Assignment 1

Testing Resources

Last updated: **Friday 23rd February 2:51am**

Most recent changes are shown in red ... older changes are shown in brown.

This document describes a testing framework for determining whether your implementation of the `PersonName` data type is correct. This testing framework contains a subset of the tests that will be made in auto-marking. If you pass all the tests here, you ought to pass the majority of the auto-marking tests. Note, however, that passing all of these tests does *not* guarantee that you will pass *all* of the auto-marking tests.

The first step in setting up the testing framework is to make a testing directory under your vxdb local storage:

```
$ ssh nw-syd-vxdb
$ source /localstorage/$USER/env
$ mkdir /localstorage/$USER/testing
```

The next step is to set up the testing infrastructure:

```
$ cd /localstorage/$USER/
$ tar -xf /web/cs9315/24T1/assignments/ass1/testing/testing.tar
```

This sets up a number of files and directories under the `testing` directory. Note that some of these are actually symbolic links to our files so that (a) you can't change them, and (b) you save storage space under your own directory.

The `tar` command creates (among other things) the following:

- `run_test.py` a Python script that will run all of the tests; use this after you have gotten your implementation of `PersonName` into a reasonable state; this is a symlink to a script in the COMP9315 account
- a `tests/` subdirectory, containing all of the test files; this is a symlink to files under the COMP9315 account, so all of the tests are read-only unless you copy them manually to your `testing/` directory

There are three schemas provided with the testing framework, and each schema has corresponding data sets.

```
Users(realname::PersonName)
```

```
Students(id::int, name::PersonName, program::int, plan::char[6])
```

```
Students(id::int, name::text, program::int, plan::char[6])
```

Under `tests/`, you will find three subdirectories:

`0_sanity-checks`

Doesn't contain data to be stored. Assumes that you have an (empty) database with a `PersonName` data type loaded, and allows you to run a bunch of simple checks on values of type `PersonName`. Useful for testing your parsing function `pname_in()`, your output function `pname_out` and your operators.

`1_users`

Contains a schema and database for a table with just one attribute of type `PersonName`. There are three data files of differing sizes. You should create a new database, add the schema and then insert the data from one data file. There are a series of tests to check whether all of the data was loaded and can be manipulted correctly.

`2_students`
Contains a schema for a table of student data. There are four data files ranging in size from 100 tuples to 100000 tuples; each data file is a superset of the preceding data file. The queries test a wide range of `PersonName` functionality, including indexing. You should definitely check whether your code can deal with `data4.sql`; some bugs don't manifest themselves until you try with large amounts of data.

You can look at the tests in these directories and copy-paste them into a `psql` session if you want to perform individual tests. Or you can run a comprehensive set of tests using the Python script described below. Each directory contains an `info.txt` file giving more details about the files in the directory.

Next, you should move your `pname.c` and `pname.source` files into the new testing directory.

```
... this is the probable location that your files are in ...
... if they are somewhere else copy from that location instead ...
$ cp postgresql-15.6/src/tutorial/pname.c testing/pname.c
$ cp postgresql-15.6/src/tutorial/pname.source testing/pname.source
```

From this point on, you can do your assignment development in the `testing/` directory. The provided `Makefile` will do the same job as the one in the `postgresql-15.6/src/tutorial/` directory. Don't forget to run the command:

```
$ source /localstorage/$USER/env
```

and (re)start your PostgreSQL server every time you want to do a development/debugging session on your `PersonName` data type.

You can now run the tests.

```
$ ssh nw-syd-vxdb
$ source /localstorage/$USER/env
$ p1
$ cd /localstorage/$USER/testing
$ ./run_test.py
... internal logs from postgresql are placed in pg.log ...
... external logs from the testing script are placed in test.log ...
... start by reading test.log after run_test.py has finished ...
```

You *can* download and run the tests at home with this link (**WARNING**: contains a very large file). We won't provide assistance in setting up your home environment for doing this assignment; it's your problem to work it out.

One thing you will need to do is edit `run_tests.py` and remove the check that you are on `vxdb` at the start of `main()`. And change the check for a `postgresql-15.6` directory to suit local conditions.

NOTES:

- `pname.source` should do the bare minimum needed to create the PersonName type as specified.
- `pname.source` should **not** try to create any tables, insert any data, or drop the `PersonName` type.
- Everything done in `pname.source` *should* be undo-able with "DROP TYPE PersonName CASCADE".
  (but `pname.source` should not actually do this ... see previous point)
- These tests remove some information like timing, paths, and width from output to allow for comparison to expected output.
- Do your own tests, especially to make sure that queries are completed in a reasonable time and you are storing data in a reasonable amount of space.

Please report any issues ASAP on the Webcms3 comments for this page.

Have fun, *jas*