terminate execution. If set to FALSE the virtual user will ignore the error and proceed with executing the next transaction. It is therefore important to be aware that if set to FALSE firstly if there has been a configuration error resulting in repeated errors then the workload might not be reported accurately and secondly you may not be aware of any occasional errors being reported as they are silently ignored. I recommend running pre-tests with RAISEERROR set to TRUE to ensure a configuration is valid before setting it to FALSE for a measured test run.

Verbose Output

Verbose Output is shown as VERBOSE in the Driver Script. Setting this value to TRUE will print both the Queries and their results for each virtual user however will add to the Query time by the time required to print the results.

Degree of Parallelism

On a simple level the Degree of Parallelism defines the number of Parallel Execution Server processes that the Queries will be executed with. The Degree of Parallelism is defined as the degree of parallel in the driver script. You should consult a good reference on Parallel Execution as the actual execution environment is more complex including both Producer and Consumer Parallel Execution Servers. This value will be determined by your available hardware resources and may be different for both the Power and Throughput tests. HammerDB will ensure that the test will run at your chosen degree of parallelism (also dependant on your settings of parallel_min and parallel_max servers).

Refresh Function

The refresh function checkbox corresponds to refresh on in the Driver Script. When this checkbox is enabled the first virtual user will run the refresh function as opposed to running a query set. Note that if you choose only one virtual user and select the refresh function checkbox then your virtual user will run a power test as detailed further in this document. The refresh function as the name implies inserts and deletes rows from the ORDERS and LINEITEM tables and the times of this function are required as input to calculating the QphH.

Number of Update Sets/Trickle Refresh Delay(ms)/Refresh Verbose

If you have enabled the refresh function then the values for Number of Update Sets/Trickle Refresh Delay(ms)/Refresh Verbose become active and these correspond to update_sets trickle_refresh and REFRESHVERBOSE in the driver script respectively. The update sets determines how many times the virtual users will cycle through the refresh functions whilst noting that the function always starts at 1 and therefore cannot be restarted against the same schema until the schema has been refreshed. The Trickle Refresh Delay value sets the delay between each insert and delete with a default of 1 second ensuring that the refresh function does not place a significant load on the system, The Refresh Verbose value means that the virtual user running the refresh function reports on its activities.

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

Loading the Driver Script

Once you have selected and saved your driver options under the treeview and Driver Script select Load as shown in Figure 13.
Figure 13 Select Driver Script

This will populate the Script Editor window with the driver script shown in Figure 14.
These scripts provide the workload 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 values given in the EDITABLE OPTIONS section paying close attention to the scale factor that must match the scale factor used in the schema build. Additionally the driver scripts are regular HammerDB scripts and a copy may be saved externally and modified as you desire for a genuinely Open Source approach to load testing.

**Pre-Test 1 Running a Power Test Query**

In this example we will create a single virtual user and choose to display their output to verify the schema and database configuration. To do this Under the Virtual Users menu select from the Options menu or treeview as shown in Figure 15 select Virtual user Options and enter the number 1 for the Virtual Users. Also check the Show Output button to see what your users are doing whilst the test is running. Note that in contrast to the OLTP tests displaying the output will not reduce the overall level of performance as each query will be long-running and take significantly more time on the database server than is used by the client. The additional VERBOSE option within the script is available to provide a further level of detail, when complete click OK.
There are three other related options under the Virtual User Options dialogue, namely User Delay(ms), Repeat Delay(ms) and Iterations. Iterations defines the number of times that HammerDB should execute a script in its entirety. With regards to running the TPC-H driver script this can be thought of as the number of times a Virtual User logs on to the database, runs the defined number of query sets and logs off again. Consequently you already have an option for similar functionality within the TPC-H driver script itself to define the number of query sets and should therefore should use this instead. 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-H driver script the recommended approach is to leave the Iterations and User and Repeat Delays at the default settings. You also have the option to log the output, if you wish to generate a performance profile with the breakdown of the timing of each query you should also select this option. When you have completed the selection press OK. Click the Create Virtual Users button as shown in Figure 16 to create the Virtual User, it will be created but not start running yet.
Figure 16 Create Virtual Users

You can observe as shown in Figure 17 that the virtual user has been created but is showing a status of idle. You can destroy the Virtual User by pressing the Red Traffic light icon that has appeared in place of the Create Virtual Users button. To begin running the power test queries press the button Run Virtual Users as shown in Figure 19, the name of the button will appear in the information pane.
Figure 17 Virtual Users Created

You can observe the Virtual User icon change to signify activity. The Virtual User has logged on to the database and is running queries as can be observed in the Virtual User Output as shown in Figure 18. Note that the Queries do not run in sequence order from Query 1 to Query 22 and instead run according to a pre-defined random order depending on the virtual user number that you can observe defined within the script.
When the Virtual User has completed all of its designated query set it will exit showing a positive status as shown in Figure 19. Once the Virtual User is displaying this positive status it has logged off the database. 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 User can be destroyed to stop a running test, however will exit only after the current query has finished executing.
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. Within HammerDB you also have the option to convert and replay workloads traced using Oracle Trace as detailed in the documentation by Oracle Trace File Replay. As shown in Figure 20 this functionality is particularly useful for Query based workloads to capture and replay if desired an identical set of queries or to observe the workload that was run.
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 Optimal DOP and In-memory for Power and Throughput Tests**

Once the configuration has been verified the next stage is to focus upon performance. The best place to start with verifying performance is to determine optimal DOP and In-memory settings. To do this follow all of the steps for Pre-Test 1 and focus on modifying the value of the parameter “degree_of_parallel” as shown in Figure 21 before each run (Also make note of the parameters parallel_min and parallel_max servers and the optimizer_dynamic_sampling Oracle parameters as you may look to also modify these during pre-testing).
Additionally note that if you have enabled In-memory functionality this will also have a significant impact on performance testing. For example as shown in Figure 22 there will be background worker processes populating the in-memory column store on access when using a default environment and therefore may impact initial query times.

Figure 22 Background Worker Processes
It is recommended not to run the Refresh Function during this phase of testing.

Re-run the schema verification test with a single virtual user. This can calculate the optimal DOP for the Power Test, however note that you should also measure the DOP for the throughput test with multiple virtual users and choose the optimal value for the two types of test. If wishing to simulate a test environment approximating a TPC-H test the DOP must be set the same for both the Power and Throughput test.

To calculate the optimal DOP for the Power test start with a degree_of_parallel of 1 for the first test. Note that this setting as shown in Figure 23 will disable parallelism for the session. (A similar effect can be achieved system wide by setting parallel_max_servers to 0).
Do not ignore query performance tests with parallelism disabled when using in-memory features as modern CPU features significantly improve the query times achievable per core. After noting the query times with parallelism disabled proceed to testing with a degree of parallelism of 2 by modifying the value directly in the script. As shown in Figure 24 the number of parallel processes may appear as double that of your chosen value, this is normal behaviour “If inter-operation parallelism is possible, the total number of query servers can be twice the specified degree of parallelism”. And you should therefore account for this behaviour during testing.

<table>
<thead>
<tr>
<th>PID</th>
<th>USER</th>
<th>PR</th>
<th>NI</th>
<th>VIRT</th>
<th>RES</th>
<th>SHR</th>
<th>%CPU</th>
<th>%MEM</th>
<th>TIME+</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>2903</td>
<td>oracle</td>
<td>29</td>
<td>0</td>
<td>2404m</td>
<td>349m</td>
<td>344m</td>
<td>R</td>
<td>59.5</td>
<td>4.4</td>
<td>6:16.46 ora_p003_orcl</td>
</tr>
<tr>
<td>2885</td>
<td>oracle</td>
<td>29</td>
<td>0</td>
<td>2404m</td>
<td>416m</td>
<td>406m</td>
<td>R</td>
<td>57.5</td>
<td>5.1</td>
<td>6:16.67 ora_p002_orcl</td>
</tr>
<tr>
<td>2881</td>
<td>oracle</td>
<td>29</td>
<td>0</td>
<td>2406m</td>
<td>336m</td>
<td>332m</td>
<td>R</td>
<td>29.3</td>
<td>4.2</td>
<td>6:11.35 ora_p000_orcl</td>
</tr>
<tr>
<td>2883</td>
<td>oracle</td>
<td>29</td>
<td>0</td>
<td>2404m</td>
<td>264m</td>
<td>260m</td>
<td>S</td>
<td>25.9</td>
<td>3.3</td>
<td>6:10.53 ora_p001_orcl</td>
</tr>
</tbody>
</table>

Figure 24 Parallel Query

and note the time for the query set to complete and then repeat the test with a DOP of 2 and so on. Plotting the time taken for a query set in a graph will produce a result such as that shown in Figure 25.

![DOP Performance Graph](image)

Figure 25 Degree of Parallel

Note that although you have found the optimal DOP for a single user power test you should be aware that to produce this result may require significantly more resources than that achieved by a lower DOP with a slightly longer completion time. Therefore you should also calculate the DOP for the number of virtual users required for the throughput test at your chosen scale factor size by noting the maximum completion time of any of the multiple virtual users. For example Figure 26 shows the maximum completion time for the required 2 virtual users on a SF1 schema to be the 20 seconds of virtual user two.
To observe performance during the test you can use the Transaction Counter in exactly the same way as you use the transaction counter for OLTP tests. The Transaction Counter options be selected from the Options menu or the treeview and this displays the Transaction Counter Options as shown in Figure 27.

**Figure 27 Transaction Counter Options**

**Transaction Counter Options**

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

**Connect String**

The Connect String must be a standard format Oracle connect string for a user with permissions to read the
GV$SYSSTAT table, you can validate by logging on with this user using sql*plus.

```sql
SQL> select sum(executions) from gv$sqlarea where command_type = 3 and parsing_user_id in (select user# from sys.user$ where type# = 1 and astatus = 0 and name not in ('SYS','SYSTEM','SYSMAN','DBSNMP'))

SUM(EXECUTIONS)  
---------------  
     1  
```

A typical choice is the SYSTEM user however at version 12c permissions have changed and the system user no longer has permission to read from sys.user$ by default and therefore the HammerDB transaction counter will report an error of "SQL PARSE FAILED" as a result of the query failing as shown.

```sql
SQL> select * from sys.user$;
ERROR at line 1:
ORA-01031: insufficient privileges
```

To correct this behaviour logon as sysdba and grant read permission to the system user on sys.user$ as shown.

```sql
SQL> connect sys/oracle@pdb1 as sysdba  
Connected.  
SQL> grant read on sys.user$ to system;  
Grant succeeded.  
SQL> connect system/oracle@pdb1  
Connected.  
SQL> select sum(executions) from gv$sqlarea where command_type = 3 and parsing_user_id in (select user# from sys.user$ where type# = 1 and astatus = 0 and name not in ('SYS','SYSTEM','SYSMAN','DBSNMP'))

SUM(EXECUTIONS)  
---------------  
     116  
```

**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 Oracle database and the default value of 10 seconds is a good choice for an accurate representation.

**RAC Global Transactions**

The RAC Global Transactions checkbox determines whether values should be displayed only for the instance to which the transaction counter is connected or whether it is to display values for the entire cluster.

**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 exactly as done for running OLTP tests to begin observing the query rate. The transaction Counter will become active and start collecting throughput data as shown in Figure 28.
Figure 28 Waiting for Data

After the first refresh time interval you will be able to observe the transaction counter updating according to the throughput of your system. The actual throughput you observe for a single Virtual User will vary according to the capabilities of your system.

The main distinction between the transaction counter measurements for an OLTP test and a DSS test is that in this case the transaction counter will show the value for queries per hour. Note that this is not the same metric used to calculate the QphH value and as such is solely an indication of the query throughput of your system based on the number of SELECT calls processed. For larger schema sizes or smaller systems without significant resources you may find that the Query transaction counter does not register significant query throughput as the majority of the period of time is spent waiting for queries to be processed, this is normal and to be expected, additionally you should also not expect a flat transaction counter profile as would be expected from an OLTP workload as query times are significantly longer than transaction times.

Running the DSS Tests

To compile data similar to that collected for a TPC-H workload you need to run a Power Test including refresh functions following by a throughput test.

Power Test

A component of the TPC-H test is the refresh function and to adhere as closely as possible to a TPC-H test
the refresh function should be run either side of the Power Test. To enable this functionality HammerDB has a special power test mode, whereby if refresh_on is set to true as shown in Figure 29 and only one virtual user is configured then HammerDB will run a Power Test. Note that once you selected refresh_on for a single Virtual User in Power Test Mode the value of update_sets will be set to 1 and the value of trickle_refresh set to 0 and the value of REFRESH_VERBOSE set to false, all these values will be set automatically to ensure optimal running of the Power Test.

Figure 29 Enabling the Refresh Function

It is important to reiterate before running a Power Test that as shown in Figure 30 you cannot re-run the same refresh functions against the same schema due to constraint violations and therefore need to refresh the schema each and every time you run a refresh function. For this reason you should ensure that you have already followed the guidance previously given in this document to restore your schema to the original data.
Create a single Virtual User as shown in Figure 31 and ensure that you enable logging to capture all of the query times.

Create a single Virtual User as you did previously, you should see and accept a prompt that logging has been activated as shown in Figure 32.
Run the Virtual User as previous, you should now see that the Virtual User performs the New Sales Refresh a single Query Set and the Old Sales Refresh function which comprises a complete Power Test.

Fetch the data from the HammerDB log as shown below you should see the data for both the refresh functions as well as all of the query times that can be used to graph a performance profile.
Throughput Test

After the power test you should run the throughput test (if the refresh function has been run it is necessary to refresh the schema). For the throughput test you need to also run the refresh function however this time the aim is to trickle the refresh function slowly while multiple query streams are run. To find the correct number of query streams refer to the HammerDB DSS introduction document. Configure the options as shown in Figure 34 and therefore enable the refresh function, this time the update sets, trickle refresh and REFRESH_VERBOSE options will also be enabled when refresh_on is set to true.
Configure the correct number of Virtual Users to enable the first Virtual User to run the Refresh Functions and additional Virtual Users to run the Query Streams as defined in the specification for the test. Run the Virtual Users as shown in Figure 35, note that the Refresh Function will run more slowly as expected.
When the Virtual Users running the Query sets have completed the throughput tests, note the longest (not the shortest) time taken for a full query set to complete. As shown in Figure 36 this value is 26 seconds. You do not need to wait for the trickled refresh function to complete, however must have configured enough update sets to ensure that the refresh function remains running whilst the throughput test completes.
Calculating QphH

It is important to reiterate that HammerDB results are not official TPC-H results and cannot be compared with official HammerDB results in any way. Nevertheless you can gain a close approximation to official TPC-H data by entering the results of your Power and Throughput tests into the HammerDB TPC-H Calculator where the text is marked red as shown in Figure 37.
Figure 37 QphH Calculator

Coupled with performance profiles of query times this data provides the optimal way to test Oracle Query performance, parallel query and in-memory column store features.

**Support and Questions**

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