实验 2 索引

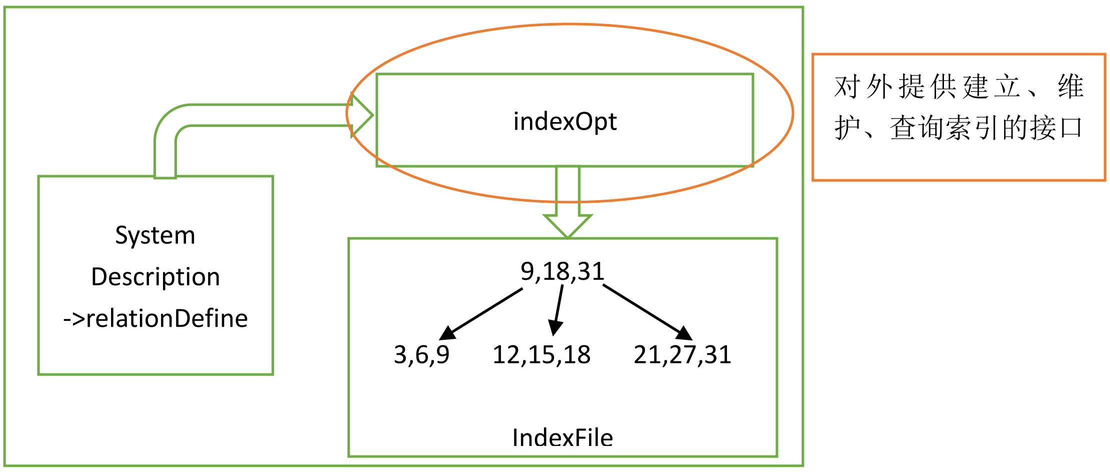
小组成员：吴天贞 陈师哲 周梦溪 蓝玮毓 张文慧

实验目的：

1. 索引的初始建立（根据表信息建立 B+树索引，作为外存索引保存）
2. 索引的维护（表格数据变更时，自动更新该表格上的索引）
3. 索引的查找（根据 key 值在索引中搜索得到实际位置）

实验设计

1. 整体框架


indexOpt
对外提供建立、维护、查询索引的接口
System Description ->relationDefine
9,18,31
3,6,9
12,15,18
21,27,31
IndexFile

2. 数据结构

- 系统状态信息：

```
struct SysDesc {
        long sizeOfFile;            //    数据库系统的容量
        long sizePerPage;           //    每一个页的大小
        long totalPage;             //    总共的页数
        long pageAvai;              //    当前有多少可用的页

        long bitMapAddr;        //    bitMap 在数据库系统中的起始地址
        long sizeBitMap;            //    bitMap 的大小，以字节为单位

        long dataAddr;              //    数据库系统中数据区的大小
        long segmentNum;            //    数据库系统中最多容纳段的数目
        long segmentAddr;           //    数据库系统中存储段的起始地址
        long segmentCur;          //    目前使用过的段的数量

        long curfid;              //    目前可以分配的 fid 号
        long curFileNum;          //    目前文件(表)的个数
        struct FileDesc fileDesc[MAX_FILE_NUM];   //每一个文件的描述
        struct relationDefine redef[MAX_FILE_NUM];        //数据字典
    };
```

- ■ 属性定义：

```
    struct attributeDefine
    {
        char attributeName[NAMELENGTH];     //属性名
        int type;                       //整型 1、字符型 2、日期型 3
        int length;                     //属性长度
        int recordDeviation;            //记录内偏移
    };
```

- ■ 关系定义：

```
    struct relationDefine
    {
        long fileID;                            //文件标识
        char relationName[NAMELENGTH]; //关系名
        char constructor[NAMELENGTH];    //建立者
        int attributeNum;                       //属性个数
        int recordLength;                       //记录长度
        int recordNum;                          //记录总数
        struct attributeDefine attribute[ATTRIBUTENUM];//属性定义表
    };
```

- B+树索引结构
  - ■ 树节点定义：

```
    typedef struct
    {
        int key;        //索引 key 值
        int pos;        //记录实际位置
    }Element;

    typedef struct
    {
        int type;       //叶子节点或普通节点
        int count;      //该节点内实际的 element 个数
        Element pair[MAX];    //节点内 element
        int parent;           //节点的父节点
    }Node;
```

3. 函数接口

1）B+树数据结构

函数	输入	功能
search	index, skey	在索引文件 index 中检索 key 值为 skey 的记录，返回 pos
insert	index, elem	在索引文件 index 中插入一个 element，element 结构中保存了 key 值与 pos
del	index, skey	在索引文件 index 中删除 key 值为 skey 的 element
display	Index	遍历索引文件树结构，输出节点信息
getRoot	index, node	将索引文件的根节点保存在 node 结构中
searchNode	index, node, key	在索引文件 index 中将 key 值所在的节点保存在 node 中
searchRecord	Node, key	返回在 B+树 node 节点中 key 值所在的 element 序号
insertRecord	index, node, elem	把一个 elem 插入 B+树的 node 节点中，写入 index 文件中
splitNode	index, node, elem, pos	插入 elem 后，将 node 节点分裂，更新 index 文件
enlargeKey	index, node	
delRecord	index, node, key	在索引文件 index 中删除 node 节点中 key 值对应的 element
transRecord	index, left, right, dir, pos	移动 node 中的 element 到相邻节点
mergeNode	index, left, right	合并两个节点
ensmallKey	index, node	
changeParent	index, node, child, parent	将 node 节点的父节点更改为 parent，更新 index 文件
displayNode	index, node	遍历树节点内的 element，输出节点信息
displayElement	element	输出 element 的信息

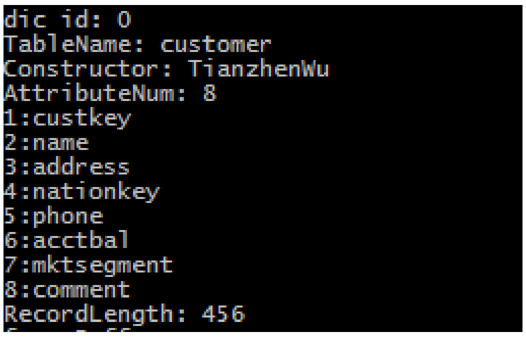
2）建立、维护索引

函数	输入	功能
createIndexOn	head, fid, column	根据文件（即表格）fid 的 column 属性建立索引。 根据 fid 查询数据字典，获取 column 对应属性序号，打开索引文件，遍历表中数据，读出记录属性值，将一个对应的存储 key 值与 pos 的 elem 插入 B+树索引结构，写入索引文件。
deleteIndex	head, fid, column	删除整个索引
insertInIndex	head, fid, key, position	表中插入一行数据时，在该表的所有已建立的索引中插入一个对应的 element
deleteInIndex	head, fid, key	表中删除一行数据时，在表的所有索引中删除

		一个对应的 elem

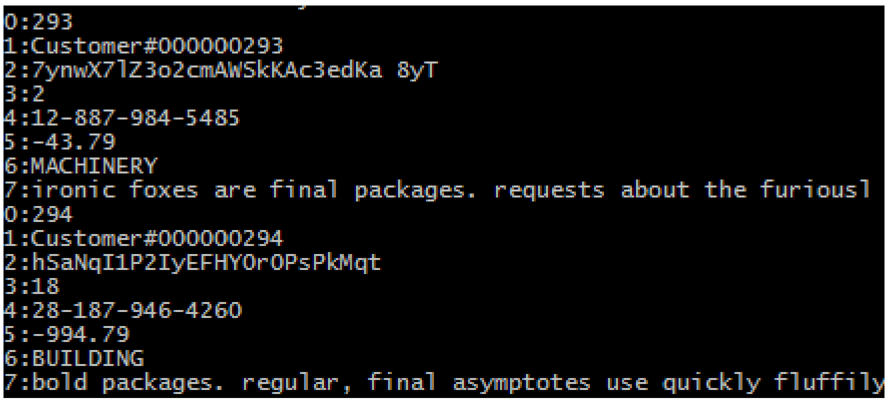
结果展示

1. 创建一个新的表文件，并导入数据。
   表格名为 customer，有 8 个属性，记录长度为 456byte



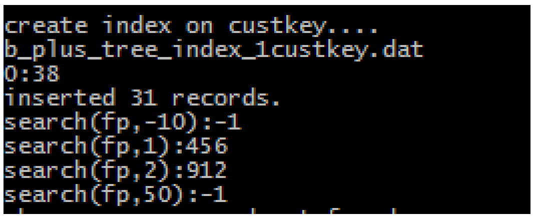
```
dic id: 0
TableName: customer
Constructor: TianzhenWu
AttributeNum: 8
1:custkey
2:name
3:address
4:nationkey
5:phone
6:acctbal
7:mktsegment
8:comment
RecordLength: 456
```

2. 部分表格数据示例
   0-7 为一条表格记录的 8 个属性对应的值，下图为其中两条示例数据。



```
0:293
1:Customer#000000293
2:7ynwX7lZ3o2cmAWSkKAc3edKa 8yT
3:2
4:12-887-984-5485
5:-43.79
6:MACHINERY
7:ironic foxes are final packages. requests about the furiousl
0:294
1:Customer#000000294
2:hSaNqI1P2IyEFHYOrOPsPkMqt
3:18
4:28-187-946-4260
5:-994.79
6:BUILDING
7:bold packages. regular, final asymptotes use quickly fluffily
```

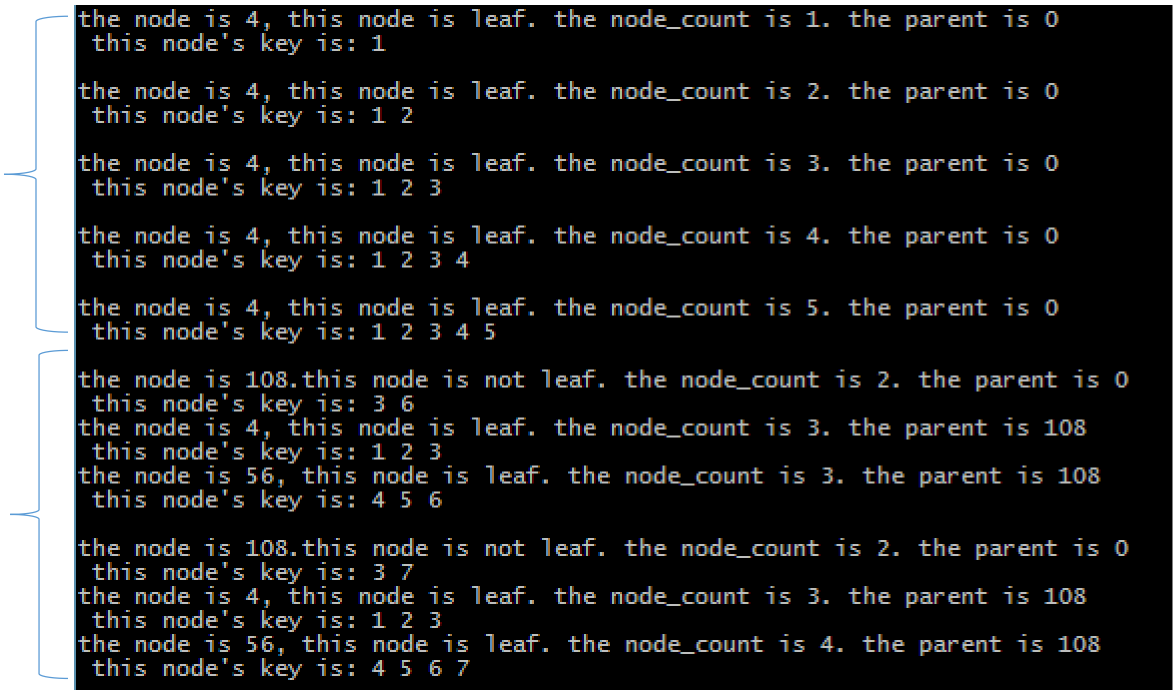
3. 在属性 custkey 上建立索引，并根据索引查找相应记录的位置
   外存索引文件名根据 fid 和属性名按照规则建立，在索引中插入了 30 条数据，属性值分别为 1-30，在索引文件中搜索 key 值为-10,0,1,2,50 的记录，输出返回的地址。其中 key 值为 1 和 2 时得到相应地址，其他 key 值不存在时返回-1。



```
create index on custkey....
b_plus_tree_index_1custkey.dat
0:38
inserted 31 records.
search(fp,-10):-1
search(fp,1):456
search(fp,2):912
search(fp,50):-1
```

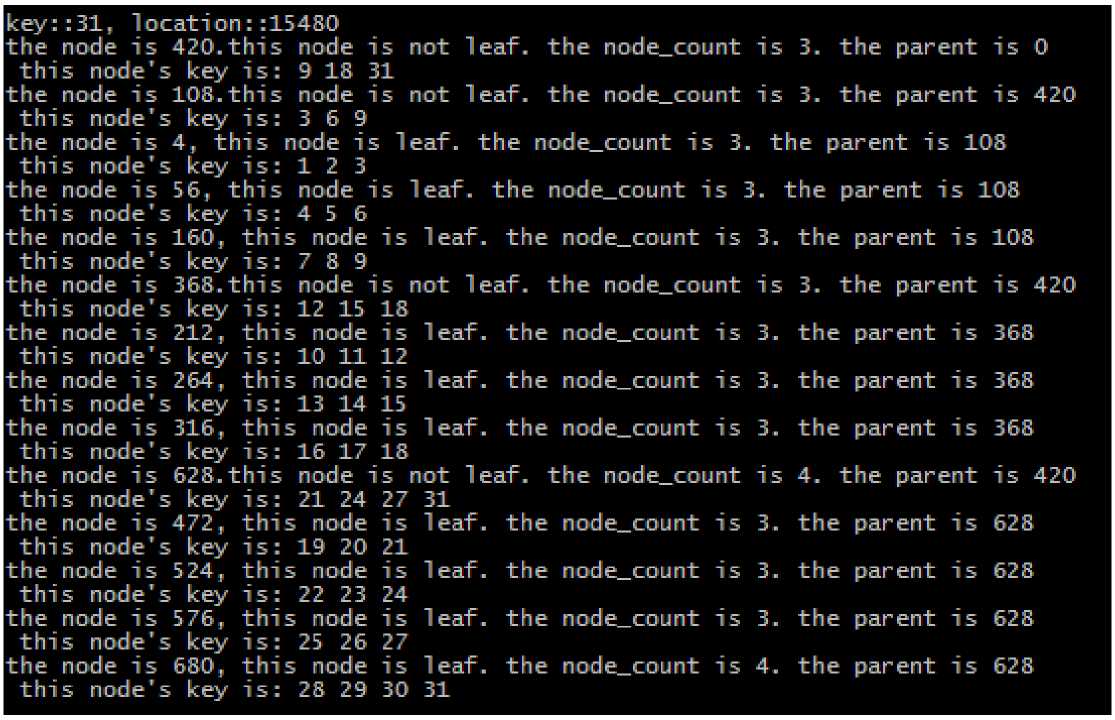
4. 输出索引
   依次插入 key 值为 1-7 的 element，输出每插入一条记录索引的变化。插入 1-5 时，节

点不分裂；插入 6 时，节点分裂为两个，左节点中存储 1-3 对应的 element，右节点中为 4-6 对应的 element。上层非叶子结点的值为 3 和 6，表示下层节点的中的 key 的最大值。



```
the node is 4, this node is leaf. the node_count is 1. the parent is 0
 this node's key is: 1

the node is 4, this node is leaf. the node_count is 2. the parent is 0
 this node's key is: 1 2

the node is 4, this node is leaf. the node_count is 3. the parent is 0
 this node's key is: 1 2 3

the node is 4, this node is leaf. the node_count is 4. the parent is 0
 this node's key is: 1 2 3 4

the node is 4, this node is leaf. the node_count is 5. the parent is 0
 this node's key is: 1 2 3 4 5

the node is 108.this node is not leaf. the node_count is 2. the parent is 0
 this node's key is: 3 6
the node is 4, this node is leaf. the node_count is 3. the parent is 108
 this node's key is: 1 2 3
the node is 56, this node is leaf. the node_count is 3. the parent is 108
 this node's key is: 4 5 6

the node is 108.this node is not leaf. the node_count is 2. the parent is 0
 this node's key is: 3 7
the node is 4, this node is leaf. the node_count is 3. the parent is 108
 this node's key is: 1 2 3
the node is 56, this node is leaf. the node_count is 4. the parent is 108
 this node's key is: 4 5 6 7
```

插入 31 个 element 后，B+树有三层，第一层的根节点值为 9,18,31，分别指向第二层的三个节点，第二层三个节点的值分别为 3,6,9 和 12,15,18 和 21,24,27,31。第三层为叶子节点，分别有 key 值为 1-31 的 element，存储对应表格记录的位置。



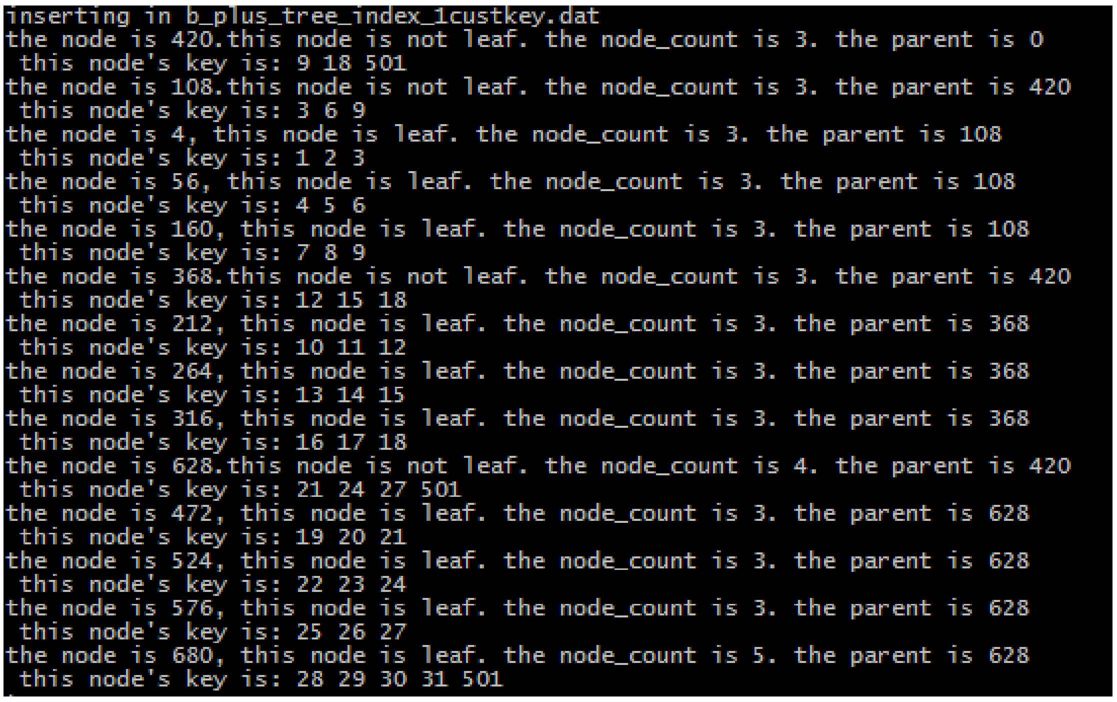
```
key::31, location::15480
the node is 420.this node is not leaf. the node_count is 3. the parent is 0
 this node's key is: 9 18 31
the node is 108.this node is not leaf. the node_count is 3. the parent is 420
 this node's key is: 3 6 9
the node is 4, this node is leaf. the node_count is 3. the parent is 108
 this node's key is: 1 2 3
the node is 56, this node is leaf. the node_count is 3. the parent is 108
 this node's key is: 4 5 6
the node is 160, this node is leaf. the node_count is 3. the parent is 108
 this node's key is: 7 8 9
the node is 368.this node is not leaf. the node_count is 3. the parent is 420
 this node's key is: 12 15 18
the node is 212, this node is leaf. the node_count is 3. the parent is 368
 this node's key is: 10 11 12
the node is 264, this node is leaf. the node_count is 3. the parent is 368
 this node's key is: 13 14 15
the node is 316, this node is leaf. the node_count is 3. the parent is 368
 this node's key is: 16 17 18
the node is 628.this node is not leaf. the node_count is 4. the parent is 420
 this node's key is: 21 24 27 31
the node is 472, this node is leaf. the node_count is 3. the parent is 628
 this node's key is: 19 20 21
the node is 524, this node is leaf. the node_count is 3. the parent is 628
 this node's key is: 22 23 24
the node is 576, this node is leaf. the node_count is 3. the parent is 628
 this node's key is: 25 26 27
the node is 680, this node is leaf. the node_count is 4. the parent is 628
 this node's key is: 28 29 30 31
```

5. 自动更新索引

向已建立好索引的表格中继续插入一条新的数据，自动更新在表格上的索引。

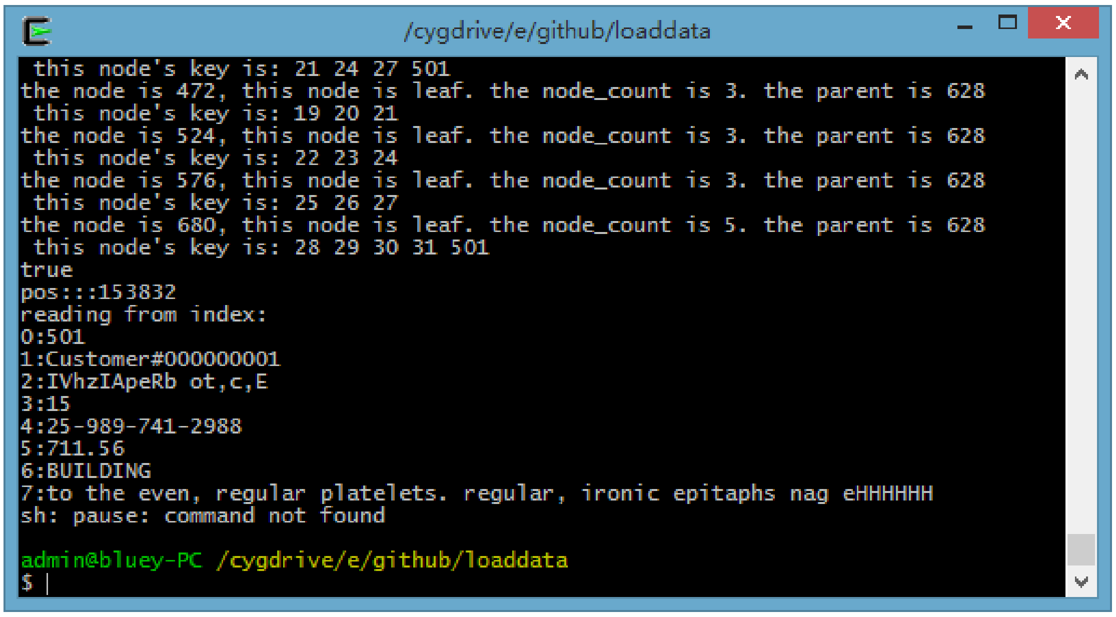
插 入 数 据 为 "**501**|Customer#000000001|IVhzIApeRb ot,c,E|15|25-989-741-

2988|711.56|BUILDING|to the even...HHH|"，key 值为 501，自动更新的索引中加入了 key 值为 501 的 element。



```
inserting in b_plus_tree_index_1custkey.dat
the node is 420.this node is not leaf. the node_count is 3. the parent is 0
 this node's key is: 9 18 501
the node is 108.this node is not leaf. the node_count is 3. the parent is 420
 this node's key is: 3 6 9
the node is 4, this node is leaf. the node_count is 3. the parent is 108
 this node's key is: 1 2 3
the node is 56, this node is leaf. the node_count is 3. the parent is 108
 this node's key is: 4 5 6
the node is 160, this node is leaf. the node_count is 3. the parent is 108
 this node's key is: 7 8 9
the node is 368.this node is not leaf. the node_count is 3. the parent is 420
 this node's key is: 12 15 18
the node is 212, this node is leaf. the node_count is 3. the parent is 368
 this node's key is: 10 11 12
the node is 264, this node is leaf. the node_count is 3. the parent is 368
 this node's key is: 13 14 15
the node is 316, this node is leaf. the node_count is 3. the parent is 368
 this node's key is: 16 17 18
the node is 628.this node is not leaf. the node_count is 4. the parent is 420
 this node's key is: 21 24 27 501
the node is 472, this node is leaf. the node_count is 3. the parent is 628
 this node's key is: 19 20 21
the node is 524, this node is leaf. the node_count is 3. the parent is 628
 this node's key is: 22 23 24
the node is 576, this node is leaf. the node_count is 3. the parent is 628
 this node's key is: 25 26 27
the node is 680, this node is leaf. the node_count is 5. the parent is 628
 this node's key is: 28 29 30 31 501
```

根据索引返回的 key 值为 501 的记录位置，从数据库中读取该记录，输出该记录。



```
 this node's key is: 21 24 27 501
the node is 472, this node is leaf. the node_count is 3. the parent is 628
 this node's key is: 19 20 21
the node is 524, this node is leaf. the node_count is 3. the parent is 628
 this node's key is: 22 23 24
the node is 576, this node is leaf. the node_count is 3. the parent is 628
 this node's key is: 25 26 27
the node is 680, this node is leaf. the node_count is 5. the parent is 628
 this node's key is: 28 29 30 31 501
true
pos:::153832
reading from index:
0:501
1:Customer#000000001
2:IVhzIApeRb ot,c,E
3:15
4:25-989-741-2988
5:711.56
6:BUILDING
7:to the even, regular platelets. regular, ironic epitaphs nag eHHHHHH
sh: pause: command not found

admin@bluey-PC /cygdrive/e/github/loaddata
$ |
```

代码清单

见 https://github.com/havenohavewifi/loaddata/tree/weiyu-version