COMP9315 23T1 DBMS Implementation

Assignment 1 Testing

More test cases will be used in marking

Last updated: **Friday 10th March 11:33pm**

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

Back to the specification

[Mar 10th Update]: A problem in the test script, which might induce the test error if you put longitude before latitude when displaying the output, is corrected. Please download and test with the latest version.

This document describes a useful testing framework for determining whether your implementation of the `GeoCoord` 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.

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
$ p1
... start your database server ...
$ mkdir /localstorage/$USER/testing
```

The next step is to copy main testing scripts. You can download the script for local usage by clicking the path.

```
$ cp -P /web/cs9315/23T1/assignment/1/testing/run_test.p
$ cp -P /web/cs9315/23T1/assignment/1/testing/Makefile /
```

The testing script `run_tests.py` is written for the `vxdb` server. To use it in your local environment, you will need to edit `run_tests.py` and remove the check that you are on `vxdb` at the start of `main()`.

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

```
Locations(lid::int, coord::GeoCoord);
StoreInfo(sid::int, name::text, noproduct::int, coord::(
```

The next step is to get the test files from the class directory.

```
$ cd /localstorage/$USER/testing
$ tar -xf /web/cs9315/23T1/assignment/1/testing/testing
```

The above `tar` command creates a subdirectory called `tests/` in your `testing/` directory. 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. 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 `GeoCoord` data type loaded, and allows you to run a bunch of simple checks on values of type `GeoCoord`. Useful for testing your parsing function `gcoord_in()`, your output function `gcoord_out` and your operators.

`1_locations`

Contains a schema and database for a table with just one attribute of type `GeoCoord`. There are two 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_storeinfo`

Contains a schema for a table of student data. There are three data files ranging in size from 100 tuples to 10000 tuples; each data file is a superset of the preceding data file. The queries test a wide range of `GeoCoord` functionality, including indexing. You should definitely check whether your code can deal with `data3.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 `gcoord.c` and `gcoord.source` files into the new testing directory.

```
... this assume your files are in /localstorage/$USER/postgres
... if they are somewhere else copy from that location instead
$ cd /localstorage/$USER
$ cp postgresql-15.1/src/tutorial/gcoord.c testing/gcoo
$ cp postgresql-15.1/src/tutorial/gcoord.source testing/
```

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-15.1/src/tutorial/` directory. You can now run the tests.

```
$ cd /localstorage/$USER/testing
... edit your .c and .source files ...
$ make
$ ./run_test.py
... internal logs from postgresql are placed in pg.log ...
... external logs from the testing script are placed in test.l
... start by reading test.log after run_test.py has finished .
```

NOTES:

- `gcoord.source` should do the bare minimum needed to create the GeoCoord type as specified.
- `gcoord.source` should not try to create any tables, insert any data, or drop the `GeoCoord` type.
- Everything done in `gcoord.source` should be undo-able with "DROP TYPE GeoCoord CASCADE".
- The test will automatically convert your format into the default canonical form. If it raises the error 'Invalid Canonical Form' or 'Invalid Output' during test, you need to check whether the output format is valid.
- 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.
- **Plagiarism checking** will be conducted in marking.

Please report any issues ASAP on the Ed forum for this page.