Buffer Manager: Project 1 Assignment

CMU Computer Science 415 Spring 2003
Database Applications

January 27, 2003
Due: 8pm February 5, 2003

1 Administrivia

You should work in groups of three for this assignment. It is very important that each group member grasp the concepts covered in this handout, so please sit down together and go through things. This assignment should not take up your life for the next two weeks, but you will need to spend a fair amount of time reading through everything before you begin to hack. You are not allowed to discuss the assignment or your implementation with other groups, but you are allowed to share test cases. Please post any and all questions to the bboard as well, and we will try to provide responses as quickly as we can.

2 Overview of Project 1 - Buffer Manager

This project will add functionality to the real code of PostgreSQL. You need to delve into the actual code. Due to time constraints, we limit our investigation to one core module, the buffer manager. The actual coding for this project is minimal. The major parts of this project are:

- Understanding different memory management strategies
- Examining and understanding existing code
- Modifying the existing code to change replacement policy

Due to disk space constraints, we are providing a leaner PostgreSQL source tree. If you wish to download the additional features that you won't need for this project (such as contributed utilities, regression tests, manuals, tutorials, and other language support), feel free. Be aware that space is limited, so do not store files unrelated to the project in your account.

3 Tasks and Steps

1. Learn Different Strategies. (20%)
2. Compile and Test PostgreSQL.
3. Examine Code and Implement MRU. (40%)
4. Implement CLOCK. (40%)
5. Test.
6. Submit.

You should read this assignment *completely* before beginning to work.

4 Learn Different Strategies (20%)

The textbook describes LRU, MRU and CLOCK strategies in section 9.4.1. However, you may need to read the whole section of 9.4 to get a full image of buffer manager in DBMS.

Now we will do a small exercise to test your knowledge of the algorithms. For LRU, MRU, and CLOCK trace the buffer contents for the workload listed below. For your solutions, list each time that a memory access caused a page fault, and which (if any) page is thrown out of memory to make room. Record your answers in a file called ”strategies”. Start with LRU and list the time, a tab, and the page (if any) removed from memory. Then do MRU and CLOCK leaving a blank line in between. For example (just an example, no relation to the actual answer):

Time	Page Removed	
T1		←LRU
T2	W	
T5	Z	
		(a blank line here)
T2		←MRU
T5	Z	
		(a blank line here)
T4		←CLOCK
T5	W	

Notice the first page faults did not remove a page, so that column is blank.

Assume there are 4 pages your buffer manager must manage. All four pages are empty to start. For each access the page is pinned, and then immediately unpinned[1].

Time	Page Read
T1	A
T2	B
T3	C
T4	D
T5	A
T6	A

Time	Page Read
T7	B
T8	E
T9	F
T10	A
T11	G
T12	B

Time	Page Read
T13	A
T14	B
T15	C
T16	G
T17	G
T18	F

Time	Page Read
T19	B
T20	E
T21	A
T22	C
T23	D
T24	A

5 Compile and Test PostgreSQL

First, you need to make sure your profile file (`.bash_profile`) includes following lines (We added them for you, but if there is a mistake, please go in and correct it):

```
export PGDATA=$HOME/pgdata                    #Postgres data directory
export PGPORT=(port number)                   #Postgres server port assigned to you
export PG_BIN=$HOME/bin/pgsql                 #Postgres install directory
export PG_SRC=$HOME/src/postgresql-7.2.2      #Postgres source directory
```

We have provided a script that will set everything needed for this assignment up for you. To run it, first copy it over to your home directory with the command:

```
cp /usr0/dbclass/hw1.sh $HOME
```

And then run the script by typing:

```
./hw1.sh
```

[1] You can not make this same assumption when writing the actual code. A page may be pinned for any length of time

This should take between five and ten minutes to run. Don't panic. The script will first create two directories off of your home directory: src/ and bin/. Do not delete these. It will also unpack a Postgres version 7.2.2 base, as you will need that version to complete the assignment. The base version requires about 32MB to unpack and compile. The full version requires about 50MB. No code was changed between these versions and the actual version. The base version just has some directories (features) removed and the Makefiles have been adjusted properly.

Next, copy over the "hw1 directory into your "src" directory like so:

```
cp -r /usr0/dbclass/hw1 src/
```

The contents of this will be explained in the "Testing" section of this handout.

More information on using the `pg_ctl/postmaster/postgres` are available at:

http://www.postgresql.org/idocs/index.php?app-pg-ctl.html
http://www.postgresql.org/idocs/index.php?app-postmaster.html
http://www.postgresql.org/idocs/index.php?app-postgres.html

An on-line book about using Postgres is available at:

http://www.commandprompt.com/ppbook/

6 Examine Code and Implement MRU (40%)

The actual implementation of MRU is rather straightforward once you understand the existing code. The existing code is not extremely clear, but it is not unreadable. It may take you a few hours (or more) to understand it. Since understanding the existing code is a significant part of the assignment, the TAs and Professor will not assist you in understanding the existing code (beyond what we discuss here).

The actual buffer manager code is neatly separated from the rest of the codebase. Located in `src/backend/storage/buffer` and `src/include/storage`. For MRU, we are interested in modifying only two files.

freelist.c which manages which pages are not pinned in memory, and which pages are eligible for replacement.

buf_internals.h which contains the definition for each buffer description (called BufferDesc). Most of the fields in BufferDesc are managed by other parts of the code, but for efficiency some fields are used to manage the buffer space that is eligible for replacement (when it's on the existing LRU freelist).

Some initialization of the queue occurs in buf_table.c. However, you must do all your initialization in freelist.c.

You may modify these two files to change the implementation from the existing LRU queue to the MRU discipline. You may find that you do not need to edit both files or change many of the functions.

Unfortunately there is no partial credit for this section. Be sure it is correct. A bad buffer manager makes the rest of the system useless!

7 Implement CLOCK (40%)

LRU (Least Recently Used) is the most common policy used by operating system and database systems. However, direct implementation of LRU is pretty expensive. So some systems use an algorithm called "CLOCK" instead. While "CLOCK" approximates the behavior of the LRU algorithm, it is much faster.

The description of "CLOCK" algorithm is available in Section 9.4.1 of the textbook. We give a description here to assist your implementation.

To implement the "CLOCK" replacement policy, you need to maintain the following information. You are free to maintain additional information if you want. But the following two pieces of information are necessary.

1. A *referenced* bit associated with each page . Each time a page in the buffer pool is accessed, the *referenced* bit associated with the corresponding page is set to 1. In your implementation, you can simply turn it on if *pin_count* goes 0 when you unpin a page.
2. The "clock hand" (*current* in the textbook): between 0 and DEF_NBuffers-1. Each time a page needs to be found for replacement, the search begins from from the *current* page and advances to '*current = (current + 1) % DEF_NBuffers*' if not found (i.e. in a clockwise fashion).

Each page in the buffer pool is in one of three states:

- pinned: *pin_count>0*
- referenced: *pin_count=0* but *referenced=1*
- available: *pin_count=0* and *referenced=0*

To find a page for replacement, you start from the *current* page, repeat the following steps until you find a page or all pages are pinned. If:

- page[*current*] is pinned: advance *current* and try again.
- page[*current*] is referenced: set status to 'available', advance *current* and try again.
- page[*current*] is available: use this page and advace *current*.

As in MRU, we are only interested in modifying `freelist.c` and `buf_internals.h`. Although some initialization code may be better put in other files like `buf_init.c`, you must do all your initialization in `freelist.c`.

Again, you should figure out where and how to modify the files yourself. We will not assist you beyond what we discuss here.

8 Testing

There are many ways to test your implementations. One good way that we have provided you with lies in our testing scripts in hw1/test and hw1/QueryTest.

We provide some files in `src/hw1/QueryTest`, which include:

- `compile.sh`
- `prepare.sql` and `query.sql`
- `R` and `S`
- `readme`

Before you work with these files, you need to do is go into `prepare.sql` and put your home directory information in place of the question marks.

After modifying the code in parts six and seven, use `compile.sh` to re-compile and re-install PostgreSQL. You just type:

```
./compile.sh <your freelist.c> <your buf_internals.h>
```

To run a query on your newly compiled version of PostgreSQL, consult the readme. You can change '`-B <nbuffers>`' in line '`$PG_BIN/bin/pg_ctl start -l $1 -o "-o '-B 16 -te -fm`

`-fh'"'` of `readme` if you want to change number of buffers. The statistics are saved in "logfile" you assigned. You can change these queries or commands as you wish to test more thoroughly.

Another good way to test lies in `src/hw1/test`. The files include:

- `Makefile`
- `fullbuftest.c`
- `fullstubs.c`
- `readme`
- test-clock
- test-mru

View the readme file to get more information about how the testers work. We have provided a few sample test cases to get you going. Feel free to come up with your own, and share them on the bboard with the class. Please limit the newsgroup discussion to test cases ONLY. There will be no discussion of any code or implementation details. Test cases, answers, and verifications are welcome, however.

Since a large portion of this assignment is looking over code and figuring out what it is doing, we will not assist you in working the test script. We feel that looking through the provided files will really help you to understand the buffer replacement scheme better, as well as to give you a good chance of learning how to debug and test your own systems code. Since these scripts will not be submitted, you can edit them and play with them as you see fit.

9 Submit

You must submit exactly 8 files (even if they are incomplete or unchanged). They are

1. README - what works, what doesn't, how many slip days, etc.
2. strategies - answers to the first part, in the correct format!
3. freelist.c.mru - MRU implementation
4. buf_internals.h.mru - MRU implementation
5. freelist.c.clock - CLOCK implementation
6. buf_internals.h.clock - CLOCK implementation
7. PERFORMANCE - explained below
8. GROUP - explained below

Your PERFORMANCE file should talk a bit about performance analysis of the 3 different replacement policies. The items in QueryTest should help you through this part. You can use that query, or your own to measure speed differences among the policies. Talk a little bit about why different policies would produce different (or the same) results. Lastly, please provide the query execution times and buffer hit rates that you achieved for the provided query in "QueryTest". These can be found in the logfile after execution.

Gather all of these files into a directory called hw1_submit off of one of the group members home directories on the itaniums. Then, email Joe (josepht@andrew.cmu.edu) with all group members names and andrew ids, as well as which group member the submission will be under. He will send you a submission confirmation upon retrieving your files, so until you receive that, do not assume he has gotten it. If a day elapses without a response, send him another mail and CC the other TAs to make sure we get it.

When you submit, be sure not to mix the files up. Additional files will not be accepted. Each group only needs to submit once. If there are multiple submissions, the latest timestamped submission by any group member will be used for grading and the calculation of slip days. Remember that slip days will be deducted from all group members.

10 The GROUP File

For this assignment and next, you are required to submit a file with your homework that lists all 3 group members names and id's, as well as an honest percentage of work that that particular person completed of the assignment. These numbers must total 100. The file should be formatted like so:

Member 1 Name Member 1 ID %

Member 2 Name Member 2 ID %

Member 3 Name Member 3 ID %