

DB 2차 과제 보고서

12171628 방솔찬

연락처 010-7538-3823 bsc981111@naver.com

1. 구현방법 및 설명

주어진 모든 function을 구현하였으며 그 구현을 위해서 b+tree code를 만들었습니다.

B+ tree의 block size 와 root id, depth를 저장하고 관리를 위해서 Fileheader class를 만들었습니다. 이 클래스는 파일의 맨 앞 12 바이트를 읽어서 정보를 불러오고 또 파일에 쓰는 역할을 합니다. 그리고 b+tree의 클래스를 만들어서 **bid가 주어지면 block을 불러오는 read_block 함수와 파일의 해당 위치에 쓰는 write_block 함수**, root node split이 일어날 경우 root node update 하는 set_rootnode 함수, leafnode와 nonleaf node에서 insert할 때 key가 sort된 상태로 들어가게 도와주는 insert_leafnode(), insert_nonleafnode() 함수, block에서 다음 레벨 노드 bid를 찾는 search_block 함수, 현재 채워지지 않는 다음 bid를 return하는 getNewBID() 함수가 있다.

insert에서는 b+tree에서의 insert연산을 구현하였습니다. key값을 통해서 노드가 삽입될 위치를 찾고 탐색할 때는 depth를 이용하여 leaf node인지 non leaf node인지 구분 했습니다. 삽입될 때는 insert_leafnode() 함수를 통해서 정렬된 상태로 들어가게 했습니다. Split이 필요할 때는 새로운 블록에 저장 후 파일로 다시 입력했습니다. 만약 split이 필요할 때 node max degree가 짝수일 경우 새로생기는 노드의 entry개수가 1개 더 많게 설정했습니다.

Search()에서는 해당 key값이 있는 leaf node를 순회하는 각각 block의 bid를 통해서 찾았습니다. Depth를 통해서 해당 key를 찾을 수 있는 leaf node를 찾고 그 내부에서 탐색을 하여서 entry를 return 하였습니다.

Range_search()에서는 startkey에서 endkey까지의 모든 entry를 return 합니다.

Startkey가 있는 entry를 위에 search() 함수와 동일한 방식으로 찾고 endkey까지의 값을 모두 스트림을 통해서 출력했습니다.

Printnode()에서는 root와 그자식들의 key값을 출력하는 연산을 하였습니다. 만약 depth가 0일 때는 루트노드의 key 값들만 출력하고 depth가 1이상인 경우, root node가 가르키는 bid를 vector bid_list에 저장한 후 두번째 level이 leafnode 이나 non leaf node에 따라서 key 값을 bid_list에서 하나하나 찾아서 출력했습니다.

2. 컴파일 언어 및 환경

언어는 c++ 이며 windows10 환경에서 visual studio 2019를 통해 작성되고 컴파일 되었습니다.

3. 출력 결과

1. Create()

```
C:\Users\solchan_bhang\Desktop\db 과제\2\new\sol\Release>sol.exe c btree.bin 36
```

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000 24 00 00 00 00 00 00 00 00 00 00 00 00 00
```

2. Insert()

```
C:\Users\solchan_bhang\Desktop\db 과제\2\new\sol\Release>sol.exe i btree.bin sample_insertion_input.txt
```

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000 24 00 00 00 09 00 00 00 02 00 00 00 08 00 00 00
00000010 2E 02 00 00 1B 00 00 00 8A 02 00 00 00 00 00 00
00000020 00 00 00 00 00 00 00 00 00 00 00 00 16 00 00 00
00000030 C8 01 00 00 AE 00 00 00 F2 01 00 00 0E 03 00 00
00000040 22 02 00 00 7F 01 00 00 00 00 00 00 00 00 00 00
00000050 12 00 00 00 01 00 00 00 22 00 00 00 16 00 00 00
00000060 55 00 00 00 05 00 00 00 8B 00 00 00 0A 00 00 00
00000070 00 00 00 00 00 00 00 00 1A 01 00 00 DF 00 00 00
00000080 31 01 00 00 5D 03 00 00 57 01 00 00 4F 02 00 00
00000090 00 00 00 00 00 00 00 00 13 00 00 00 55 00 00 00
000000A0 D0 03 00 00 7F 00 00 00 4B 01 00 00 00 00 00 00
000000B0 00 00 00 00 00 00 00 00 00 00 00 00 0A 00 00 00
000000C0 05 03 00 00 A5 02 00 00 48 03 00 00 39 02 00 00
000000D0 4D 03 00 00 8F 03 00 00 00 00 00 00 00 00 00 00
000000E0 07 00 00 00 7D 03 00 00 CF 02 00 00 92 03 00 00
000000F0 11 03 00 00 B6 03 00 00 C2 01 00 00 00 00 00 00
00000100 00 00 00 00 10 00 00 00 02 00 00 00 25 02 00 00
00000110 12 00 00 00 5E 02 00 00 0C 00 00 00 99 02 00 00
00000120 0F 00 00 00 D6 02 00 00 14 00 00 00 03 00 00 00
00000130 A6 00 00 00 0E 00 00 00 C8 01 00 00 08 00 00 00
00000140 05 03 00 00 11 00 00 00 00 00 00 00 00 00 00 00
00000150 8B 00 00 00 92 03 00 00 A4 00 00 00 E5 02 00 00
00000160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000170 0B 00 00 00 A6 00 00 00 6B 00 00 00 C3 00 00 00
00000180 13 03 00 00 DB 00 00 00 96 00 00 00 E2 00 00 00
00000190 8E 01 00 00 0D 00 00 00 5E 02 00 00 E1 00 00 00
000001A0 7B 02 00 00 11 00 00 00 7C 02 00 00 64 02 00 00
000001B0 00 00 00 00 00 00 00 00 0F 00 00 00 ED 00 00 00
```

```

000001C0 EC 01 00 00 F7 00 00 00 C8 01 00 00 13 01 00 00
000001D0 71 00 00 00 00 00 00 00 00 00 00 00 04 00 00 00
000001E0 0B 00 00 00 ED 00 00 00 0D 00 00 00 1A 01 00 00
000001F0 04 00 00 00 7A 01 00 00 13 00 00 00 A5 01 00 00
00000200 15 00 00 00 99 02 00 00 63 02 00 00 A1 02 00 00
00000210 6F 01 00 00 B6 02 00 00 E0 01 00 00 00 00 00 00
00000220 00 00 00 00 14 00 00 00 D1 03 00 00 9B 02 00 00
00000230 DE 03 00 00 6F 00 00 00 DF 03 00 00 A7 03 00 00
00000240 00 00 00 00 00 00 00 00 00 00 00 00 06 00 00 00
00000250 7D 03 00 00 07 00 00 00 D1 03 00 00 10 00 00 00
00000260 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000270 25 02 00 00 0C 01 00 00 27 02 00 00 C8 00 00 00
00000280 2B 02 00 00 6C 01 00 00 45 02 00 00 2A 02 00 00
00000290 0C 00 00 00 7A 01 00 00 26 03 00 00 9F 01 00 00
000002A0 47 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002B0 00 00 00 00 15 00 00 00 D6 02 00 00 F4 01 00 00
000002C0 DE 02 00 00 06 03 00 00 F2 02 00 00 45 03 00 00
000002D0 00 00 00 00 00 00 00 00 06 00 00 00 A5 01 00 00
000002E0 5B 03 00 00 AA 01 00 00 61 03 00 00 C2 01 00 00
000002F0 69 00 00 00 00 00 00 00 00 00 00 00 02 00 00 00
00000300 22 00 00 00 2D 03 00 00 2E 00 00 00 DB 01 00 00
00000310 4A 00 00 00 7A 03 00 00 00 00 00 00 00 00 00 00
00000320 05 00 00 00

```

3. Point search

```
C:\Users\solchan_bhang\Desktop\ddb 과제\2\new\sol\Release>sol.exe s btree.bin sample_search_input.txt search_output.txt
```

search_output - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```

74,890
282,223
893,719
164,741
549,268
694,480
8,558
226,398
34,813
456,174

```

4. Range search

```
C:\Users\solchan_bhang\Desktop\ddb 과제\2\new\sol\Release>sol.exe r btree.bin sample_range_search.txt range_output.txt
```

range_output - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

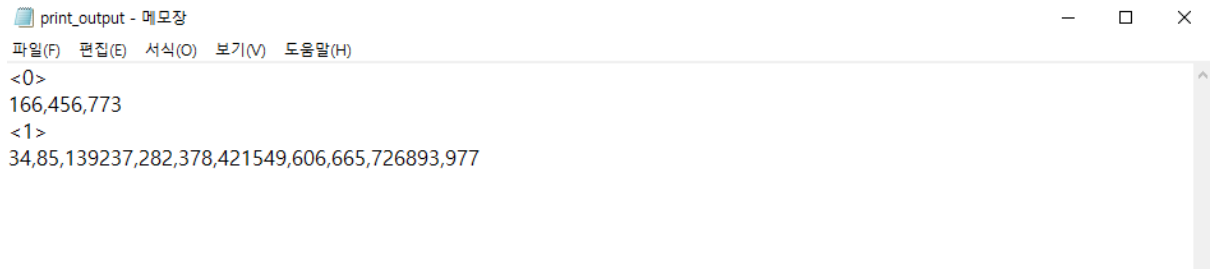
```

127,331 139,914 164,741 166,107 195,787 219,150 226,398 237,492 247,456 275,113 282,223
415,327 421,859 426,865 450,105 456,174 498,782
546,383 549,268 551,200 555,364 581,554
635,17 636,612 665,611 673,367 694,480
|

```

5. print()

```
C:\Users\solchan_bhang\Desktop\db 과제\2\new\sol\Release>sol.exe p btree.bin print_output.txt
```



```
print_output - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
<0>
166,456,773
<1>
34,85,139237,282,378,421549,606,665,726893,977
```

수업은 매주매주 알차고 짝차 있는 수업이었지만 이 과제는 지금 4-1학기를 끝내면서 들은 과제 중 제일 어려운 과제 였습니다. 이는 자료구조를 막 끝낸 3학년에겐 버거운 과제여서 대부분의 학생들이 포기하거나 완성하지 못할 것 같습니다. 다음 학년을 위해서 과제 난이도를 낮추는 것도 학생성취도에 좋은 영향이 될 것 같습니다.

한학기동안 좋은 수업 감사히 들었습니다.