# TPC-DS 2.**11**.0 “how to” Guide

## Introduction

The purpose of this guide is to describe the most common tasks necessary to implement a TPC-DS benchmark. The target audience is individuals who want to install, populate, run and analyze the database, queries and data maintenance workloads for TPC-DS.

This guide does not discuss anything related to actually publishing TPC-DS results (e.g.: tuning, data layout, pricing, auxiliary data structures, auditing, etc.).

For more details, see the official TPC-DS specification.

## Change Log

|  |  |  |
| --- | --- | --- |
| Date | Version | Comments |
| Feb. 8, 2007 | 1.0 | Initial draft for kit 1.1.52 (Doug Johnson/Netezza) |
| Feb. 22, 2007 | 1.1 | Added comments from Meikel Poess/Oracle and Shirley Wang/MS |
| Oct. 9, 2015 | 1.2 | Made changes to reflect TPC-DS V2 Specification |
| Apr, 23, 2020 | 1.3 | Combine missing parts from original version  add paths to files you probaby have to modify |

## How to download the latest kit

The kit is downloadable from [www.tpc.org](http://www.tpc.org/), using the link for the TPC-DS benchmark.

## What’s in the kit

There are hundreds of files in the kit but only a few that you have to read or modify. Here’s the list of files that you should read:

1. /specification/specification.docx: latest specification document

Here’s the list of files that you will probably have to modify:

* 1. /tools/tpcds.sql: file with SQLcd to create the data warehouse tables. You will probably need to add allocation or distribution information specific to your database technology and configuration.

1. /tools/tpcds\_source.sql: file with SQL to create the source/staging tables. You will probably need to add allocation or distribution information specific to your database technology and configuration.
2. /tools/Makefile.suite: file with Unix-style make commands to build the dsdgen and dsqgenbinaries for a target platform. You will need to specify the execution environment (see make/build section below).
3. vcproj files: files used to build the tools for Windows Visual C++.
4. /query\_templates: folder with query template files (.tpl). You don’t manually change these files but the dsqgen program transforms them into executable SQL.

## How to make/build the binaries

### For AIX, LINUX, HPUX, NCR and Solaris

1. Copy Makefile.suite to Makefile
2. Edit Makefile and find the line containing “OS = “
3. Read the comments and append your target OS. For example: “OS = LINUX”
4. Run “make”

### For Windows

1. Install Microsoft Visual Studio 2005 or later
2. Open the entire solution by double-clicking on dsdgen.sln (you might see an error saying “project file grammar.vcproj” has failed to load; you can safely ignore this error)
3. From list of projects, right click on dsdgen and select “build” (or from top menu, Build -> Build Solution). This will build mkheader and distcomp before building dsdgen.
4. Repeat step 3 for building dsqgen
5. To cross compile for X64 and IA64 on X86 platforms, install Microsoft Visual Studio 2005 “Team Suite” SKU, modify the target platform from Build -> Configuration Manager, and repeat steps 3 & 4.

## How to generate the load data

The dsdgen utility generates input data to (a) load the initial data warehouse and (b) “refresh” the data warehouse for the data maintenance workload. This section describes how to generate the load data.

Run “dsdgen –h” for the help info. Note that many of the options are “advanced” and usually not needed.

Example to generate the load data files for a 100GB in the /tmp directory:

dsdgen –scale 100 –dir /tmp

The official scale factors are 1TB, 3TB, 10TB, 30TB and 100TB.

The output files will be of the form “<table>.csv”. Even though file suffix is “.csv”, the default field delimiter is ‘|’. Use the “-delimiter ‘<c>’” option to change delimiters.

Since dsdgen generates 200-300GB/hour serially on a 2-3GHz x86 processor, it is useful to run multiple parallel streams when generating large amounts of data. Here’s an example for generating 100 GB with 4 parallel streams simultaneously on Linux/Unix :

dsdgen –scale 100 –dir /tmp –parallel 4 –child 1 &

dsdgen –scale 100 –dir /tmp –parallel 4 –child 2 &

dsdgen –scale 100 –dir /tmp –parallel 4 –child 3 &

dsdgen –scale 100 –dir /tmp –parallel 4 –child 4 &

Note that dsdgen always reads the “tpcds.idx” file so if you run it from somewhere other than the “kit” directory, then you need to copy tpcds.idx to the current directory.

## How to load the data

Run the loader provided with your DBMS to load the dsdgen generated data files into the data warehouse tables.

Note that the default delimiter is ‘|’ so you may need to specify a different delimiter with dsdgen or the loader if the defaults don’t match.

Also, the default “null” value is “||” so if your loader expects (for example), “|NULL|”, then you will need to override the loader’s value for nulls.

## How to generate query SQL from templates

The “dsqgen” utility is used to transform the query templates (see query\_templates/\*.tpl) into executable SQL for your target DBMS. The unmodified templates are not executable.

Run “dsqgen–h” for the help info. Note that many of the options are “advanced” and usually not needed.

Since some common SQL features do not have ANSI standard forms (e.g. “LIMIT” and “BEGIN/COMMIT”), the dsqgen utility must be told which “dialect” to use. The following “dialect templates” are supported: db2.tpl, netezza.tpl, oracle.tpl, sqlserver.tpl. The following example generates a SQL file (named query\_0.sql) from the query99 template for a 100GB database using Oracle syntax.

dsqgen –template query99.tpl –directory query\_templates –dialect oracle –scale 100

Note that dsqgen also reads the “tpcds.idx” file (in the “kit” directory) and the “ansi.tpl” file (in the “kit/query\_templates”) so you’ll need to copy something somewhere.

## How to run the queries

You can, of course, run the queries any way you want but the “official” method is to run N concurrent query streams where N is a function of the database size. Unlike TPC-H, the TPC-DS query workload does not have a single-stream query component. The relationship between database size (SF) and query streams (N) is:

|  |  |
| --- | --- |
| **SF** | **N** |
| 100 | 7 |
| 300 | 9 |
| 1,000 | 11 |
| 3,000 | 13 |
| 10,000 | 15 |
| 30,000 | 17 |
| 100,000 | 19 |

dsqgen –scale <sf> -dir <path to query templates> -streams <n>

Example:

dsqgen –scale 100 -dir /tmp –streams 5

## How to generate the refresh data

Run “dsdgen –h” for the help info. Note that many of the options are “advanced” and usually not needed.

Example to generate the refresh data files in /tmp for the 3rd “update” stream:

dsdgen –scale 100 –dir /tmp –update 3

The output files will be of the form “s\_<table>\_<stream>.csv”. The default field delimiter is ‘|’. Use the “-delimiter ‘<c>’” option to change delimiters.

Since dsdgen only generates 200-300GB/hour (on a 2-3GHz x86 processor), it is useful to run multiple parallel streams when generating large amounts of data. Here’s an example for the 3rd stream/child of 10 parallel streams:

dsdgen –scale 100 –dir /tmp –update 3 –parallel 10

–child 3

Note that dsdgen always reads the “tpcds.idx” file so if you run it from somewhere other than the “kit” directory, then you need to copy tpcds.idx to the current directory.