Nanyang Technological University

CZ4031 Database System Principles Assignment 1

Group 36

Gabriel Low Zhi You - U2022715G

Sanskriti Verma - U2023954E

Japhet Tan Zhi Rong - U2022037F

Peng Weixing - U1921133E

Zou Zeren - U2022422H

Contribution Table

Name	Contribution
Gabriel Low Zhi You	Storage component, Experiment 1, Aid in Indexing component, report documentation, docker implementation
Sanskriti Verma	Indexing component, Experiment 2, Aid in Storage component, report documentation, docker implementation
Japhet Tan Zhi Rong	Storage component, Experiment 3, Aid in Indexing component, report documentation, docker implementation
Peng Weixing	Storage component, Experiment 4, Aid in Indexing component, report documentation, docker implementation
Zou Zeren	Indexing component, Experiment 5, Aid in Storage component, report documentation, docker implementation

Introduction

Description

The aim of our project is to design and implement the storage and indexing component of a database management system. In order to perform the same, we have made use of Golang.

Project Overview

Our implementation is made up of the following packages, each with its own set of responsibilities:

- File System Structure: Serializes records into byte arrays and manages serialized record storage.
- B+Tree: Implementation of the B+ Tree data structure, including search, insertion, and deletion operations.
- Examples: Executes the assigned test experiments and outputs the necessary metrics.

Instructions

Please email WeiXing at peng0099@e.ntu.edu.sg if you face difficulties running the program.

Download the supplied zip file. How to run:

<u>Method 1:</u>
Run the Windows executables

- experiments_200b.exe for block size 200b
- experiments_500b.exe for block size 500b

<u>Method 2:</u>
Via Docker
Docker build

1. Download the folder to local
2. Open terminal in local root
    a. **cd<your project root>**
    b. Eg. cd /Users/zeren/GolandProjects/CZ4031-Project
3. Run command to build image
    a. **docker build -t gobptree .**
4. Run command to run go application
    a. **docker run gobptree**

Method 3:

Manually constructing the project

1. Install Go according to the instructions at https://golang.org/doc/install#install, depending on your operating system.

2. Open terminal in local root
    a. **cd<your project root>**
    b. Eg. cd /Users/zeren/GolandProjects/CZ4031-Project
3. From the root directory, run command
    a. Go run **./exp.go**

Storage Component

File System Structure

The following struct depicts the internal structure of a file system instance:

```
type VirtualDisk struct {
    Capacity    int // Capacity in bytes
    BlockSize   int // Block size in bytes
    BlockHeight int // Number of blocks preceding in the disk
    Blocks      []Block
    LuTable     map[*byte]RecordLocation // Look-up table - Key: Address of record, Value: Block Index
}

type Block struct {
    NumRecord uint16 // 1 byte
    Content   []byte
}

type RecordLocation struct {
    BlockIndex int
    Index      int
}
```

{Figure 1: Structure of the File System}

The file system structure consists of 3 components, namely the VirtualDisk, the Block and the Record location.

The Virtual Disk instance consists of the following members:

- Capacity: the capacity of the Virtual Disk in bytes, in our experiment, this is set to 100 MB.
- BlockSize: the size of one block in bytes

- BlockHeight: number of used blocks in the disk
- Blocks: array of blocks storing the blocks present in the Virtual Disk
- LuTable: Look-up table to map key: address of record. Able to locate the block index and internal index of a record by their memory address

The Block instance consists of the following members:
- NumRecord: number of records currently in the block
- Content: byte array storing the raw contents of the records in the block

The RecordLocation instance consists of the following members:
- BlockIndex: index of the block in which record is stored
- Index: index within the block in which record is stored

Record Serialization

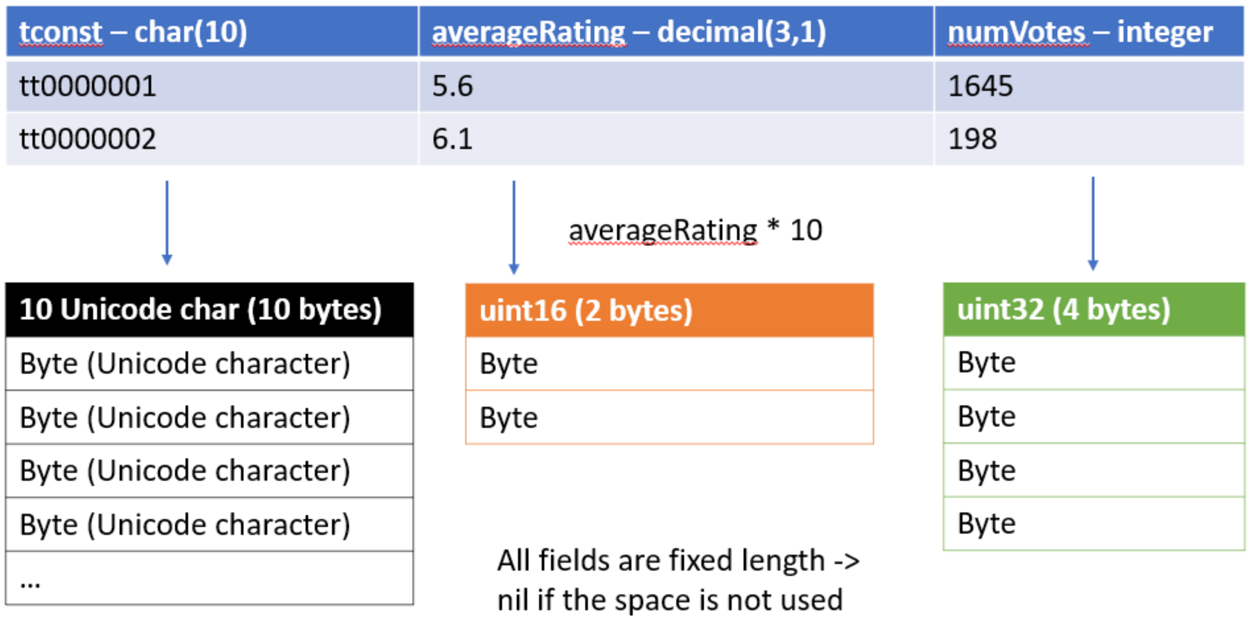
There are two field packing strategies that we can use, namely fixed format with variable length (FFVL) and fixed format with fixed length (FFFL). Since this project does not consider any other data, apart from the one provided, our group chose to go with the fixed format with fixed length strategy. The rationale for choosing this strategy are as follows:

1. Simpler implementation
2. The benefit of efficient storage by using FFVL is not significant. For instance, tconst's length is between 9 and 10, by implementing FFVL we could save 1 byte of space for some records, while incurring 1 additional byte (uint8) of space by storing the length information. No space is saved as a result.

```
Loading records from file....
Min tconst length: 9
Max tconst length: 10
Min avgRating: 1
Max avgRating: 10
Min numVotes: 5
Max numVotes: 2279223
```

(Figure 2: Simple data analysis)

To maintain uniformity in data size for each record, we serialize each record and store it as a byte array. Each record attribute is kept at a defined offset inside the array, making future retrieval easier and faster. Furthermore, additional memory is not required to specify the size of each field.



(Figure 3: Storage of fields in a record)

Field storage strategy is as follows:

- **Tconst**: Tconst is stored as 10 Unicode characters taking 1 byte each for a total of 10 bytes.
- **AverageRating**: In order to store AverageRating efficiently, the rating is first multiplied by 10 and stored as uint16 taking 2 bytes. The rating can be converted back by dividing 10.
- **NumVotes** is stored as uint32 taking 4 bytes, since the max is 2279223.

```
package fs

import ...

/.../
const (
    TconstSize     = 10
    AvgratingSize  = 2
    NumvotesSize   = 4
    RecordSize     = TconstSize + AvgratingSize + NumvotesSize
)
```

{Figure 4: Maximum length of each field}

The maximum string lengths for each field are shown in Figure 4. When a field is added to a record, its maximum length is used to calculate the offset.

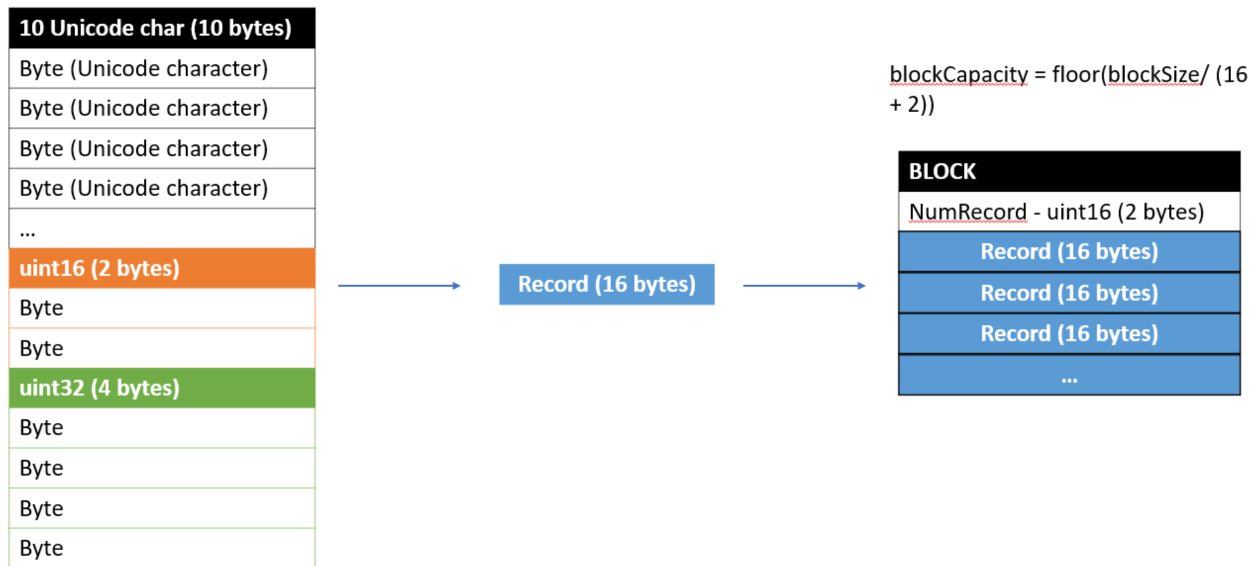
```go
type Record struct {
    Tconst        string
    AverageRating float32
    NumVotes      uint32
}

// RecordToBytes pack record into bytes
func RecordToBytes(record *Record) []byte {
    var bin []byte

    // Pack tconst
    tconstB := make([]byte, TconstSize)
    copy(tconstB, record.Tconst)
    bin = append(bin, tconstB...)

    // Pack averageRating
    avgRatingB := make([]byte, AvgratingSize)
    avgRating := uint16(record.AverageRating * 10) // Avg rating is stored as int -> /10 to convert back
    binary.BigEndian.PutUint16(avgRatingB, avgRating)
    bin = append(bin, avgRatingB...)

    // Pack numVotes
    numVotesB := make([]byte, NumvotesSize)
    binary.BigEndian.PutUint32(numVotesB, record.NumVotes)
    bin = append(bin, numVotesB...)

    return bin
}
```

{Figure 5: Serialization of record}

In this figure, an object consisting of NumVotes, AverageRating and Tconst are converted into byte arrays and further concatenated and returned.

Block Storage

A few records are then put in a block together. The records in the block are not kept in any particular order. This is done for several reasons, including:
1. Increased speed of record insertion and deletion
2. The nonclustered index's ease of implementation



(Figure 6: Record packing in a block)

In figure 6, the first 2 bytes in a block will store NumRecord as uint16 which holds the number of records in the block. The following bytes will hold the records taking 16 bytes each. In figure 7, blocks would be filled with new records. If no capacity is available in current blocks, a new block will be constructed.

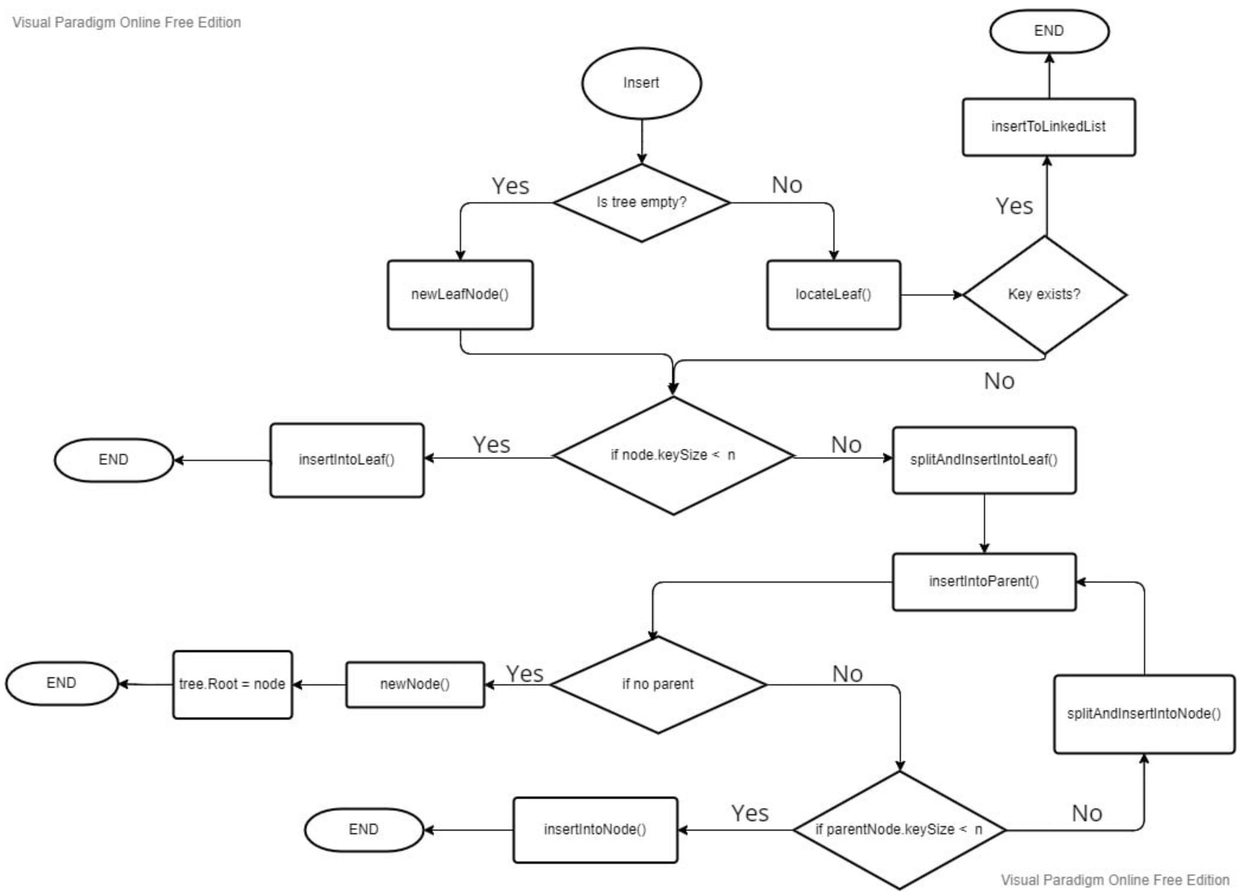
In order to simulate block behavior in actual file systems, the *VirtualDisk* is designed to retrieve records by blocks. For single record queries (i.e retrieve 1 record from the disk), the program will use the Look-up table (*LuTable*) to get the *RecordLocation* by record's address. The raw content of the record will be assessed immediately via Blocks[blockIndex][Index]. The record is then converted from raw bytes to *Record* object. On the other hand, for multi-record queries, the process is similar but the system will cache unique block indexes in an array first before assessing the content of the block, illustrated in figure 6.

```go
var accessedDataBlockIndexes []int

var totalAverageRating float32
for _, addr := range records {
    r := fs.AddrToRecord(vd, addr)
    totalAverageRating += r.AverageRating

    loc := vd.LuTable[addr]
    exists := false
    for _, a := range accessedDataBlockIndexes {
        if a == loc.BlockIndex {
            exists = true
        }
    }
    if !exists {
        accessedDataBlockIndexes = append(accessedDataBlockIndexes, loc.BlockIndex)
    }
}
```

{Figure 7: Block access mechanism }

B+ Tree Component

Overview



(Figure 8: Control flow logic for insertion)

Node Structure

B+ tree node is shown as follows:

```
type Node struct {
    /.../
    IsLeaf   bool
    Key      []uint32  //uint32 - 4 bytes
    Children []*Node   //Children[i] points to node with key < Key[i], Ptr[i+1] for key >= Key[i]
    DataPtr  []*Record //DataPtr[i] points to the data node with key = Key[i]
    Next     *Node     //For leaf node only, the next leaf node if any
    Parent   *Node     //The parent node
}
```

{Figure 9: Structure of a B+ tree node}

Each node consists of these members:

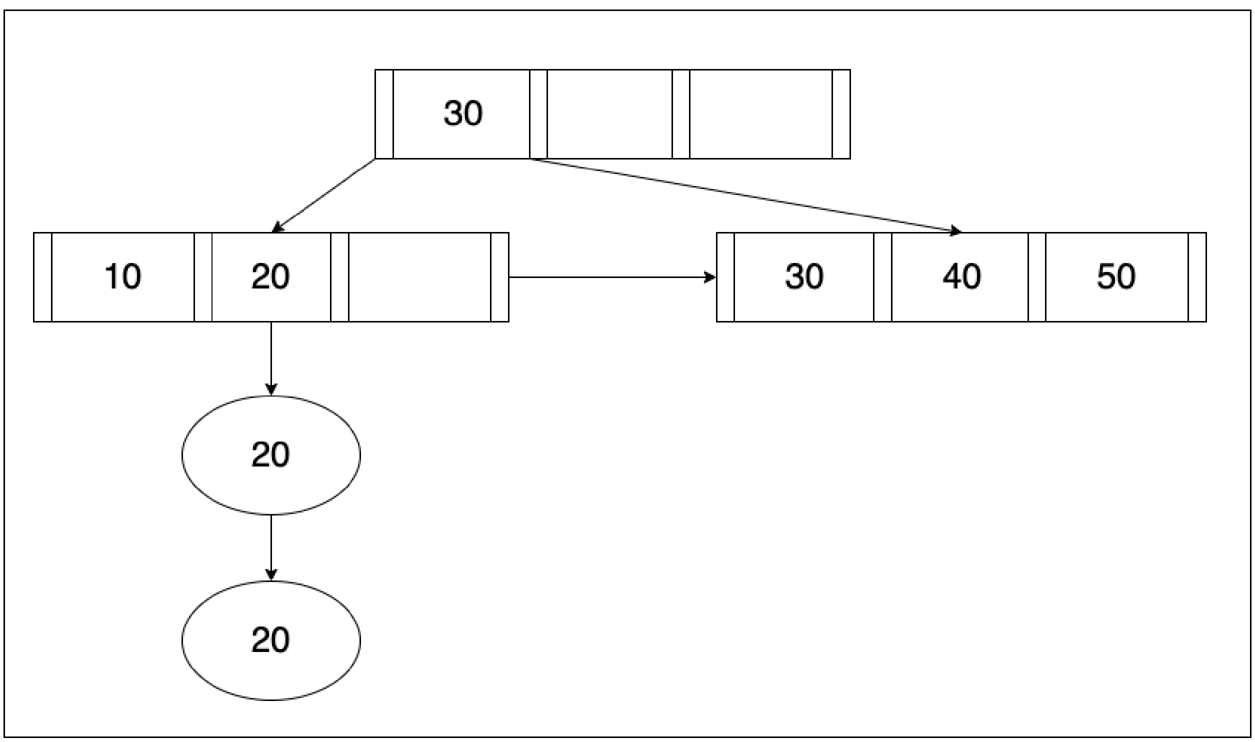
- IsLeaf: a boolean value to check if a particular node is a leaf node
- Key: slice of keys used for indexing and traversing the B+ tree
- Children: Children[i+1] points to node with key = Key[i]
- DataPtr: DataPtr[i] points to the data node with key = Key[i]
- Next: a pointer from one leaf node to the next leaf node, if any
- Parent: a pointer to the parent node

Duplicate Records Linked List

```go
type Record struct {
    Addr *byte
    Next *Record
}
```

(Figure 10: Sample Structure of each record)

Figure 10 displays how each record would hold a serialized byte array consisting of the data for each record, in addition to a pointer for the next record. This allows storing of the records in a linked list



(Figure 11: Linked List for B+ Trees)

To cope with the presence of multiple duplicate keys inside the record collection, we chose to add records with duplicate keys in a linked list manner. The correct insertion location in the leaf node is found when an insert operation is done.

If no other records with the same key exist, a new leaf node entry referencing the record is created. If a record with the same key already exists in the B+ Tree, the newly inserted record is added to the end of the existing entries.

We add numerous records in a linked list form from the first record. The fact that all nodes in a linked list will have the same key is one advantage of this technique. A key search would get the entire linked list.

Each record in a linked list would include a serialized byte array holding the contents for that record as well as a reference to the next record.

Number of Keys

	Key: uint32	Pointers: (To data or leaf)	Parent: ptr to parent	IsLeaf: bool
Size (bytes)	4	8	8	1

(Figure 12: Sizes for components in a node)

Set x as the number of keys
(Block size) = ((Ptrs) + (Keys)) * x + (Last ptr) + (Parent) + (IsLeaf)

For block size 200B:
200 = (8+4) * x + 8 + 8 + 1
Floor(x) = **15**

For block size 500B:
500 = (8+4) * x + 8 + 8 + 1
Floor(x) = **40**

Experiments

The following experiments are done for block size 200B and block size 500B

Experiment 1

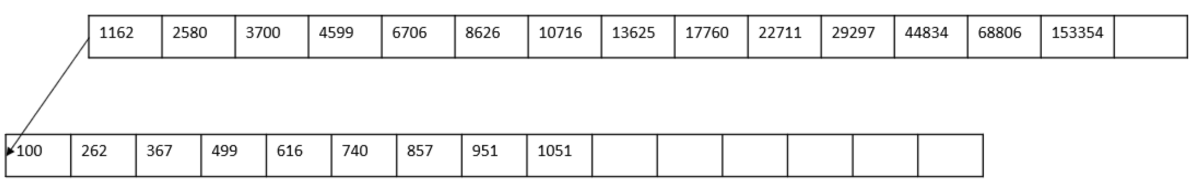
Storing the data on the disk

Block Size (B)	Number of Blocks	Size of Database (MB)
200	97302	19.46
500	39642	19.82

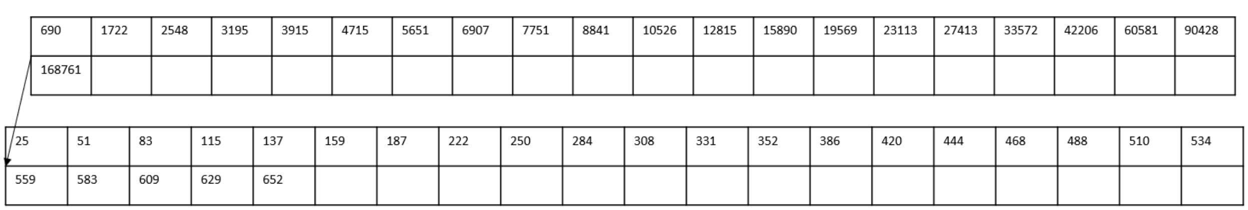
Experiment 2

Building a B+ tree on the attribute “numVotes” by inserting the records sequentially

Block Size (B)	n	Number of nodes	Height of B+ tree	Root node and 1st child node content
200	15	1882	4	Refer to figure 13
500	40	670	3	Refer to figure 14



(Figure 13: Content of root node and 1st child node for block size 200B)

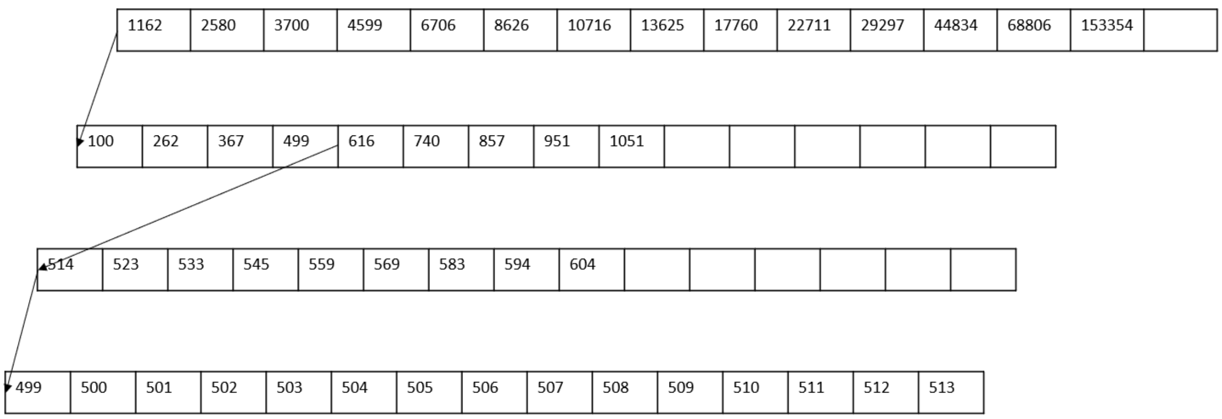


(Figure 14: Content of root node and 1st child node for block size 500B)

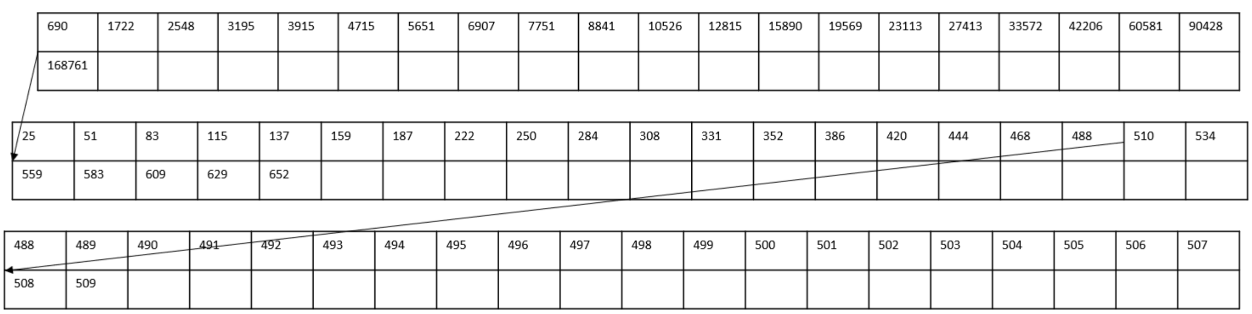
Experiment 3

Retrieving movies with attribute “numVotes” equal to 500

Block Size	No. of index node accessed	Content of index node accessed	No. of data blocks accessed	Content of data blocks accessed	Average of “averageRating’s” returned
200B	4	Refer to figure 15	110	Refer to figure 17	6.7318187
500B	3	Refer to figure 16	109	Refer to figure 18	6.7318187



(Figure 15: Content of index nodes accessed for block size 200B)



(Figure 16: Content of index nodes accessed for block size 500B)

Block #326:	Block #819:	Block #1075:	Block #2070:	Block #2471:
{tt0013624 6.5 21}	{tt0024549 6 361}	{tt0028276 5.9 30}	{tt0041946 5.5 39}	{tt0047356 7 10}
{tt0013626 6.7 2031}	{tt0024550 6.4 24}	{tt0028277 7.7 500}	{tt0041947 6.1 517}	{tt0047357 6.4 27}
{tt0013627 5.2 10}	{tt0024551 2.9 9}	{tt0028278 7.6 5}	{tt0041948 6.6 902}	{tt0047358 5.9 224}

{tt0013629 6.7 25}	{tt0024553 5.9 137}	{tt0028279 6.5 131}	{tt0041949 6.3 355}	{tt0047359 6.7 20}
{tt0013631 6.6 12}	{tt0024554 5.3 822}	{tt0028280 5.7 35}	{tt0041951 7.4 53}	{tt0047360 4.6 12}
{tt0013658 6.9 31}	{tt0024555 5.5 195}	{tt0028281 6.4 51}	{tt0041952 7.6 690}	{tt0047361 7.3 500}
{tt0013662 6.9 418}	{tt0024558 6.4 11}	{tt0028282 6.5 278}	{tt0041953 6.9 469}	{tt0047362 5.8 5}
{tt0013668 6.7 22}	{tt0024559 6.1 140}	{tt0028283 5.2 134}	{tt0041954 7.3 2435}	{tt0047363 6.7 30}
{tt0013672 6.7 25}	{tt0024560 6.9 397}	{tt0028284 6.1 105}	{tt0041955 6.7 1119}	{tt0047364 4.3 11}
{tt0013674 7 500}	{tt0024561 6.8 500}	{tt0028285 6.2 13}	{tt0041956 6.5 500}	{tt0047365 6.2 2363}
{tt0013679 6.9 7}	{tt0024562 5.4 10}	{tt0028286 5.9 187}	{tt0041957 7.7 10}	{tt0047366 6.4 317}

(Figure 17: Content of data blocks accessed for block size 200B)

Block #133:	Block #334:	Block #438:	Block #843:	Block #1006:
{tt0013658 6.9 31}	{tt0024561 6.8 500}	{tt0028277 7.7 500}	{tt0041933 6.8 29}	{tt0047330 5.2 5}
{tt0013662 6.9 418}	{tt0024562 5.4 10}	{tt0028278 7.6 5}	{tt0041934 6.2 146}	{tt0047331 6.1 254}
{tt0013668 6.7 22}	{tt0024563 6 93}	{tt0028279 6.5 131}	{tt0041935 6.6 124}	{tt0047333 5.6 84}
{tt0013672 6.7 25}	{tt0024564 6 55}	{tt0028280 5.7 35}	{tt0041938 6.5 159}	{tt0047334 6.2 277}
{tt0013674 7 500}	{tt0024567 6 8}	{tt0028281 6.4 51}	{tt0041939 5.9 59}	{tt0047335 5.2 30}
{tt0013679 6.9 7}	{tt0024568 6.7 25}	{tt0028282 6.5 278}	{tt0041940 6.7 155}	{tt0047336 6.6 1135}
{tt0013681 5.6 14}	{tt0024569 5.6 142}	{tt0028283 5.2 134}	{tt0041943 5.2 8}	{tt0047337 6.1 14}
{tt0013682 7.5 64}	{tt0024570 6.2 123}	{tt0028284 6.1 105}	{tt0041944 6.8 3187}	{tt0047338 5.6 344}
{tt0013687 7.1 7}	{tt0024571 6.8 9}	{tt0028285 6.2 13}	{tt0041945 6.6 314}	{tt0047339 5.4 11}
{tt0013688 6.6 642}	{tt0024573 5.7 44}	{tt0028286 5.9 187}	{tt0041946 5.5 39}	{tt0047340 5.4 29}
{tt0013690 5.3 97}	{tt0024574 7.1 49}	{tt0028287 6.9 7}	{tt0041947 6.1 517}	{tt0047341 4.4 5}
{tt0013704 6 118}	{tt0024576 6.6 19}	{tt0028288 6.6 45}	{tt0041948 6.6 902}	{tt0047342 4.4 26}
{tt0013705 5.8 136}	{tt0024577 7 12}	{tt0028289 5.5 11}	{tt0041949 6.3 355}	{tt0047343 7.8 618}
{tt0013710 7 12}	{tt0024578 7.3 1716}	{tt0028290 5.9 120}	{tt0041951 7.4 53}	{tt0047345 5.9 19}
{tt0013716 6.2 12}	{tt0024579 6.3 24}	{tt0028291 5.9 418}	{tt0041952 7.6 690}	{tt0047348 5.9 750}
{tt0013719 6.5 22}	{tt0024580 6.1 79}	{tt0028292 5.6 36}	{tt0041953 6.9 469}	{tt0047349 6.6 1352}
{tt0013724 6 8}	{tt0024581 6.8 88}	{tt0028294 7 173}	{tt0041954 7.3 2435}	{tt0047351 6.5 79}
{tt0013727 6.9 10}	{tt0024582 7.1 20}	{tt0028296 7.2 192}	{tt0041955 6.7 1119}	{tt0047353 7.1 335}
{tt0013728 5.9 49}	{tt0024586 6.4 88}	{tt0028297 5.1 8}	{tt0041956 6.5 500}	{tt0047355 5.9 99}
{tt0013730 6.8 83}	{tt0024589 5.9 38}	{tt0028298 5 11}	{tt0041957 7.7 10}	{tt0047356 7 10}
{tt0013735 5.1 73}	{tt0024590 6.4 237}	{tt0028299 6.6 23}	{tt0041958 7.6 5154}	{tt0047357 6.4 27}
{tt0013736 6.8 8}	{tt0024592 5.6 339}	{tt0028300 6.6 17}	{tt0041959 8.1 155251}	{tt0047358 5.9 224}

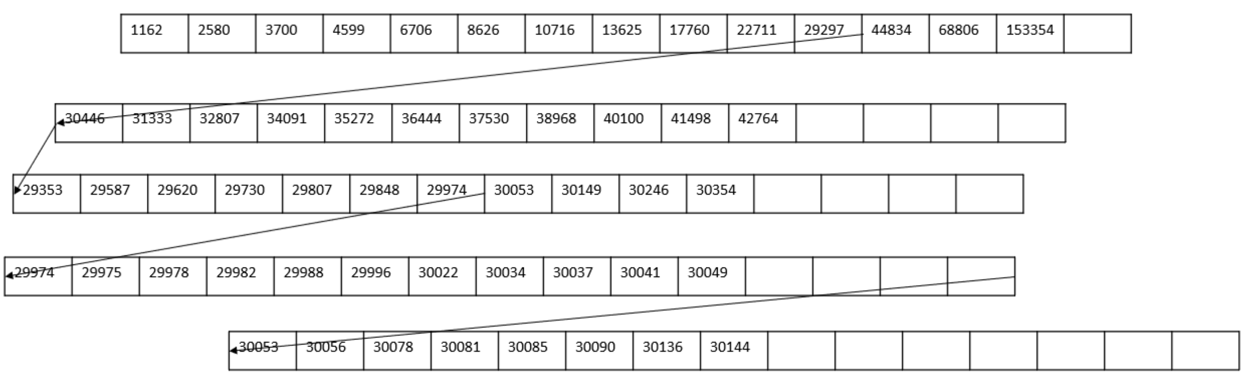
{tt0013739 6.2 6}	{tt0024593 5.7 3835}	{tt0028301 4.7 34}	{tt0041961 6 308}	{tt0047359 6.7 20}
{tt0013741 6.6 444}	{tt0024594 5.9 159}	{tt0028302 7 68}	{tt0041962 6.4 25}	{tt0047360 4.6 12}
{tt0013750 6.3 968}	{tt0024595 5.4 80}	{tt0028303 4.4 8}	{tt0041963 6.7 914}	{tt0047361 7.3 500}
{tt0013766 2.2 11}	{tt0024596 4.2 8}	{tt0028304 5.8 8}	{tt0041966 6.4 248}	{tt0047362 5.8 5}
{tt0013773 6.2 11}	{tt0024597 7.1 33}	{tt0028305 5.6 273}	{tt0041967 6.4 1793}	{tt0047363 6.7 30}

(Figure 18: Content of data blocks accessed for block size 500B)

Experiment 4

Retrieving movies with attribute “numVotes” in range 30,000 to 40,000 inclusively

Block Size	No. of index node accessed	Content of index node accessed	No. of data blocks accessed	Content of data blocks accessed	Average of “averageRating’s” returned
200B	90	Refer to figure 19	937	Refer to figure 21	6.7279124
500B	36	Refer to figure 20	918	Refer to figure 22	6.7279124



(Figure 19: Content of index nodes accessed for block size 200B)

690	1722	2548	3195	3915	4715	5651	6907	7751	8841	10526	12815	15890	19569	23113	27413	33572	42206	60581	90428
168761																			

27587	27964	28131	28320	28659	29024	29297	29603	29910	30221	30458	30639	30833	31006	31260	31527	31760	32124	32408	32635
32919	33285																		

29910	29919	29949	29954	29956	29959	29962	29974	29975	29978	29982	29988	29996	30022	30034	30037	30041	30049	30053	30056
30078	30081	30085	30090	30136	30144	30149	30158	30168	30175	30177	30195	30206							

30221	30240	30246	30247	30248	30254	30259	30262	30275	30319	30326	30333	30341	30354	30361	30370	30376	30391	30395	30402
30418	30423	30431	30446	30453	30456	30457													

30458	30462	30468	30492	30516	30522	30530	30540	30547	30548	30550	30552	30554	30569	30571	30576	30578	30585	30605	30608
30611	30619	30620	30621																

(Figure 20: Content of index nodes accessed for block size 500B)

Block #2997:	Block #968:	Block #5927:	Block #70344:	Block #52089:
{tt0054166 6.5 15}	{tt0026774 5.3 207}	{tt0091826 6.6 25}	{tt3361702 5.6 82}	{tt1456939 6.3 270}
{tt0054167 7.7 30022}	{tt0026775 6 204}	{tt0091827 3.7 589}	{tt3361726 6.8 11}	{tt1456941 6.2 30049}
{tt0054168 5.3 266}	{tt0026776 6.8 73}	{tt0091828 5.6 30037}	{tt3361740 8 5}	{tt1456944 4.2 62}
{tt0054169 5.4 443}	{tt0026777 6.9 15}	{tt0091829 5.3 4626}	{tt3361784 7.9 12}	{tt1456946 5.8 8}
{tt0054170 5.6 19}	{tt0026778 7.9 30034}	{tt0091830 7.7 6288}	{tt3361786 8.6 42}	{tt1456947 7.4 12}
{tt0054171 7.1 58}	{tt0026779 5.6 65}	{tt0091831 7.5 12}	{tt3361792 6.8 30041}	{tt1456948 6.2 11}
{tt0054172 5.6 860}	{tt0026781 6.1 327}	{tt0091832 4.4 282}	{tt3361794 8.3 9}	{tt1456949 7.2 1706}
{tt0054173 5.2 40}	{tt0026783 6.1 45}	{tt0091833 7.6 7}	{tt3361812 6.5 91}	{tt1456950 8.6 39}
{tt0054174 6.8 254}	{tt0026784 6.5 260}	{tt0091834 6.3 425}	{tt3361814 9.5 10}	{tt1456953 7.1 17}
{tt0054175 6.5 37}	{tt0026785 5.8 33}	{tt0091835 5.5 31}	{tt3361834 7 6}	{tt1456957 3 56}
{tt0054176 7.5 1906}	{tt0026786 5.9 7}	{tt0091836 5.3 2121}	{tt3361856 4.8 35}	{tt1456958 7.7 10}

(Figure 21: Content of data blocks accessed for block size 200B)

Block #1221:	Block #394:	Block #2414:	Block #28658:	Block #21221:
{tt0054166 6.5 15}	{tt0026759 5.5 680}	{tt0091801 5.8 84}	{tt3361428 6.6 17}	{tt1456875 4.9 35}
{tt0054167 7.7 30022}	{tt0026760 6.8 5}	{tt0091802 5.1 39}	{tt3361436 8.4 11}	{tt1456876 6.6 7}
{tt0054168 5.3 266}	{tt0026761 5.9 59}	{tt0091804 5.9 19}	{tt3361490 6.9 18}	{tt1456881 7 6}
{tt0054169 5.4 443}	{tt0026762 6.9 108}	{tt0091805 6.1 40}	{tt3361532 7.5 14}	{tt1456894 7.5 11}
{tt0054170 5.6 19}	{tt0026766 5.7 6}	{tt0091806 3.5 30}	{tt3361556 8.5 10}	{tt1456896 5.2 9}
{tt0054171 7.1 58}	{tt0026768 6.7 1168}	{tt0091807 5.2 55}	{tt3361572 8.7 31}	{tt1456902 5.4 5}

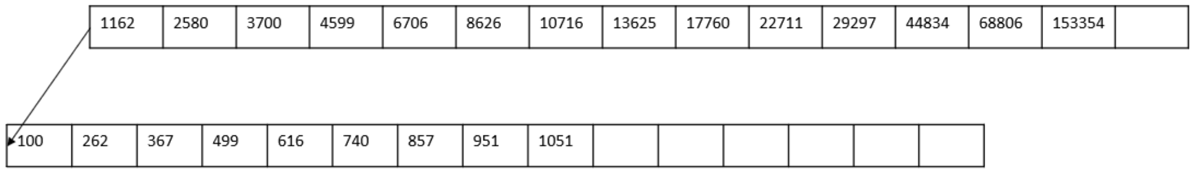
{tt0054172 5.6 860}	{tt0026769 5.9 57}	{tt0091810 6.2 1554}	{tt3361576 8.8 28}	{tt1456903 8 5}
{tt0054173 5.2 40}	{tt0026771 5.3 25}	{tt0091813 7.8 21}	{tt3361578 8.8 31}	{tt1456912 5.2 33}
{tt0054174 6.8 254}	{tt0026772 5.6 266}	{tt0091814 5.7 4328}	{tt3361580 8.8 33}	{tt1456913 6.8 12}
{tt0054175 6.5 37}	{tt0026773 4.5 146}	{tt0091815 5.7 614}	{tt3361584 9 32}	{tt1456915 5.7 80}
{tt0054176 7.5 1906}	{tt0026774 5.3 207}	{tt0091816 6.3 403}	{tt3361586 8.8 30}	{tt1456931 7.3 12}
{tt0054177 7.2 6095}	{tt0026775 6 204}	{tt0091817 7 5067}	{tt3361588 8.8 30}	{tt1456937 6.4 145}
{tt0054178 5.6 41}	{tt0026776 6.8 73}	{tt0091818 5.7 1499}	{tt3361590 8.9 29}	{tt1456939 6.3 270}
{tt0054179 5.7 11}	{tt0026777 6.9 15}	{tt0091819 5.4 214}	{tt3361614 6.5 74}	{tt1456941 6.2 30049}
{tt0054180 7.1 10}	{tt0026778 7.9 30034}	{tt0091820 6.5 99}	{tt3361618 6.8 5}	{tt1456944 4.2 62}
{tt0054181 6.3 165}	{tt0026779 5.6 65}	{tt0091821 8 280}	{tt3361630 6.3 7}	{tt1456946 5.8 8}
{tt0054182 4.7 9}	{tt0026781 6.1 327}	{tt0091823 7.8 974}	{tt3361638 7.8 91}	{tt1456947 7.4 12}
{tt0054183 7.6 73}	{tt0026783 6.1 45}	{tt0091824 5.8 69}	{tt3361644 6.2 41}	{tt1456948 6.2 11}
{tt0054184 6.2 16}	{tt0026784 6.5 260}	{tt0091825 5.9 89}	{tt3361702 5.6 82}	{tt1456949 7.2 1706}
{tt0054185 5.6 100}	{tt0026785 5.8 33}	{tt0091826 6.6 25}	{tt3361726 6.8 11}	{tt1456950 8.6 39}
{tt0054186 6.7 65}	{tt0026786 5.9 7}	{tt0091827 3.7 589}	{tt3361740 8 5}	{tt1456953 7.1 17}
{tt0054187 5.8 93}	{tt0026787 6 676}	{tt0091828 5.6 30037}	{tt3361784 7.9 12}	{tt1456957 3 56}
{tt0054188 6.5 3932}	{tt0026788 5.8 122}	{tt0091829 5.3 4626}	{tt3361786 8.6 42}	{tt1456958 7.7 10}
{tt0054189 7.8 13657}	{tt0026789 6.4 81}	{tt0091830 7.7 6288}	{tt3361792 6.8 30041}	{tt1456961 10 5}
{tt0054190 6.6 245}	{tt0026790 4.6 44}	{tt0091831 7.5 12}	{tt3361794 8.3 9}	{tt1456963 8.2 294}
{tt0054191 7 20}	{tt0026791 5.8 32}	{tt0091832 4.4 282}	{tt3361812 6.5 91}	{tt1456964 7.5 71}
{tt0054192 4.8 37}	{tt0026792 6.7 10}	{tt0091833 7.6 7}	{tt3361814 9.5 10}	{tt1456966 8.2 5}

(Figure 22: Content of data blocks accessed for block size 500B)

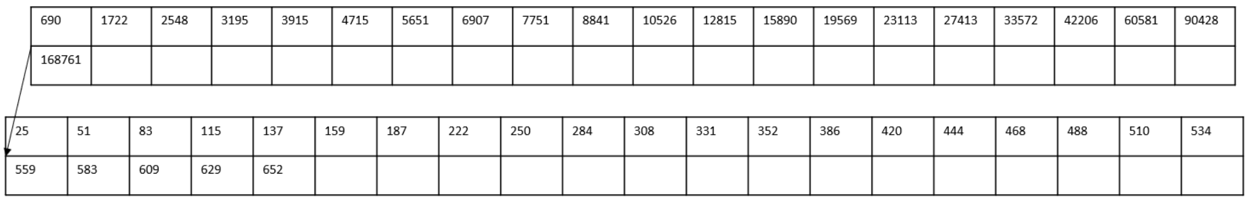
Experiment 5

Deleting movies with attribute “numVotes” equal to 1,000, and updating the B+ tree accordingly

Block Size (B)	Number of times node is deleted	Number of Nodes of updated tree	Height of updated B+ tree	Content of root node and its 1st child node
200	0	1882	4	Refer to figure 23
500	0	670	3	Refer to figure 24



(Figure 23: Content of root node and 1st child node for block size 200B)



(Figure 24: Content of root node and 1st child node for block size 500B)

Raw Experiment Data

Block size	200B	500B
Datetime accessed	1 October 2022 11:17am	1 October 2022 11:20am
Raw Data	Conducting experiment 1... New virtual storage created with capacity: 100000000b, block size: 200b Loading records from file.... Records loaded into virtal disk, total: 1070318 === Experiment 1 === Max block: 500000 Used block: 97302 Size: 19460400b (19.46MB) Usage: 19.46% === Experiment 2 === Constructing tree, it will take awhile... 100%|██████████| (1070318/1070318, 3700 it/s) Tree height: 4 Number of nodes: 1882 Parameter n: 15 Content of root node: [1162 2580 3700 4599 6706 8626 10716 13625 17760 22711 29297 44834 68806 153354 0] Content of 1st child node: [100 262 367 499 616 740 857 951 1051 0 0 0 0 0 0] === Experiment 3 === Node content while trasversing the tree (up to first 5): Node content: [1162 2580 3700 4599 6706 8626 10716 13625 17760 22711 29297 44834 68806 153354 0] Node content: [100 262 367 499 616	Conducting experiment 1... New virtual storage created with capacity: 100000000b, block size: 500b Loading records from file.... Records loaded into virtal disk, total: 1070318 === Experiment 1 === Max block: 200000 Used block: 39642 Size: 19821000b (19.82MB) Usage: 19.82% === Experiment 2 === Constructing tree, it will take awhile... 100%|██████████| (1070318/1070318, 4241 it/s) Tree height: 3 Number of nodes: 670 Parameter n: 40 Content of root node: [690 1722 2548 3195 3915 4715 5651 6907 7751 8841 10526 12815 15890 19569 23113 27413 33572 42206 60581 90428 168761 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] Content of 1st child node: [25 51 83 115 137 159 187 222 250 284 308 331 352 386 420 444 468 488 510 534 559 583 609 629 652 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] === Experiment 3 === Node content while trasversing the tree (up to first 5):

	740 857 951 1051 0 0 0 0 0 0] Node content: [514 523 533 545 559 569 583 594 604 0 0 0 0 0 0] Node content: [499 500 501 502 503 504 505 506 507 508 509 510 511 512 513] Total index node accessed: 4 Number of data blocks the process accesses: 110 Content in Block #326: {tt0013624 6.5 21} {tt0013626 6.7 2031} {tt0013627 5.2 10} {tt0013629 6.7 25} {tt0013631 6.6 12} {tt0013658 6.9 31} {tt0013662 6.9 418} {tt0013668 6.7 22} {tt0013672 6.7 25} {tt0013674 7 500} {tt0013679 6.9 7} Content in Block #819: {tt0024549 6 361} {tt0024550 6.4 24} {tt0024551 2.9 9} {tt0024553 5.9 137} {tt0024554 5.3 822} {tt0024555 5.5 195} {tt0024558 6.4 11} {tt0024559 6.1 140} {tt0024560 6.9 397} {tt0024561 6.8 500} {tt0024562 5.4 10} Content in Block #1075: {tt0028276 5.9 30} {tt0028277 7.7 500} {tt0028278 7.6 5} {tt0028279 6.5 131} {tt0028280 5.7 35} {tt0028281 6.4 51} {tt0028282 6.5 278} {tt0028283 5.2 134} {tt0028284 6.1 105} {tt0028285 6.2 13} {tt0028286 5.9 187} Content in Block #2070:	Node content: [690 1722 2548 3195 3915 4715 5651 6907 7751 8841 10526 12815 15890 19569 23113 27413 33572 42206 60581 90428 168761 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] Node content: [25 51 83 115 137 159 187 222 250 284 308 331 352 386 420 444 468 488 510 534 559 583 609 629 652 0 0 0 0 0 0 0 0 0 0 0 0 0 0] Node content: [488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] Total index node accessed: 3 Number of data blocks the process accesses: 109 Content in Block #133: {tt0013658 6.9 31} {tt0013662 6.9 418} {tt0013668 6.7 22} {tt0013672 6.7 25} {tt0013674 7 500} {tt0013679 6.9 7} {tt0013681 5.6 14} {tt0013682 7.5 64} {tt0013687 7.1 7} {tt0013688 6.6 642} {tt0013690 5.3 97} {tt0013704 6 118} {tt0013705 5.8 136} {tt0013710 7 12} {tt0013716 6.2 12} {tt0013719 6.5 22} {tt0013724 6 8} {tt0013727 6.9 10} {tt0013728 5.9 49} {tt0013730 6.8 83} {tt0013735 5.1 73} {tt0013736 6.8 8} {tt0013739 6.2 6} {tt0013741 6.6 444} {tt0013750 6.3 968} {tt0013766 2.2 11} {tt0013773 6.2 11} Content in Block #334:

	{tt0041946 5.5 39} {tt0041947 6.1 517} {tt0041948 6.6 902} {tt0041949 6.3 355} {tt0041951 7.4 53} {tt0041952 7.6 690} {tt0041953 6.9 469} {tt0041954 7.3 2435} {tt0041955 6.7 1119} {tt0041956 6.5 500} {tt0041957 7.7 10} Content in Block #2471: {tt0047356 7 10} {tt0047357 6.4 27} {tt0047358 5.9 224} {tt0047359 6.7 20} {tt0047360 4.6 12} {tt0047361 7.3 500} {tt0047362 5.8 5} {tt0047363 6.7 30} {tt0047364 4.3 11} {tt0047365 6.2 2363} {tt0047366 6.4 317} Average of averageRating: 6.7318187 === Experiment 4 === Node content while transversing the tree (up to first 5): Node content: [1162 2580 3700 4599 6706 8626 10716 13625 17760 22711 29297 44834 68806 153354 0] Node content: [30446 31333 32807 34091 35272 36444 37530 38968 40100 41498 42764 0 0 0 0] Node content: [29353 29587 29620 29730 29807 29848 29974 30053 30149 30246 30354 0 0 0 0] Node content: [29974 29975 29978 29982 29988 29996 30022 30034 30037 30041 30049 0 0 0 0] Node content: [30053 30056 30078 30081 30085 30090 30136 30144 0 0 0 0 0 0 0] Total index node accessed: 90 Number of data blocks the process accesses: 937 Content in Block #2997:	{tt0024561 6.8 500} {tt0024562 5.4 10} {tt0024563 6 93} {tt0024564 6 55} {tt0024567 6 8} {tt0024568 6.7 25} {tt0024569 5.6 142} {tt0024570 6.2 123} {tt0024571 6.8 9} {tt0024573 5.7 44} {tt0024574 7.1 49} {tt0024576 6.6 19} {tt0024577 7 12} {tt0024578 7.3 1716} {tt0024579 6.3 24} {tt0024580 6.1 79} {tt0024581 6.8 88} {tt0024582 7.1 20} {tt0024586 6.4 88} {tt0024589 5.9 38} {tt0024590 6.4 237} {tt0024592 5.6 339} {tt0024593 5.7 3835} {tt0024594 5.9 159} {tt0024595 5.4 80} {tt0024596 4.2 8} {tt0024597 7.1 33} Content in Block #438: {tt0028277 7.7 500} {tt0028278 7.6 5} {tt0028279 6.5 131} {tt0028280 5.7 35} {tt0028281 6.4 51} {tt0028282 6.5 278} {tt0028283 5.2 134} {tt0028284 6.1 105} {tt0028285 6.2 13} {tt0028286 5.9 187} {tt0028287 6.9 7} {tt0028288 6.6 45} {tt0028289 5.5 11} {tt0028290 5.9 120} {tt0028291 5.9 418} {tt0028292 5.6 36} {tt0028294 7 173} {tt0028296 7.2 192} {tt0028297 5.1 8} {tt0028298 5 11} {tt0028299 6.6 23}

	{tt0054166 6.5 15} {tt0054167 7.7 30022} {tt0054168 5.3 266} {tt0054169 5.4 443} {tt0054170 5.6 19} {tt0054171 7.1 58} {tt0054172 5.6 860} {tt0054173 5.2 40} {tt0054174 6.8 254} {tt0054175 6.5 37} {tt0054176 7.5 1906} Content in Block #968: {tt0026774 5.3 207} {tt0026775 6 204} {tt0026776 6.8 73} {tt0026777 6.9 15} {tt0026778 7.9 30034} {tt0026779 5.6 65} {tt0026781 6.1 327} {tt0026783 6.1 45} {tt0026784 6.5 260} {tt0026785 5.8 33} {tt0026786 5.9 7} Content in Block #5927: {tt0091826 6.6 25} {tt0091827 3.7 589} {tt0091828 5.6 30037} {tt0091829 5.3 4626} {tt0091830 7.7 6288} {tt0091831 7.5 12} {tt0091832 4.4 282} {tt0091833 7.6 7} {tt0091834 6.3 425} {tt0091835 5.5 31} {tt0091836 5.3 2121} Content in Block #70344: {tt3361702 5.6 82} {tt3361726 6.8 11} {tt3361740 8 5} {tt3361784 7.9 12} {tt3361786 8.6 42} {tt3361792 6.8 30041} {tt3361794 8.3 9} {tt3361812 6.5 91} {tt3361814 9.5 10} {tt3361834 7 6} {tt3361856 4.8 35}	{tt0028300 6.6 17} {tt0028301 4.7 34} {tt0028302 7 68} {tt0028303 4.4 8} {tt0028304 5.8 8} {tt0028305 5.6 273} Content in Block #843: {tt0041933 6.8 29} {tt0041934 6.2 146} {tt0041935 6.6 124} {tt0041938 6.5 159} {tt0041939 5.9 59} {tt0041940 6.7 155} {tt0041943 5.2 8} {tt0041944 6.8 3187} {tt0041945 6.6 314} {tt0041946 5.5 39} {tt0041947 6.1 517} {tt0041948 6.6 902} {tt0041949 6.3 355} {tt0041951 7.4 53} {tt0041952 7.6 690} {tt0041953 6.9 469} {tt0041954 7.3 2435} {tt0041955 6.7 1119} {tt0041956 6.5 500} {tt0041957 7.7 10} {tt0041958 7.6 5154} {tt0041959 8.1 155251} {tt0041961 6 308} {tt0041962 6.4 25} {tt0041963 6.7 914} {tt0041966 6.4 248} {tt0041967 6.4 1793} Content in Block #1006: {tt0047330 5.2 5} {tt0047331 6.1 254} {tt0047333 5.6 84} {tt0047334 6.2 277} {tt0047335 5.2 30} {tt0047336 6.6 1135} {tt0047337 6.1 14} {tt0047338 5.6 344} {tt0047339 5.4 11} {tt0047340 5.4 29} {tt0047341 4.4 5} {tt0047342 4.4 26} {tt0047343 7.8 618}

	Content in Block #52089: {tt1456939 6.3 270} {tt1456941 6.2 30049} {tt1456944 4.2 62} {tt1456946 5.8 8} {tt1456947 7.4 12} {tt1456948 6.2 11} {tt1456949 7.2 1706} {tt1456950 8.6 39} {tt1456953 7.1 17} {tt1456957 3 56} {tt1456958 7.7 10} Average of averageRating: 6.7279124 === Experiment 5 === Number of times that a node is deleted: 0 Tree height: 4 Number of nodes: 1882 Content of root node: [1162 2580 3700 4599 6706 8626 10716 13625 17760 22711 29297 44834 68806 153354 0] Content of 1st child node: [100 262 367 499 616 740 857 951 1051 0 0 0 0 0 0]	{tt0047345 5.9 19} {tt0047348 5.9 750} {tt0047349 6.6 1352} {tt0047351 6.5 79} {tt0047353 7.1 335} {tt0047355 5.9 99} {tt0047356 7 10} {tt0047357 6.4 27} {tt0047358 5.9 224} {tt0047359 6.7 20} {tt0047360 4.6 12} {tt0047361 7.3 500} {tt0047362 5.8 5} {tt0047363 6.7 30} Average of averageRating: 6.7318187 === Experiment 4 === Node content while transversing the tree (up to first 5): Node content: [690 1722 2548 3195 3915 4715 5651 6907 7751 8841 10526 12815 15890 19569 23113 27413 33572 42206 60581 90428 168761 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] Node content: [27587 27964 28131 28320 28659 29024 29297 29603 29910 30221 30458 30639 30833 31006 31260 31527 31760 32124 32408 32635 32919 33285 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] Node content: [29910 29919 29949 29954 29956 29959 29962 29974 29975 29978 29982 29988 29996 30022 30034 30037 30041 30049 30053 30056 30078 30081 30085 30090 30136 30144 30149 30158 30168 30175 30177 30195 30206 0 0 0 0 0 0 0] Node content: [30221 30240 30246 30247 30248 30254 30259 30262 30275 30319 30326 30333 30341 30354 30361 30370 30376 30391 30395 30402 30418 30423 30431 30446 30453 30456 30457 0 0 0 0 0 0 0 0 0 0 0 0 0] Node content: [30458 30462 30468

		30492 30516 30522 30530 30540 30547 30548 30550 30552 30554 30569 30571 30576 30578 30585 30605 30608 30611 30619 30620 30621 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] Total index node accessed: 36 Number of data blocks the process accesses: 918 Content in Block #1221: {tt0054166 6.5 15} {tt0054167 7.7 30022} {tt0054168 5.3 266} {tt0054169 5.4 443} {tt0054170 5.6 19} {tt0054171 7.1 58} {tt0054172 5.6 860} {tt0054173 5.2 40} {tt0054174 6.8 254} {tt0054175 6.5 37} {tt0054176 7.5 1906} {tt0054177 7.2 6095} {tt0054178 5.6 41} {tt0054179 5.7 11} {tt0054180 7.1 10} {tt0054181 6.3 165} {tt0054182 4.7 9} {tt0054183 7.6 73} {tt0054184 6.2 16} {tt0054185 5.6 100} {tt0054186 6.7 65} {tt0054187 5.8 93} {tt0054188 6.5 3932} {tt0054189 7.8 13657} {tt0054190 6.6 245} {tt0054191 7 20} {tt0054192 4.8 37} Content in Block #394: {tt0026759 5.5 680} {tt0026760 6.8 5} {tt0026761 5.9 59} {tt0026762 6.9 108} {tt0026766 5.7 6} {tt0026768 6.7 1168} {tt0026769 5.9 57} {tt0026771 5.3 25} {tt0026772 5.6 266} {tt0026773 4.5 146}

		{tt0026774 5.3 207} {tt0026775 6 204} {tt0026776 6.8 73} {tt0026777 6.9 15} {tt0026778 7.9 30034} {tt0026779 5.6 65} {tt0026781 6.1 327} {tt0026783 6.1 45} {tt0026784 6.5 260} {tt0026785 5.8 33} {tt0026786 5.9 7} {tt0026787 6 676} {tt0026788 5.8 122} {tt0026789 6.4 81} {tt0026790 4.6 44} {tt0026791 5.8 32} {tt0026792 6.7 10} Content in Block #2414: {tt0091801 5.8 84} {tt0091802 5.1 39} {tt0091804 5.9 19} {tt0091805 6.1 40} {tt0091806 3.5 30} {tt0091807 5.2 55} {tt0091810 6.2 1554} {tt0091813 7.8 21} {tt0091814 5.7 4328} {tt0091815 5.7 614} {tt0091816 6.3 403} {tt0091817 7 5067} {tt0091818 5.7 1499} {tt0091819 5.4 214} {tt0091820 6.5 99} {tt0091821 8 280} {tt0091823 7.8 974} {tt0091824 5.8 69} {tt0091825 5.9 89} {tt0091826 6.6 25} {tt0091827 3.7 589} {tt0091828 5.6 30037} {tt0091829 5.3 4626} {tt0091830 7.7 6288} {tt0091831 7.5 12} {tt0091832 4.4 282} {tt0091833 7.6 7} Content in Block #28658: {tt3361428 6.6 17} {tt3361436 8.4 11}

		{tt3361490 6.9 18} {tt3361532 7.5 14} {tt3361556 8.5 10} {tt3361572 8.7 31} {tt3361576 8.8 28} {tt3361578 8.8 31} {tt3361580 8.8 33} {tt3361584 9 32} {tt3361586 8.8 30} {tt3361588 8.8 30} {tt3361590 8.9 29} {tt3361614 6.5 74} {tt3361618 6.8 5} {tt3361630 6.3 7} {tt3361638 7.8 91} {tt3361644 6.2 41} {tt3361702 5.6 82} {tt3361726 6.8 11} {tt3361740 8 5} {tt3361784 7.9 12} {tt3361786 8.6 42} {tt3361792 6.8 30041} {tt3361794 8.3 9} {tt3361812 6.5 91} {tt3361814 9.5 10} Content in Block #21221: {tt1456875 4.9 35} {tt1456876 6.6 7} {tt1456881 7 6} {tt1456894 7.5 11} {tt1456896 5.2 9} {tt1456902 5.4 5} {tt1456903 8 5} {tt1456912 5.2 33} {tt1456913 6.8 12} {tt1456915 5.7 80} {tt1456931 7.3 12} {tt1456937 6.4 145} {tt1456939 6.3 270} {tt1456941 6.2 30049} {tt1456944 4.2 62} {tt1456946 5.8 8} {tt1456947 7.4 12} {tt1456948 6.2 11} {tt1456949 7.2 1706} {tt1456950 8.6 39} {tt1456953 7.1 17} {tt1456957 3 56} {tt1456958 7.7 10}

		{tt1456961 10 5} {tt1456963 8.2 294} {tt1456964 7.5 71} {tt1456966 8.2 5} Average of averageRating: 6.7279124 === Experiment 5 === Number of times that a node is deleted: 0 Tree height: 3 Number of nodes: 670 Content of root node: [690 1722 2548 3195 3915 4715 5651 6907 7751 8841 10526 12815 15890 19569 23113 27413 33572 42206 60581 90428 168761 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] Content of 1st child node: [25 51 83 115 137 159 187 222 250 284 308 331 352 386 420 444 468 488 510 534 559 583 609 629 652 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]