Project 1

Due 02/11/2025

This lab will be your first experience modifying PostgreSQL, a major opensource DBMS. Your assignment is to implement a different buffer management policy. The current version of PostgreSQL (16.4) uses a clock sweep algorithm to determine which buffer page should be replaced. In this lab, you will need to implement a FIFO (first in, first out) policy alongside the existing clock sweep policy.

This lab will not feature a large amount of coding. The entire assignment can be done in roughly 100 lines of code. Most of your time will be spent reviewing and understanding the source code, so make sure you get an early start. You will want to complete this lab in groups of two. At least one member of your group should be highly familiar with C/C++.

Installation

Download version 16.4 of PostgreSQL from https://www.postgresql.org/ftp/source/v16.4/ and install it on a Linux machine. Detailed installation instructions can be found at https://www.postgresql.org/docs/15/install-procedure.html.

We have also provided a Docker container setup guide, which is attached at the end of the document. We highly recommend following it, as it is already configured for your needs.

As an additional note, the standard `./configure` command should not be used during installation if you do not have permission to install anything to the root or home directory. You will want to specify another folder within your home directory to install PostgreSQL to, with the command

```
./configure --prefix=install_path
```

The prefix option requires an absolute path, not a relative one. One example would be

```
./configure --prefix=$HOME/PostgreSQL/installation
```

Setup

Once the program is fully installed, we need to initialize and create a database to use. To initialize databases to a certain folder, use the following command:

```
install_path/bin/initdb -D install_path/data
```

We can then start the PostgreSQL backend by using the following command:

```
install_path/bin/postgres -D install_path/data
```

We can then use this terminal to get information on what is happening in our DBMS. Open another terminal and use the following command to create a database to use:

Please note that we require you to set the `database_name` **in the format** `USC ID_Firstname` **for verification purposes.** One example is `8288xxxx88_Hanwen`.

```
install_path/bin/createdb -h localhost database_name
```

We interact with our DBMS system using a program called psql. To start psql, use this command:

```
install_path/bin/psql -h localhost database_name
```

You can now use this terminal to enter SQL commands and manage your database. After this, whenever you start up PostgreSQL, you should only need to run the postgres and psql commands, even after rebuilding and reinstalling.

Modification

All of PostgreSQL's code regarding buffer management is stored in `src/backend/storage/buffer`. The functions that control the buffer replacement policy are contained in `freelist.c`; most, if not all, of your modifications will be to this file. Figuring out how the source code works is part of your assignment, so the TA can only give limited help in helping you understand the code. If you like, you can use the debugging program `gdb` on the PostgreSQL backend to take a stepbystep look at how PostgreSQL replaces its buffers. It's a good idea to get an early start in deciphering how the buffer replacement function works. Leaving this lab for the last minute is an extremely bad idea.

Whenever you modify the source code of PostgreSQL, you can rebuild and reinstall it with the commands `make` and `make install`, or preferably `gmake` and `gmake install`. Make sure PostgreSQL is not running when you do this.

Do not completely eliminate the old buffer replacement code. The testing phase will need to compare the default clock algorithm and your FIFO algorithm side by side. The best practice is switching between methods using a debug flag and an if statement.

The PostgreSQL online source code documentation will be important for this lab and the next. You can find it at http://doxygen.postgresql.org.

Testing

As stated above, we will compare your FIFO strategy's performance and the default clock strategy. You will also be provided with test cases to use to see how performances compare.

Submission

Please ensure that your modifications are correct. We may hold an in-person session to verify that your code runs correctly. During this session, we will also check the modification of your `database_name,` as previously mentioned.

You should submit a `.zip` file or `.tar.gz` by 11:59 P.M. PST on the due date. Your submission should contain the following:

1. Any source code files that you modified.
2. A short README file.
3. A short report (one page or less) with the results of your test cases, comparing the two approaches and explaining why one might have outperformed the other.

Your README file should include:

1. The names and USC IDs of all group members.
2. A list of all the source code files you modified, as well as where they belong in the src hierarchy.
3. A brief description of how you modified the code.

Appendix 1: Docker Container Setup

STEP 1:

Use the Attached Dockerfile to Set up the Docker Image

In the terminal, go to the directory that contains the Dockerfile to build the image:

Please also check the Dockerfile and understand its details (username and password).

```
docker build -t 550project_env:latest .
```

Launch the instance: (Change the `--shm-size` based on your computer configuration, if you don't use `--shm-size=` , the default shared momory is 64MB, which may be bad for Postgre's performance, and may even smaller that its default value in the postgresql.conf file)

```
docker run --name 550project --privileged -it --shm-size=4g --cap-add=SYS_PTRACE
--cap-add=CAP_SYS_ADMIN 550project_env:latest
```

After that, you should be able to connect directly (with the extension: Remote Development) to the running container using VS Code, which we highly recommend. This will help you upload files and execute commands via the terminal.

STEP 2:

Set up the Postgres and Recover the Test Table and Data

Here, assume the username is `csci550user` , and all the operations happen under the user.

You can make different directory selections from ours.

```
cd /home/csci550user

wget https://ftp.postgresql.org/pub/source/v16.4/postgresql-16.4.tar.gz
tar xzvf postgresql-16.4.tar.gz
```

```
cd postgresql-16.4/
# To debug, use the following setting
CFLAGS=-O0 ./configure --prefix=/home/csci550user/postgresql-16.4 --enable-debug

sudo make -j
make install
echo 'export PATH=/home/csci550user/postgresql-16.4/bin:$PATH' >> ~/.bashrc
source ~/.bashrc
# open another terminal, to use the following instructions.
```

Feel free to rename the database's data directory(in the example, it's "databases") as you prefer.

```
pg_ctl -D /home/csci550user/databases initdb
pg_ctl -D /home/csci550user/databases start -l logfile
# to stop the service in the future you may use
pg_ctl -D /home/csci550user/databases stop
```

Optional: Grant privilege to gdb

If you want to use the `VS Code debugger` with `gdb`, you must grant `gdb` the necessary privileges.

```
sudo visudo

# add this command in the opened file, save and exit(the default launching
program may be vim, nano, etc. Search instructions to use them correctly):
ALL ALL=(ALL) NOPASSWD: /usr/bin/gdb
```

Modify the VScode `launch.json` to be this:

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "(gdb) Attach",
            "type": "cppdbg",
            "request": "attach",
            "program": "/home/csci550user/postgresql-16.4/bin/postgres",
            "MIMode": "gdb",
            "setupCommands": [
```

```
                {
                    "description": "Enable pretty printing for gdb",
                    "text": "-enable-pretty-printing",
                    "ignoreFailures": true
                },
                {

                    "description": "Set disassembly flavor to Intel",
                    "text": "-gdb-set disassembly-flavor intel",
                    "ignoreFailures": true
                }
            ]
        }
    ]
}
```

Please change the `database_name` as we mentioned before (`ID_Firstname`).

```
# connect to the postgres using psql:
psql -U csci550user -d database_name

# Execute the sql, you need to upload your file there before
\i /home/csci550user/buffer_add.sql
```
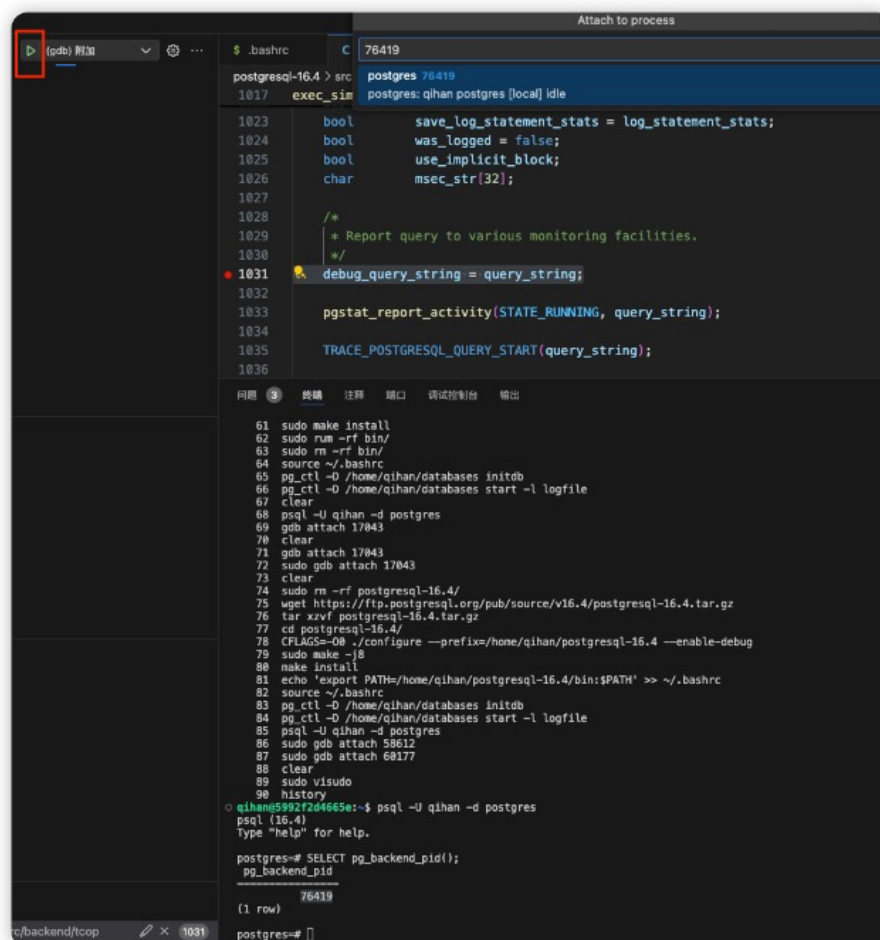
STEP 3: Try GDB to Debug.

WAY1: Debug With vs Code Debugger (Suggested)

Use `gdb` to set a breakpoint at the function `exec_simple_query`, specifically at line 1031 in the
file located at `/home/csci550user/postgresql-16.4/src/backend/tcop/postgres.c`. Manually
set the breakpoint at this position.

In the `psql` terminal, input `SELECT pg_backend_pid();` to retrieve the process ID. Assuming the process ID is `76419`, attach `gdb` to this process with the following command:

After attaching `gdb`, you can try running some queries in the `psql` terminal, such as:
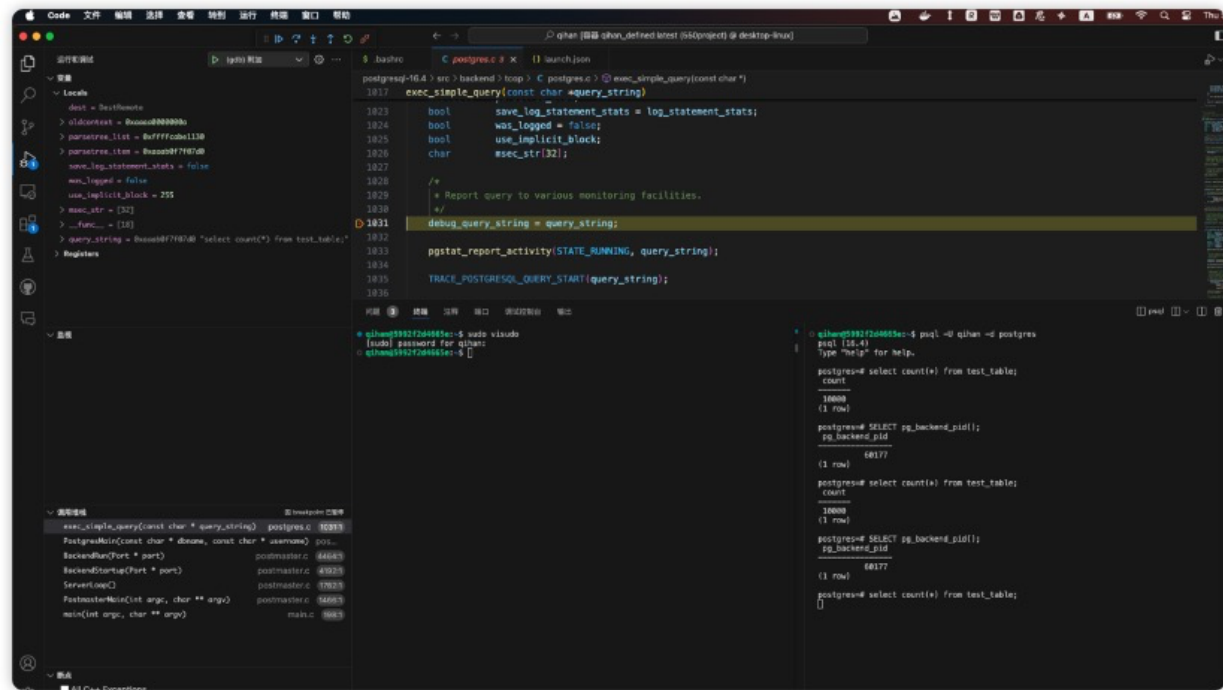
WAY2: Debug Without vs Code Debugger

```
sudo gdb attach $process_id
(gdb) b exec_simple_query
```

You can configure and investigate more details yourself.