Open Source Development Labs
Database Test 3
Decision Support for Ad Hoc Queries
User's Guide

Version 1.5

Open Source Development Labs, Inc.
12725 SW Millikan Way, Suite 400
Beaverton, OR 97005
Phone: (503) 626-2455
Fax: (503) 626-2436
Email: info@osdl.org

Contributors to this white paper include:
Jenny Zhang (OSDL)
Mary Edie Meredith (OSDL)
Mark Wong (OSDL)

1 Introduction

This document provides instructions on how to set up and use the Open Source Development Lab's Database Transaction Test 3 (OSDL-DBT-3) kit. This kit provides what is needed to execute a workload similar to the TPC-H workload using SAP DB version 7.3.0.25 and PostgreSQL 7.4 and later. This document describes all other downloads the tester need to use SAP DB and downloads necessary to generate the data for loading the database. The queries in the query templates are tailored for SAP DB. The tester can alter the syntax for other databases of the tester's desire.

2 Setup

This section covers the steps of getting and installing an SAP DB database and the OSDL-DBT-3 test kit source code. Some documentation for PostgreSQL has been added to this document.

2.1 Getting and Installing the Files

2.1.1 SAP DB

The OSDL-DBT-3 test kit was developed using SAP DB Version 7.3.0.25, however, newer versions of the 7.3.0.x database should work with the test kit. Version 7.4, currently in beta on Linux systems, should require only minimal changes to the database parameters and possibly SQL syntax. Linux binaries can be obtained from the Web at:
http://www.sapdb.org/tgz_linux.htm

Although not required for OSDL-DBT-3, the source package along with installation instructions can be found on the Web at:
http://www.sapdb.org/develop/dev_linux.htm

If the tester wishes to install SAP DB with rpms, and if the your Linux distribution supports rpm packages, they can be retrieved from the Web at:
http://www.sapdb.org/rpm_linux.htm

The following rpm packages are required:

- sapdb-ind-7.3.0.25-1.i386.rpm

- sapdb-srv-7.3.0.25-1.i386.rpm

In case the old versions of SAPDB are removed from the SAPDB web site, they can be downloaded from the ftp site using the following command:

- ftp ftp.sap.com
- ftp> cd pub/sapdb/bin/linux
- ftp> get sapdb-srv-7.3.0.25-1.i386.rpm
- ftp> get sapdb-ind-7.3.0.25-1.i386.rpm
- ftp> get sapdb-ind-7.3.0.25-1.i386.rpm

The SAP DB related documentation can be found at:
http://www.sapdb.org/sap_db_documentation.htm

2.1.2 OSDL-DBT-3 Test Kit Source

The latest stable version of the kit can be found on SourceForge via:
http://sourceforge.net/project/showfiles.php?group_id=52479

For complete instructions on retrieving the latest tested source from CVS see the Web at:
http://sourceforge.net/cvs/?group_id=52479

This is an abbreviated set of steps to download only the source code:

- cvs -d:pserver:anonymous@cvs.osdldbt.sourceforge.net:/cvsroot/osdldbt login
- cvs -z3 -d:pserver:anonymous@cvs.osdldbt.sourceforge.net:/cvsroot/osdldbt co dbt3

When prompted for a password, simply hit the enter key.

2.1.3 DataGen

OSDL-DBT-3 data generation tool dbgen is based on the TPC-H DBGEN data generator tool. The query stream SQL generation tool qgen is based on the

TPC-H QGEN tool. These are freely available and need to be downloaded from http://www.tpc.org/tpch/. Look for downloads of DBGEN and QGEN. Section 2.2.2 will describe how to integrate these downloads into the DBT-3 kit.

2.1.4 Query Templates

The TPC-H DBGEN tool kit includes the official query syntax for TPC-H queries. The tester can substitute OSDL-DBT-3 queries with TPC-H queries and alter the syntax to fit other databases. A set of query syntax for SAP DB is included in the kit in the directory dbt3/datagen/original-queries. Some of the queries are rewritten. The query templates we used to run the test kit are under the directory dbt3/datagen/queries.

2.1.5 Setting Environment Variables

The following environment variables are required to be set and the set_run_env.sh under the respective database directory in scripts/ can be used to set these variables:

- DBT3_INSTALL_PATH = directory where OSDL-DBT-3 is installed (the directory where you did the cvs checkout in section 2.1.2 and the subdirectory dbt3. So if you installed in the directory /home/sapdb it would be set to /home/sapdb/dbt3)
- DSS_PATH = directory in which to build flat files.
- DSS_QUERY = $DBT3_INSTALL_PATH/directory in which to find query templates (usually in $DBT3_INSTALL_PATH/datagen/queries).
- DSS_CONFIG = $DBT3_INSTALL_PATH/directory in which to find configuration files (usually in $DBT3_INSTALL_PATH/datagen/dbgen/).
- SID = the database instance name (usually DBT3).

We recommend adding these to the user's login file. For example, in .bash_profile, add the following lines:

- DBT3_INSTALL_PATH=/home/sapdb/dbt3

- DSS_PATH=/dbt3_data
- DSS_QUERY=$DBT3_INSTALL_PATH/datagen/queries
- DSS_CONFIG=$DBT3_INSTALL_PATH/datagen/dbgen/
- SID=DBT3

SAP DB Notes To run the kit with SAP DB, the user sapdb is required. The user sapdb is the owner of all the files and directories.

OSDL-DBT-3 assumes the SAP DB command line interface dbmcli is added to the user path. To add the path, edit the user sapdb profile. Locate variable PATH definition and add:
/opt/sapdb/depend/bin:/opt/sapdb/indep_prog/bin

SAP DB uses the x_server to handle the communications between the database server and clients. X_server has to be started at the beginning. We recommend adding the following to the system boot process. For example, in /etc/rc.d/rc.local, add the following:
/etc/init.d/sapdb start

If raw devices are used (which is highly recommended for optimal database performance), it is a good idea to bind the raw devices in boot process. For example:
raw /dev/raw/raw1 /dev/rd/c0d13p1

2.1.6 Compiling the Kit

The entire DBT-3 kit can be compiled by running make in the top level directory.

2.1.7 Building the Database

A OpenOffice.org spreadsheet is provided to aid in sizing the database under the respective database directory in doc/:
dbt3_sizing.sxc

The parameter Scale Factor can be found under the Performance tab. The scale factor can actually be any decimal number (like 1.1 or 50) so that if the tester wishes, the tester can create a database whose size is not one of the officially permitted scale factors. This might be desirable for development purposes. Any results should be advertised with the scale factor used, since the performance

varies based on the amount of data required for processing the queries. The database size is defined with reference to scale factor. For example, for scale factor 1, the database size is roughly 1 GB.

Note: This kit does not support scale factors less than 1. Although you can build a database using scale factors less than 1, the query generator (qgen) will not generate the proper variable values that correspond to scale factors less than 1.

The data displayed under the database size tab report the expected size of the database, for the parameters entered, are to be used for the physical database layout. The number to note is the total number of 8 KB pages since SAP DB uses those units in most of its configuration settings. Keep in mind that additional space must be added to account for auxiliary structures such as indexes, temporary tables and 1% for inserts.

The tester needs to allocate space for the flat files generated by dbgen used to load the database. Under the tab database size in file doc/dbt3_size, one column Flat File Size is provided to calculate the space. Once the database is loaded and backed up, there is no need to retain these flat files.

The tester needs to allocate a file system area large enough to backup the database, which is part of the database build process (a required part for SAP DB). The database backup needs to be at least as large as the database.

2.1.8 Creating the input data files

The data files can be generated using the dbgen under datagen/dbgen:
./dbgen -s <scale_factor>

2.1.9 Creating the input data files

The ../dbt3/scripts/ directory contains a list of databases that this test kit supports. Only SAP DB is currently supported and is found under the ../dbt3/scripts/sapdb subdirectory.

The tester needs to adjust some of the files in this directory to fit the environment. If the tester already has a database named DBT3, then the tester need to change the environment variable"SID" to another name.

Under each of the database directories (e.g. ../dbt3/scripts/sapdb/) is a file called build_db.sh which is used to build the database (and optionally to generate

the input data).

Database Creation

PostgreSQL It's recommended to run the complete workload using the run_workload.sh since that also involves creating the database, but the database creation script for PostgreSQL will be briefly described here.

scripts/pgsql/load_test.sh [-f <scale_factor> -p '<database_parameters>' -y <0/1>
-o <dir>]

-f	Scale factor to build.
-o	Directory to write test output into.
-p	Any PostgreSQL database parameters to use.
-y	0 to disable oprofile [default], 1 to enable.

SAP DB This script has the following usage:
./build_db.sh [-g <scale_factor>]

build_db.sh Performs these set up operations:

- If -g is given as a command line parameter, it will generate the data files
- Create the database
- Create the tables
- Load the database
- Create the indexes
- Update the optimizer statistics
- Back up the database
- Set database internal parameters

To do this build_db.sh calls the following scripts found in the same directory:

./drop_db.sh	drops the database (for a reload)
./create_db.sh	creates the database, devspaces, users
./create_tables.sh	creates the tables
./load_db.sh	loads the tables from flat files
./create_indexes.sh	creates indexes
./update_statistics.sh	update optimizer statistics
./backup_db.sh	creates backup that can be used by restore_db.sh
./set_params.sh	set the database internal variables

Adjusting the script create_db.sh The shell script, scripts/sapdb/create_db.sh, creates the physical database. The SAP DB database users dbm, dba, and dbt are created, with passwords the same as the user name, by create_db.sh. The user dbm is the database administrator account, dba is the utility account, and dbt is the user account. The dbt account is used when running the programs built in the test kit, or to execute SQL using the dbmcli program included with SAP DB. This script also sets the database parameters param_adddevspace that define the devices on which the database resides and the other parameters that allow the database to take advantage of the processors and memory on the system. Before running duild_db.sh, these lines should be tailored for the system.

For creating the database, the tester needs to alter the create_db.sh script to fit the number of disks and the size of the logs and the data files to match the scale factor as computed in section 2.5.1. The directory where the System file resides is defined by the variable 'SYS_DIR'. Change this variable to fit the your system. The devspaces are defined by the following statements:

```
param_adddevspace 1 SYS $SYS_DIR/$SID/$SID_SYS_001 F
param_adddevspace 1 DATA $SYS_DIR/$SID/$SID_DATA_001 F 524228
param_adddevspace 1 LOG $SYS_DIR_DIR/$SID/$SID_LOG_001 F 8192
```

The first line defines a file system and file for locating the System file, which contains database metadata. The second and third lines defines space (devspaces) for the data and the log. Although these lines currently put all the devspaces under one directory $SYS_DIR and point to files in a file system, a more performant configuration would be using raw partitions (substitute "R" for "F" for raw devices) and multiple DATA entries for multiple data files.

The database is automatically spread across the data files at load time. It reads/writes from multiple devices simultaneously. One should never use two partitions from the same physical disk device.

If the tester uses raw devices, the tester needs to bind them first. For example, if the raw device is /dev/raw/raw1 on partition /dev/sdc1, the tester need to execute the following command as root:

```
raw /dev/raw/raw1 /dev/sdc1
```

The last number of the param_adddevspace entry (524228 and 8192) defines the size in 8K pages for DATA and LOG devspaces. All DATA devspaces should contain the same number of pages. Use the database total page sizes computed in section 2.5.1 for the scale factor the tester is using, divide by the number of disks , add about 15temporary file space (the tester may need more depending on the load the tester achieves) and insert that number into the script.

The log size can remain as is, as we are using DEMO logging, which simply reuses

the log without backing up the log. For optimal performance the log device needs to be on a different physical disk from the DATA and SYS devspaces and needs to be a raw partition .

An adjusted parameter set for eleven DATA devspaces on raw partitions might look like the following:

```
param_adddevspace 1 SYS /sys/$SID/SYS_001 F
param_adddevspace 1 DATA /dev/raw/raw2 R 204800
param_adddevspace 2 DATA /dev/raw/raw3 R 204800
param_adddevspace 3 DATA /dev/raw/raw4 R 204800
param_adddevspace 4 DATA /dev/raw/raw5 R 204800
param_adddevspace 5 DATA /dev/raw/raw6 R 204800
param_adddevspace 6 DATA /dev/raw/raw7 R 204800
param_adddevspace 7 DATA /dev/raw/raw8 R 204800
param_adddevspace 8 DATA /dev/raw/raw9 R 204800
param_adddevspace 9 DATA /dev/raw/raw10 R 204800
param_adddevspace 10 DATA /dev/raw/raw11 R 204800
param_adddevspace 11 DATA /dev/raw/raw12 R 204800
param_adddevspace 1 LOG /dev/raw/raw1 R 8192
```

The create_db.sh script calls the script ./set_param.sh to set database parameters. There are cases (the backup and restore) where some settings have to be adjusted. When that happens, set_param.sh is executed again to restore the database to its original settings. So it is very important to enter all database parameter settings in this file.

The only lines that you should touch in the set_param.sh file are those that begin with "param_put". There are two sets of parameters in the file. The first are for database load execution. The second set is for the performance runs. Normally the param_set statements should be the same for both.

The following describes those parameters you need to adjust initially based on your system configuration.

When executing queries DATA_CACHE (8k pages of shared memory) needs to be as large a possible on the your system, allowing for memory for the database kernel and other database caches.

It is hard to pre-compute what the total database memory utilization will be, so the tester may have to experiment with DATA_CACHE (and the other cache settings) to get the optimal size. Do not forget to adjust shmmax in the operating system to allow for a large shared memory area, and allow enough swap area if the your OS pre-allocates swap area for an application before allocating memory.

Adjust the script for the number of processors on the your system by editing

the following:
    param_put MAXCPU 2

Systems with Intel processors supporting hyperthreading should use double the number of physical processors.

Calculate the MAXUSERTASKS to be a multiple of MAXCPU and to be at least as large as the number of streams required for the your chosen scale factor:
    param_put MAXUSERTASKS 16

Modifying load_db.sh Currently the load_db.sh scripts loads the tables serially rather than in parallel. This takes longer, but results in the data from tables to be co-located on the disk (for each data devspace) as they are loaded one by one. Loading in parallel is much faster, but the performance characteristic could be very different. For example the scan of a table loaded in parallel could take longer since it would likely be spread across a larger area. A serially loaded table, would not be interspersed with pages from other tables.

The tester can change the script so that it loads the tables in parallel by putting the load commands in background and waiting at the end.

Modifying create_indexes.sh and create_tables.sh The TPC-H specification allows only certain indexes to be created. The script create_indexes.sh defines all of these indexes. The testers can modify this script to improve performance.

Note that SAP DB is somewhat unusual in that every table is built as a b-tree index, based on the primary key or alternatively a system generated key if a primary key is not specified. With most other databases the tester would normally build a base table and separately build a unique key for the primary key column. It can be less efficient to do so in SAP DB. As the base table will be built as a b-tree anyway, the tester might as well make the key the primary key rather than a system generated key. This approach has proven to be the best case for update intense activities. However, since DBT-3 is query intensive, one might want to try different variations to see how query performance is effected. The script create_tables.sql currently has primary key defined.

Adjusting the script define_medium.sh, backup_db.sh and restore_db.sh The build script build_db.sh will backup the database as the final step using the script backup_db.sh. Since the backup is roughly the size of the database, the tester need to edit the script define_medium.sh and point backup medium to the directory where the tester want to put the backup.

The tester may also need to split the backup into multiple locations or into smaller files for convenience. For example, if the tester wants to put the backup on two directories /dbt3_backup1 and /dbt3_backup2, and the first medium has 500 8K pages, the medium definition should be:

medium_put data1 /dbt3_backup1/datasave1 FILE DATA 500 8 YES
medium_put data2 /dbt3_backup2/datasave2 FILE DATA 0 8 YES

The backup commands in backup_db.sh for two locations becomes:

backup_start data1 migration
backup_replace data2

Also the tester need to change the commands in restore_db.sh to:

recover_start data1
recover_replace data2

The incremental backup/restore is added in the scripts so that the database is writable after a full restore.

Executing the script build_db.sh Assuming the tester has already:

generated the data files
have modified create_db.sh, create_indexes.sh, create_tables.sh and backup_db.sh (and restore_db.sh) for the your environment
have created the environment variables needed by the database

the tester is now ready to execute the database build:

cd .. /dbt3/sapdb/scripts
nohup ./build_db.sh > build_db_$SID .out 2>&1 &

If this is the first time for creating the database, ignore "database not found' messages during the "drop database" phase.

At the conclusion of the build_db.sh script, the database should be left in the "warrm" state ready for query activity. To confirm this, execute:

/opt/sapdb/depend/bin/dbmcli -d DBT3 -u dbm,dbm db_state

The result should be:

OK
State
WARM

Restoring the database If the database is already backed up, the tester can chose to restore the database using the script restore.sh. As described

in 2.5.3.5, the tester must check that the mediums are correctly defined in defime_medium.sh.

3 Running the Test Kit

Before running the kit, testers should begin by creating the database in a way that fits their system configuration environment, as described in Section 2. Part of the test will execute a database load step and time it, but it is best to do it first in a controlled manner. Using these procedures, the tester can experiment with database customizations to tune load or query performance. It is up to the tester to do so in a manner consistent with the TPC-H specifications, if that is important to the tester.

Under dbt3/dbdriver/scripts there are several scripts that are used to drive the database workload.

Testers can choose to run all the tests in OSDL-DBT-3 as well as part of the tests. The following section describes how to run all the tests in OSDL-DBT-3. The other scripts are explained in section 4.

Tester may also create several DBT-3 databases so that several scale factors can be tested or various implementation strategies compared. They will only need to change environment variables to point to the correct database as described in Section 2.2 prior to executing the test kit scripts.

3.1 Run DBT-3

3.1.1 PostgreSQL

To run all three tests (load, power, throughput), execute the scripts/run_workload.sh script.

scripts/pgsql/run_workload.sh [-f <scale_factor> -g -p '<load database parameters>' -q '<power database parameters>' -r '<throughput database parameters>' -y

-e Flag to get EXPLAIN ANALYZE results instead of query results.
-f Database scale factor to use, 1 by default.
-g Build database data files.
-n Number of streams to use, 1 by default.
<0/1>] -o Flag to enable oprofile.
-p Any PostgreSQL database parameters to use for the load test.
-q Any PostgreSQL database parameters to use for the power test.
-r Any PostgreSQL database parameters to use for the throughput test.
-v Flag for verbose script output.

3.1.2 SAP DB

To run all the tests in OSDL-DBT-3, change directory to $DBT3_INSTALL_PATH/dbdriver/scripts and execute the command:

./run_dbt3.sh <scale_factor> <num_stream> [seed]

Note: run_dbt3.sh executes one load test and two performance tests. It requires that $DBT3_INSTALL_PATH and $DSS_QUERY environment variables be defined as described in Section 2.2. It also assumes that the input data files for the load have been previously generated as described in Section 2.5.2 and that $DSS_PATH points to the directory containing those input files (or links to them).

<scale_factor>: is the scale factor of the run. It should be the same as the one used when generating data files.

<num_stream>: is the number of streams in throughput test. The minimum required stream count for throughput test is listed in the following table:

SF	Streams
1	2
10	3
30	4
100	5
300	6
1000	7
3000	8
10000	9

[seed]: is optional. If this parameter is specified, the run will use this as the seed. Otherwise a seed is generated according to system time.

Example:

./run_dbt3 1 2

starts OSDL-DBT-3 run with scale factor 1, 2 streams in throughput test.

The system time is used as seed.

Note: It is a good idea to check the results of all queries. If any report "no rows returned" then the test is invalid. By running the kit first with scale factor 1, you can check the resulting output against those given in the TPC-H specification to be certain no bugs exist in the database or the kit install. This is especially useful if you rewrite the SQL for any of the queries and you want to confirm its correctness.

If several databases are created, the tester can change the environment variable $SID to point to the database under test. Be aware that the script parameter <scale_factor> need to be changed to corresponding database scale factor.

3.2 Run Sub-Test and Other Tools

In addition to running the whole OSDL-DBT-3, testers can run parts of the test. Except for the load test, all other tests assume the database is built.

3.2.1 Run the Load Test

The load test includes building the database and all activities required to bring the database to the configuration that immediately before the beginning of the performance test. To run the load test, change directory to dbdriver/scripts and execute the command:

./run_load_test.sh <scale_factor>

<scale_factor>: is the scale factor of the run. It should be the same as the one used when generating data files.

Example:

./run_load_test.sh 1

builds the database of scale factor 1.

3.2.2 Run Performance Test

The performance test involves the power test and the throughput test.

To run the performance test, change directory to dbdriver/scripts and execute the command:

./run_perf_test.sh <scale_factor> <perf_run_number> <num_stream>

<scale_factor>: is the scale factor of the run. It should be the same as the one used when generating data files.

<perf_run_number>: is the performance run number (1 or 2).

<num_stream>: is the number of streams in throughput test. For details, refer to section 3.1

Example:
./run_perf_test.sh 1 1 2
starts a performance test 1 with scale factor 1, 2 streams in throughput test.

3.2.3 Run Power Test

The power test includes one refresh function 1, one query stream and one refresh function 2.

To run the power test, change directory to dbdriver/scripts and execute the command:
./run_power_test.sh <scale_factor> <perf_run_number>

<scale_factor>: is the scale factor of the run. It should be the same as the one used when generating data files.

<perf_run_number>: is the performance run number (1 or 2).

Example:
./run_power_test.sh 1 1
starts a power test with scale factor 1.

3.2.4 Run Refresh Function 1

To run the refresh function 1(inserts), change directory to dbdriver/scripts and execute the command:
./run_rf1.sh <scale_factor>

<scale_factor>: is the scale factor of the run. It must be the same as the one used when generating data files (otherwise the insert keys may not match those appropriate for the database).

Example:
./run_rf1.sh 1

starts refresh function 1. If the insert data are not generated, the insert data will be generated for scale factor 1.

3.2.5 Run Refresh Function 2

To run the refresh function 2 (deletes), change directory to dbdriver/scripts and execute the command:

./run_rf2.sh <scale_factor>

<scale_factor>: is the scale factor of the run. It must be the same as the one used when generating data files (otherwise the keys may not match those in the database).

Example:

./run_rf2.sh 1

starts refresh function 2. If the delete data are not generated, the delete data will be generated for scale factor 1.

3.2.6 Run Query Stream for Power test

This script parses the query templates, generate query stream and executes the queries.

To run the query stream, change directory to dbdriver/scripts and execute the command:

./run_power_query.sh <scale_factor> <perf_run_number>

<scale_factor>: is the scale factor of the run. It should be the same as the one used when generating data files.

<perf_run_number>: is the performance run number (1 or 2).

Example:

./run_power_query.sh 1 1

starts a query stream for the power test.

3.2.7 Run Throughput Test

The throughput test executes certain number of query streams and update streams.

To run the throughput test, change directory to dbdriver/scripts and execute the command:

./run_throughput_test.sh <scale_factor> <perf_run_number> <num_stream>

<scale_factor>: is the scale factor of the run. It should be the same as the one used when generating data files.

<perf_run_number>: is the performance run number (1 or 2).

<num_stream>: is the number of streams in throughput test. For details, refer to section 3.1

Example:

./run_throughput_test.sh 1 1 2

starts 2 parallel query streams and 2 refresh streams.

3.2.8 Run Query Stream for Throughput Test

This script parses the query templates, generate query stream and executes the queries.

To run the query stream, change directory to dbdriver/scripts and execute the command:

./run_throughput_query.sh <scale_factor> <perf_run_number> <stream_num>

<scale_factor>: is the scale factor of the run. It should be the same as the one used when generating data files.

<perf_run_number>: is the performance run number (1 or 2).

<stream_num>: is the stream number of this query stream. Valid steam numbers depend on the scale factor chosen (see Section 3.1).

Example:

./run_throughput_query.sh 1 1 2

starts the second query stream for the throughput test.

3.2.9 Run Refresh Stream for Throughput Test

This script executes a pair of refresh functions by calling run_rf1.sh and run_rf2.sh.

To run the refresh stream, change directory to dbdriver/scripts and execute the

command:
./run_refresh_stream.sh <scale_factor> <stream_num> <perf_run_number>

<scale_factor>: is the scale factor of the run. It should be the same as the one used when generating data files.

<stream_num>: is the stream number of this query stream.

<perf_run_number>: is the performance run number (1 or 2).

Example:
./run_refresh_stream.sh 1 2 1
starts the second refresh stream for the throughput test.

3.2.10 Test a single query.

Testers may need to test a change to the standard SQL for a specific query prior to executing the entire query set. Query templates are located in the directory defined by the environment variable $DSS_QUERY.

The queries are numbered 1 through 22 with the SQL in files named N.sql, where N is the query number. These files are generated by Qgen (from the TPC web site) and have substitution parameters indicated by a colon followed by a parameter number (:1 for example). To execute a query, these parameters are substituted for valid values, then the SQL is formatted to be accepted by the SAP DB command line tool, dbmcli. Once you have modified a given query, say Query 20, cd to directory $DBT3_INSTALL_PATH/dbt3/dbdriver/scripts then execute:
./run_single_query.sh <SF> 20

where <SF> is the scale factor you used to build the database. The scale factor is very important, since only certain parameter value ranges are valid for any given scale factor. The tool will search for the file 20.sql in $DSS_QUERY, substitute appropriate parameters, execute the query, display the data resulting from the query, and return the elapsed time in seconds used to execute the query.

3.2.11 Change the query set.

The kit assumes that you have 22 queries in the directory $DSS_QUERY. If you wish to run a workload other than the official set of queries, you may do so, as long as you name the queries 1.sql through 22.sql. If you have fewer than 22 queries, create blank files for those you do not use, and they will be effectively

ignored. If you want to run more than 22 queries, you will need to patch QGEN (see dbgen subdirectory from the download of DBGEN from www.tpc.org).

If the tester rewrite any of the official queries, it is a good idea to verify the query on a database of scale factor 1. The TPC-H specification has query validation results for each query.

4 Test Results

Under the directory $DBT3_INSTALL_DIR/dbdriver/scripts, several scripts are provided to calculate the performance metric:

- q_time.sh queries the table time_statistics, and returns the elapsed time for each individual query or refresh function as well as each test (for example, power test, throughput test, etc.)
- get_power.sh queries the table time_statistics and calculates the query processing power at the chosen scale factor. The units of Power@size are Queries per hour * Scale-Factor. Using this script requires a complete power test.
- get_throughput.sh queries the table time_statistics and calculates the throughput numerical quantity at the chosen scale factor. The numerical quantity is defined as the ratio of the total number of queries executed over the length of the measurement interval. The unit is of Throughput@size are Queries per hour * Scale-Factor. Using this script requires a complete throughput test.
- get_composite.sh calls ./get_power.sh and ./get_throughput.sh and calculates the composite query-per-hour performance metric. The units of Composite Query-Per-Hour@size are Queries per hour * Scale-Factor.