COMP9311 22T3 Database Systems

Assignment 2

Testing Facilities

Last updated: **Sunday 23rd October 6:18pm**
Most recent changes are shown in red ... older changes are shown in brown.

[Assignment Spec] [Database] [Schema Summary] **[Testing]** [Samples]

This document describes the testing facilities available for Assignment 2.

After creating the database and loading the `mymyunsw.dump`, you have a basic MyMyUNSW database, with all of the data tables and their tuples.

To help test you views and functions against the Assignment 2 database, it is very useful to load one more file, whose contents are described in more detail below. Use the following command to do this:

```
$ psql  ass2  -f  /home/cs9311/web/22T3/ass/two/check.sql
```

This loads some additional tables containing expected results for calls to the views and functions. In order to prevent problems with people loading it multiple times and producing incorrect expected results tables, it drops each table before loading it. A side-effect of this is that the very first time you load the `check.sql` file, you may receive some messages like:

```
psql:check.sql:141: NOTICE:  type "testingresult" does not exist, skipping
...
psql:check.sql:410: NOTICE:  table "q1_expected" does not exist, skipping
DROP TABLE
...
psql:check.sql:421: NOTICE:  table "q2_expected" does not exist, skipping
DROP TABLE
...
```

As long as these are all `NOTICE`s and not `ERROR`s, everything is ok. You'll get these notices messages every time you load `check.sql` into a fresh copy of the MyMyUNSW database. You can safely ignore them, as long as they are accompanied by `CREATE TABLE` messages.

The `check.sql` file contains examples of expected output for the Assignment 2 exercises and functions to use this data to check *your* views/functions. The expected outputs are mostly available in the form of tables with names like `q1_expected`, `q3_expected`, `q8a_expected`, and so on. The expected results are used by the checking functions, but you can also use them "by hand" to get some idea of the expected results that your SQL queries should be producing, e.g.

```
$ psql ass2
... PostgreSQL welcome stuff ...
ass2=# select * from q1_expected;
... table showing expected results for "select * from Q1;" ...
ass2=# select * from q2_expected;
... table showing expected results for "select * from Q2;" ...
ass2=# ... etc. etc. etc.
```

The expected results tables were generated from queries like the following:

```
q1  ... select * from Q1 order by ...;
q2  ... select * from Q2 order by ...;
...
q8a ... select * from Q8(arg1, ...);
...
etc. etc. etc.
```

The checking scripts also use these queries, and you can use them yourself for your initial checks on the correctness of your queries.

The `check.sql` file also contains a collection of PLpgSQL functions that take your views/functions and compare their results against the expected results. They assume that your views are defined the same as in the `ass2.sql` file. The check functions are invoked as follows:

```
ass2=# select check_q1();
... gives a message on correctness of Q1 view ...
ass2=# select check_q2();
... gives a message on correctness of Q2 view ...
ass2=# select check_q8a();
... gives a message on correctness of Q8 function for one example query ...
```

The `check_qX()` functions test for the following conditions:

- your view/function (`QX`) does not yet exist
- your view returns too many tuples
- your view is missing some tuples
- your view has extra tuples and is missing others

Note that the most likely reason for the last condition being reported is that the format of individual attributes does not match the required format. You can check this by examining the results of your view and the expected results closely; looking for subtle table alignment/layout differences. Having “just one extra space” is not treated as a trivial error, because it's possible that the output of these views (in a real-life scenario) might be used as input to some other program, which could crash because the expected format was not followed exactly.

Note also that if you type the name of a `check_qX()` function incorrectly, PostgreSQL gives you the following error message:

```
ass2=# select * from check_q33();
ERROR:  function check_q33() does not exist
HINT:  No function matches the given name and argument types.
You may need to add explicit type casts.
```

The error is useful; the hint about adding explicit type casts is not useful.

The `check.sql` file has one checking function for each of the expected results tables (i.e. for each test case). There is also a function that will run all of the checking functions and produce a table of test/result pairs, which eventually should look like:

```
ass2=# select * from check_all();
 test | result
------+---------
 q1   | correct
 q2   | correct
...
 q8a  | correct
...
(N rows)
```

Note that it may take some time to produce results because some of the individual tests are slow.

While you're not required to completely understand the code in `check.sql` at this stage, it might be useful to take a look through it. You might pick up a few useful tricks.