CSci 4707: Practice of Database Systems
Lab 2
Spring 2017

Due: 03/21/2017 **18:30** on Moodle
There will be a 10% penalty off your grade for a 24 hours late submission.
This lab has to be done in a group of two.

A. Introduction

This lab needs to be done in a group of two. You are going to change the buffer replacement policy that is currently used in PostgreSQL, a major open source DMBS. In particular, the original replacement policy of PostgreSQL is a clocksweep algorithm with LRU (Least Recently Used). In this lab, you are asked to implement an MRU (Most Recently Used) replacement policy without using the clocksweep algorithm.

B. Most Recently Used

The basic idea of MRU can be seen in here. However, you will need to adapt the algorithm with the Database buffer management policy concept, e.g., pin count.

C. Setting-Up PostgreSQL (Same as Lab 1)

PostgreSQL is a large software. You will want at least 200 MB of space in your machine. Download version 9.6.1 of PostgreSQL and create the installation location by executing the following commands:

```
// Download the PostgreSQL and extract it.
$wget https://ftp.postgresql.org/pub/source/v9.6.1/postgresql-9.6.1.tar.gz
$tar xvzf postgresql-9.6.1.tar.gz

// Create the installation folder, called “install”.
$mkdir install
```

Then, we retrieve the full path of both source and installation folder by executing the following command:

```
// Retrieve the full path of the directory.
$pwd
```

```
# This will return a full path of your current directory. Throughout this
documentation, this path is called as $W$. So, Don’t forget to replace it with
your actual working directory.
```

Thus, the path of the PostgreSQL’s source code is located at “`$W$/postgresql-9.6.1`” and the path where we will install PostgreSQL is located at “`$W$/install`”.

Next, you will configure the installation of PostgreSQL and install PostgreSQL on the given path by executing the following commands:

```
// Go to the source directory.
$cd postgresql-9.6.1

// Run the configure command. Don’t forget to change $W$ to your actual
directory.
$./configure --prefix=$W$/install

// Run the makefile.
$make
# You should see the following output.
All of PostgreSQL successfully made. Ready to install.

// Run the installation.
$make install
# You should see the following output.
PostgreSQL installation complete.
```

After the installation has finished, you will need to initialize and create a database to use. In this example, we will initialize the database at “`$W$/install/data`” folder by executing the following command:

```
// Go to the installation folder.
$cd ../install
// Initialize database.
$bin/initdb -D data
```

Once the database has been initialized, we can start the PostgreSQL backend by executing the following command:

```
// Execute the backend.
$bin/postgres -D data
```

We can then use this terminal to get information on what is happening in our DBMS.

To run the client, open another terminal and use the following command to create a database to use: (In this example, we call the database as ProjectDB)

```
// Create the ProjectDB database.
$cd $W$/install
$bin/createdb -h localhost ProjectDB
```

We interact with our DBMS by using a program called psql. To start psql, use the following command:

```
// Run psql
$bin/psql -h localhost ProjectDB
```

The next time you want to start the database again, you only need to run two consoles where one is for the backend and one is for the psql console:

```
// Execute the backend.
$bin/postgres -D data

// Execute the psql console.
$bin/psql -h localhost ProjectDB
```

Several important commands for the lab are:

```
// You must exit psql by typing the following command and do not use Ctrl+C
ProjectDB=#\q

// To run an sql script located at $W$/script.sql
ProjectDB=#\i $W$/script.sql
```

D. Modification

Most (if not all) of your changes will be located in the following directory:
“`$W$/postgresql-9.6.1/src/backend/storage/buffer/`”.
To implement the MRU algorithm, you will want to maintain the timestamp of each buffer when they are called or used. For simplicity, you will maintain a global integer value which will be assigned to a buffer whenever a buffer page is requested, then, increment the integer.

For verifying the correctness of your implementation, you need to print out the timestamp of all candidate (not in use) buffers as: “Candidate buffers: <a comma separated list of timestamps>”

and also explicitly print out the timestamp of the buffer that is going to be used/replaced as: “Replaced buffer: <timestamp>”. Initially, all buffers will have zero timestamp.

E. Compiling and Testing Your Changes

To compile your changes, you must run the following three commands from the source directory, i.e., `$W$/postgresql-9.6.1`:

```
// Clean the executable.
$make clean
// Remake
$make
// Install the changes.
$make install
```

To test your changes, you will first need to change the memory size of PostgreSQL. Since the test case that is given to you is very small compared to the default memory size of PostgreSQL, it is a good idea to change the memory size in the configuration file of the PostgreSQL. The configuration is located in: “`$W$/install/data/postgresql.conf`”. You will change the shared_buffer size to shared_buffers = 128kB. To test your change, run the data_set/buffer_add.sql file and look at the replaced buffer’s timestamp compared to all other candidate buffers’ timestamp. In order to run buffer_add.sql, you need to make one modification to the file. Please make sure you change the path of data_set/values10k.dat file according to its location.

F. Help

The PostgreSQL online source code documentation will be an invaluable resource for this lab and the next. You can find it here.

Since reading and understanding an open-source framework is the main goal of the lab, the TA is only allowed to do the following:
Help you with the installation of PostgreSQL.
Help you in understanding the concept of buffer management policy (not specific to PostgreSQL query optimizer).
Help you in understanding what the assignment wants you to do.

Unfortunately, the TA is not allowed to help you in navigating the source code (including telling you the functionality of a function or variables) since one of the goal of the lab is to make sure that you understand the source code of the query optimizer in PostgreSQL. Once you understand the source code and know where to make the change, it is very easy to solve the lab.

G. Submission Guideline

One submission per group! You must submit a zip file named lab2.zip which contain:
(1) “changes”: A folder that contains all files that you have changed in order to fulfill the lab. Please only submit the file itself (don’t include the directory). For example: if you change a file in src/backend/foo.c, only put foo.c in the folder.
(2) “solution.txt”: Modify the solution.txt template from the zip file as per your changes.
Each violation to the submission guideline will result in a 10% penalty.

H. Grading Criteria

- 20 Points: Your changes must be able to compile without any errors.
- 50 Points: The MRU replacement policy including incorporating the printing instructions.
- 30 Points: The quality of your solution.txt