HW9. BTree_part2

부산대학교 정보컴퓨터공학부
2020-55645
신세환
제출일: 2024-06-11

보고서에는 다음과 같은 내용을 포함하도록 한다. 양식은 자유롭게 수정하여 사용한다.

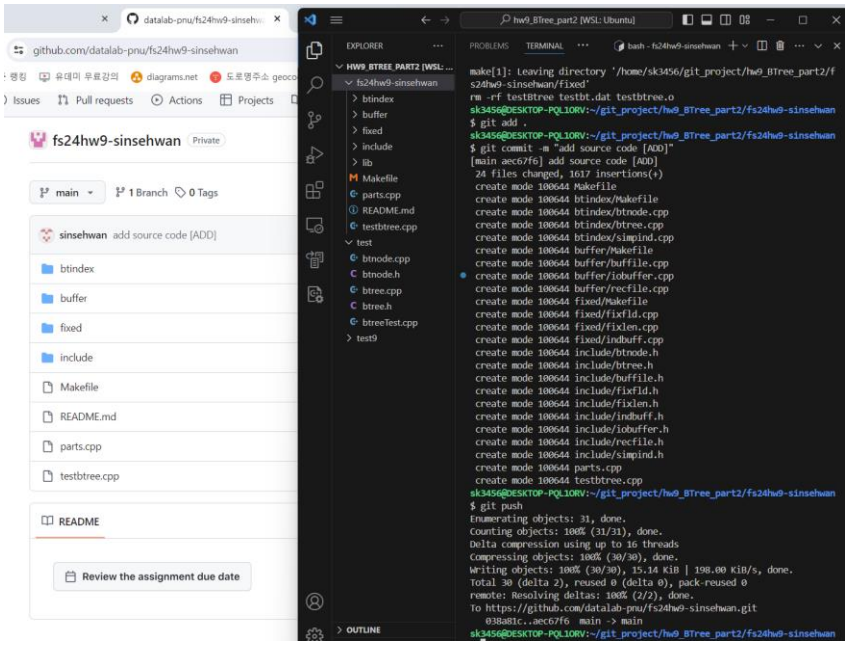
1. 문제정의

교재에서 제공된 코드만을 활용해서 만들어진 B+Tree로 기존 HW제출 때 사용했던 Student클래스와 CourseRegistration클래스를 활용해서 Student클래스의 학생 id를 키로 사용하는 B+Tree와 CourseRegistration클래스의 학생 id를 key로 사용하도록 만들어서 이를 테스트하고 동작시키는 코드를 만들어보자.

2. Github 화면

(1) git clone, commit, push

```
sk3456@DESKTOP-PQL1ORV:~/git_project/hw9_BTree_part2/fs24hw9-sinsehwan$ git add .
sk3456@DESKTOP-PQL1ORV:~/git_project/hw9_BTree_part2/fs24hw9-sinsehwan$ git commit -m "add source code [ADD]"
[main aec67f6] add source code [ADD]
 24 files changed, 1617 insertions(+)
 create mode 100644 Makefile
 create mode 100644 btindex/Makefile
 create mode 100644 btindex/btnode.cpp
 create mode 100644 btindex/btree.cpp
 create mode 100644 btindex/simpind.cpp
 create mode 100644 buffer/Makefile
 create mode 100644 buffer/buffile.cpp
 create mode 100644 buffer/iobuffer.cpp
 create mode 100644 buffer/recfile.cpp
 create mode 100644 fixed/Makefile
 create mode 100644 fixed/fixfld.cpp
 create mode 100644 fixed/fixlen.cpp
 create mode 100644 fixed/indbuff.cpp
 create mode 100644 include/btnode.h
 create mode 100644 include/btree.h
 create mode 100644 include/buffile.h
 create mode 100644 include/fixfld.h
 create mode 100644 include/fixlen.h
 create mode 100644 include/indbuff.h
 create mode 100644 include/iobuffer.h
 create mode 100644 include/recfile.h
 create mode 100644 include/simpind.h
 create mode 100644 parts.cpp
 create mode 100644 testbtree.cpp
sk3456@DESKTOP-PQL1ORV:~/git_project/hw9_BTree_part2/fs24hw9-sinsehwan$ git push
Enumerating objects: 31, done.
Counting objects: 100% (31/31), done.
Delta compression using up to 16 threads
Compressing objects: 100% (30/30), done.
Writing objects: 100% (30/30), 15.14 KiB | 198.00 KiB/s, done.
Total 30 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/datalab-pnu/fs24hw9-sinsehwan.git
   038a81c..aec67f6  main -> main
sk3456@DESKTOP-PQL1ORV:~/git_project/hw9_BTree_part2/fs24hw9-sinsehwan$
```



3. 논의 사항

근본적으로 교재에서 제공된 코드만을 사용해서는 구현하는 것이 불가능하였다.

```
BTree of height 2 is
Simple Index max keys 5 num keys 2
        Key[0] D RecAddr 224
        Key[1] T RecAddr 128
root-------------
Node at level 2 address 224 Simple Index max keys 5 num keys 3
        Key[0] A RecAddr 4
        Key[1] C RecAddr 0
        Key[2] D RecAddr 2
Node at level 2 address 128 Simple Index max keys 5 num keys 0
end of BTree
```

깃허브에 제공된 코드를 실행시켰을 때 insert를 5번째 실행했을 때, 즉 최초 Split을 수행하면서 B+Tree의 높이가 달라질 때 왼쪽 자식 노드는 정상적으로 관리되는데 오른쪽 자식 노드는 정상적으로 출력되지 않았다.

그래서 새로 만들어진 newNode 자체에 오류가 있는지 점검해보려고 다음과 같은 코드를 추가해서 출력해보았다.

```
cout << "print " << " newNode address : " << newNode->RecAddr << endl;//
(*newNode).Print(cout);//
cout << "print end₩n";//
```

이랬을 때

```
print  newNode address : 128
Simple Index max keys 5 num keys 2
        Key[0] S RecAddr 1
        Key[1] T RecAddr 3
print end
```

newNode 자체는 정상적으로 존재한다. 그러므로
BTree::insert함수의
Root.Keys[1]=newNode->LargestKey();
Root.RecAddrs[1]=newNode->RecAddr; 부분에서 Key와 RecAddrs모두 정상적으로 Root의 두 번째 요소로 들어가기 때문에 문제없이 출력이 되어야 한다.

그리고 while문에서 level<0일 때 root를 Split하고 높이를 1 증가시킨 후 insert를 수행해줘야 하는데 이 부분이 누락되어있었다.

종합적으로 문제가 생겨서 아예 insert함수의 while문을

```
while (result==-1) // if overflow and not root
{
        //remember the largest key
        largestKey=thisNode->LargestKey();
        // split the node
        newNode = NewNode();
        thisNode->Split(newNode);
        Store(thisNode); Store(newNode);
        level--; // go up to parent level
        if (level >= 0)
        {
                parentNode = Nodes[level];
        }
        else{
                parentNode = NewNode();
                Root = *parentNode;
        }
        // insert newNode into parent of thisNode
        cout<<"Before----"<<endl;
        parentNode->Print(cout);
        result = parentNode->UpdateKey(largestKey,thisNode->LargestKey());
        result = parentNode->Insert (newNode->LargestKey(),newNode->RecAddr);
        cout<<"After----"<<endl;
```

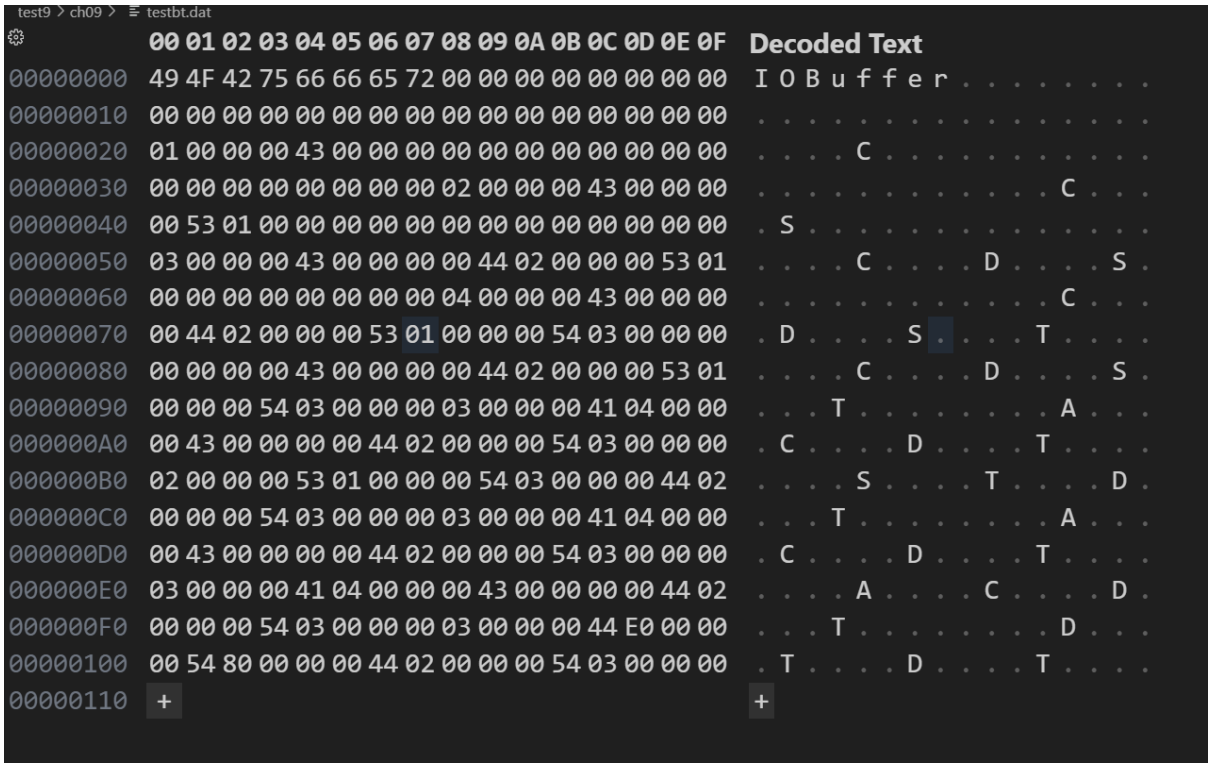
```
                parentNode->Print(cout);
                thisNode=parentNode;
        }
```

이런식으로 if문의 분기를 수정해서 Root를 새로 만들어야 하는 경우 parentNode를 NewNode()로 만들어서 다시 설정하도록 구현했다.

그러나 실행결과로

```
end of BTree
Inserting R
13
BTree of height 1 is
Simple Index max keys 5 num keys 4
        Key[0] G RecAddr 11
        Key[1] N RecAddr 10
        Key[2] R RecAddr 13
        Key[3] U RecAddr 12
root-------------
end of BTree
Inserting K
14
Before----
Simple Index max keys 5 num keys 0
After----
Simple Index max keys 5 num keys 1
        Key[0] U RecAddr 560
BTree of height 1 is
Simple Index max keys 5 num keys 0
root-------------
end of BTree
Inserting E
15
BTree of height 1 is
Simple Index max keys 5 num keys 1
        Key[0] E RecAddr 15
root-------------
end of BTree
Inserting H
16
BTree of height 1 is
Simple Index max keys 5 num keys 2
        Key[0] E RecAddr 15
        Key[1] H RecAddr 16
root-------------
end of BTree
Inserting O
```

이런식으로 Split이 발생해야 하는 시점마다 데이터가 사라지는 현상이 생겼다
이상하게 느껴서 다시 기존 코드대로 수정한 다음 실행파일 실행 시 생기는 testbt.dat파일을 열어봤는데



이런식으로 데이터 자체가 중복된 데이터들이 여러 개 존재하는 걸로 봐선 구조 자체가 잘못된 것 같다.

또한 Split이 발생하지 않는 범위까지 (i < 5)
테스트를 했을 때에는 testbt.dat파일 자체에 아무 내용도 생성되지 않았다.
따라서 교재에 제공된 코드는 사실상 B+Tree가 정상적으로 구현된 코드라고 볼 수 없고 이 코드만을 활용해서 과제를 구현하는 것은 불가능하다.