Design Document

B+tree

제출일	2022. 09. 23	**전공**	컴퓨터소프트웨어학부
과목	데이터베이스시스템 및 응용	**학번**	2019041703
담당교수	김상욱 교수님	**이름**	김경민

0. 전역 변수 및 기초 구성

1) Node

m : 저장된 키의 개수

p: key 와 value 순서쌍의 pair 를 저장

r: nonleafnode 는 가장오른쪽 child 를 , leaf 노드는 오른쪽 leaf 노드를 저장

parentnum: load 할 때 indexfile 의 parentnum 을 저장

isconnected: load 할 때 parent 와 연결여부를 true false 로 저장

2) 전역변수

root = tree 의 최상단의 노드를 저장

degree = 한 노드에 저장 할 수 있는 최대 child 수를 저장

mDegree = 한 노드에 저장 할 수 있는 최대 pair 쌍의 수을 저장

nodeindex = save 할 때 현재 노드의 번호를 저장 하기 위해 사용

parentNode = findParentIndex 함수를 통해 특정노드의 parent node 를 저장

parentIndex = findParentIndex 함수를 통해 특정노드의 parent node 의 pair index 를 저장

leafIndex = load 할 때 leafnode 의 nowindex 를 순서대로 저장

flag = load 할 때 leaf 노드 인지 구분하기 위한 Boolean 변수 저장

1. Data save&&load

```
4
$ 0 1
6100,1 8764,1 10764,1
$ 1 2
886,1 987,1 1272,1 @
$ 1 3
6100,1 6488,1 @
$ 1 4
8764,1 9058,1 9292,1 @
$ 1 5
10764,1 12370,1 @
```

첫 줄에는 degree 값을 넣고

그 후 부모노드의 index 와 자신의 index 를 넣고

Key 와 value 값을 을 ,로 구분하여 넣는다.

만약 노드가 leafnode 라면 key,value 를 출력후 @를 붙여준다.

Dfs 를 이용해 leaf 노드가 나올때까지 이동한다

데이터를 읽고 tree arraylist 에 저장해준다

@를 통해 leaf 노드를 찾으면 leafindex 에 해당노드의 index 를 저장후 parentnum 을 index 로 가지는 node 에 연결해준다

Parentnum 을 따라 올라가다가 Isconnected 가 true 가 되면 연결을 멈춘다.

2. Data insert

1. 키를 넣어줄 leaf node 를 찾은 후 key 를 삽입한다.

2. 만약 해당 노드의 m 이 mDegree 보다 크다면 leafSplit 함수가 호출된다

3. leafSplit 이 호출되면 노드(leftnode)의 pair 개수는 mDegree+1/2 로 rightnode 의 m 의 값은 mDegree 의 값이 홀수라면 노드와 같게 짝수라면 +1 을 넣는다

4-1. node 의 split 된 pair 쌍을 rightnode 에 넣어주고 node 가 root 가 아닌경우 node 의 parent 를 findParentIndex 함수로 찾는다.

4-2. 만약 node(leftnode)가 root 라면 새로운 leftnode 를 만들어주고 leftnode 와 rightnode 에 각각 값을 넣어준다 그 후 root 를 clear 해준후 rightnode 의 첫번째 pari 값을 넣고 left 노드와 rightnode 와 root 를 연결해준다.

5. Parentnode 에 오른쪽 노드의 1 번째 값을 넣어주고 연결해준다. 만약 parentnode 의 m 이 mDegree 보다 크다면 nonleafSplit 함수가 호출된다.

6. Nonleafsplit 이 호출되면 node(leftnode)의 m 의 개수를 현재 노드에 저장되어있는 m/2 로 right 의 노드개수는 node.m 의 개수가 짝수면 leftnum-1 , 홀수면 leftnum(node.m/2)으로 정해준다. 왜냐하면 중간값은 parent 로 가야 하기 때문이다.

7-1. rightnode 에 값을 넣어주고 node 가 root 가 아니라면 node 의 r 을 중간 pair 의 child 로해준다.

Node 에 pair 쌍을 삭제해준후 findparentIndex 함수로 parent 를 찾고 parent 노드와 rightnode 를 연결시켜준다

Parentnode 의 m 이 mderee 보다 커지면 nonleafsplit 을 다시 call 해준다.

7-2 .만약 node 가 root 라면 leftnode 와 rightnode 를

생성해주고 root 와 연결시켜준다.

3. Data delete

1. delete 할 leafnode 를 찾고 key 를 삭제한다 삭제 후 노드의 개수가 mdegree/2 보다 같거나 크다면 삭제된 key 가 0 번째 인덱스라면 swapkey 함수를 통해 nonleaf 의 key 를 교체해주고 아니라면 return 해준다

2. 만약 노드의 개수가 mdegree 보다 작다면 underflow 가 난다. Leftsibling 과 rightsibling 을 찾고 rightsibling 이 mDegree/2+1 이상의 m 을가진다면 takeRight 함수를 호출하여

Key 를 가져오고 return 해준다

3. 아니라면 leftsibling 이 mDegree/2 +1 이상의 m 을 가진다면 takeLeft 함수를 호출하여 key 를 가져오고 return 해준다

4. 둘다 성립되지 않으면 mergeLeaf 함수를 호출한다.

5. Mergeleaf 가 호출되면 parent node 를 찾고 null 이 아닌동안에 rightsibling 이 null 이 아니라면 rightsibling 과 merge 를 진행한다 rightnode 에 값을 넣어주고 swapkey 를 진행하준다

6-1. 만약 parentnode 가 root 고 m 이 0 이라면 rightsibling 을 root 로 지정해준다.

6-1. 만약 parentnode 가 underflow 가 난다면 mergeNonleaf 를 호출한다

7. rightsibling 이 null 이고 leftsibling 이 null 이 아닌 경우에서도 같은 방식으로 진행한다

8. mergeNonleaf 함수가 호출되면 인자로 받은 node 의 parent 를 찾고 null 이 아닌동안에 rightsibling 이 null 이 아니라면 rightsibling 과 merge 를 진행한다. Mergeleaf 와 다른점이라면 parentnode 의 parentindex 의 key 와 value 값을 rightsibling 에 넣어준다는 것이다.

9. 같은 방식으로 rightsibling 이 null 이고 leftsibling 이 null 이 아닌경우 merge 를 진행하는데 parentnode 의 parentindex-1 의 key 와 value 값을 넣어준다 . 같은방식으로 인자로받은 node 의 parent 가 underflow 가 일어난다면 다시 nonLeafmerge 를 호출해준다.

4.Single search

1. Single search 가 호출되면 key 가 존재하는 leaf 노드를 찾기위해 root 에서부터 내려온다

2. key 보다 값이 큰 pair 의 child 로 내려가다가 leaf 노드에 도달하면 key 와 같은 key 를 가진 pair 의 value 를 출력한다

5. Ranged search

start 키를 siglesearch 와 같은 방식으로 찾은후 endkey 의 값보다 큰 값이 나올때까지 leaf 를 타고 쭉 이동하며 출력한다. m-1 의 인덱스에 가면 leaf.r 로 이동하 au 출력한다.

6. complie

terminal 에서 파일이 있는 경로까지 이동 후 아래 명령어 실행한다

1)creation

```
java Bptree.java -c index.dat 4
```

java Bptree.java -c index 파일

2)Insertion

```
java Bptree.java -i index.dat input_test.csv
```

java Bptree.java -i index 파일 input 파일

3)Deletion

```
java Bptree.java -d index.dat delete_test.csv
```

java Bptree.java -d index 파일 delete 파일

4)Single search

```
java Bptree.java -s index.dat 1500
```

java Bptree.java -s index 파일 searchKey

5)Ranged search

```
java Bptree.java -r index.dat 1000 5000
```

java Bptree.java -r index 파일 startKey endKey