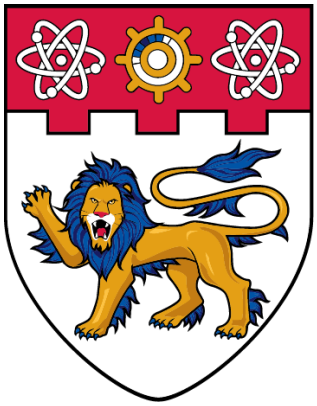


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ4031
Project 1 Report

Group 26
Nathan Soh (U2021299L)
Lim Zhi Qing (U2021897L)
Su Myat Aung (U2021467G)
Juan Samuel Sugianto (U2020871L)
Eugene Lim Zhi Jie (U2022192C)

Introduction

This report outlines our design and implementation of a database management system, which includes the storage and indexing component. The underlying data structure of the indexing component is a B+ Tree and the entire system was developed using Java.

The data to be stored in this system contains the IMDb rating and votes information for movies where:

1. tconst (string) - alphanumeric unique identifier of the title
2. averageRating - weighted average of all the individual user ratings
3. numVotes - number of votes the title has received

A sample of the data is shown below:

tconst	averageRating	numVotes
tt0000001	5.6	1645
tt0000002	6.1	198
tt0000003	6.5	1342
tt0000004	6.2	120

1. Design of the Storage Component

Fields
As mentioned above, the data contains 3 types of attributes: tconst, numVotes and averageRating. These are the design considerations for the attributes:

tconst
Since tconst is always represented by an array of 9 characters, a **String** would be a suitable data type to store this attribute. Since it is always 9 characters, the tconst field would be **9 bytes**.

averageRating
The averageRating attribute is represented by a decimal number between 0.0 and 10.0 which is fixed at one decimal point. Hence, the **Float** data type would be suitable to store this attribute as a float can store decimal numbers to a precision of 6 to 7 digits. This would make the size of the averageRating be the size of a float, which is **4 bytes**.

numVotes
The numVotes attribute is represented by a non-negative whole number. Therefore, the **Integer (int)** data type would be sufficient to store this attribute as the int data type can store up to +2147483647. This would make the size of the numVotes field to be the size of an int, which is **4 bytes**.

Field	Data Type	Size
tconst	String (string)	9 Bytes
averageRating	Float (float)	4 Bytes
numVotes	Integer (int)	4 Bytes

Records

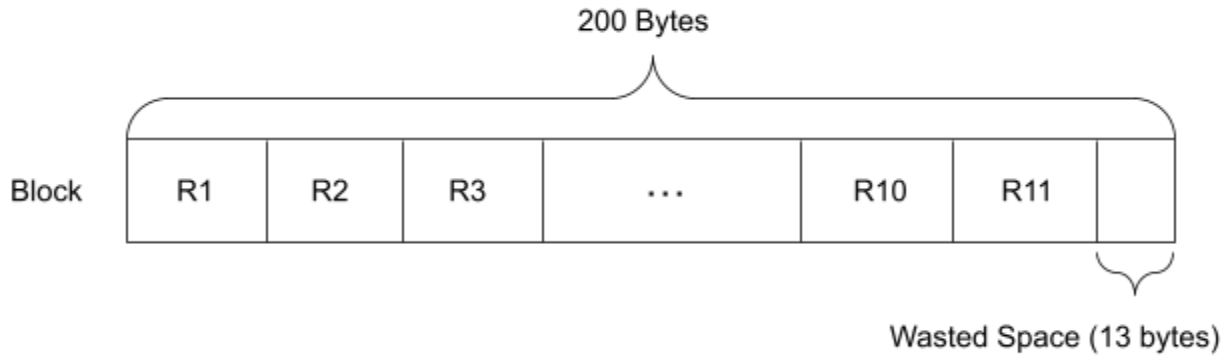
A Record class is built to implement the record in a database and it packs the fields of the same tuple together. Hence the attributes of this class are the fields outlined above. As all the fields are fixed length, we have implemented the records to have a fixed format and fixed length, which will make reading the data easier.

Record (17 bytes)		
Tconst (9 bytes)	**averageRating (4 bytes)**	**numVotes (4 bytes)**
tt0000001	5.6	1645

Blocks

A Block class is built to implement a block in memory and it packs records together. The block is implemented by having the attribute “records” which is an array of Records to store the records in the block, allowing us to store the records sequentially. We have also chosen to implement the block to be unspanned, meaning that all records must be able to fit within a block. Although this will cause space to be wasted, it would allow easy searching for records within the blocks.

Considering that a block size is 200 bytes, and the size of a record is 17 bytes, each block would be able to hold 11 records. This can be visualised with the diagram below:



2. Design of the B+ Tree

Design of B+ Tree component

In the implementation, the B+ Tree is a structure that contains the following objects:

Block	Data Type
records	List<Record>

records: a pointer variable that points to a list of records in a block

Non-Leaf Node	Data Type
children	List<Node>

children: a pointer to the children nodes for each internal node

Leaf Node	Data Type
record	List<List<RecordBlock>>
prevNode	Node
nextNode	Node
parentNode	Non-Leaf Node

record: a list of record blocks that share the same key value
prevNode: a pointer to the left neighbouring leaf node for easier deletion and node coalescing
nextNode: a pointer to the adjacent neighbouring leaf node, which is used for insertion, deletion and searching of keys
parentNode: a pointer to the parent internal node for node merging

When a record is inserted when the B+ Tree is initially empty, a root node is created and the record is appended to the pointer of that root node/key. Else, when the B+ Tree is not empty, subsequent records will be added to the root node. If the root node already maintains the maximum number of keys possible, a new child node will be created and the height of the B+ Tree increases.
When deleting keys, the deleteKey() function first searches for the bucket in the B+ Tree that contains the target key. If the deleted key was in the first index of the node, the parent node is updated to reflect the change in the integer key. Otherwise, the function will attempt to borrow a key from the left or right sibling node if the node contains less than the minimum number of keys required for a node to exist in the B+ Tree. If the

borrowing of keys from sibling nodes is not permissible, the function will attempt to merge the two nodes and update the parent nodes accordingly.

Data structure of a node

In the B+ Tree, a node is a structure that contains *n* keys, and *n*+1 pointers. The keys of a B+ Tree are integers that are sorted within each node, and across nodes. The following figure belows depicts the structure of such a node.

In an internal node, each pointer points to a child node, corresponding to a range of key values. The *i* th key is equal to the lower bound of the range pointed by the *i*+1 th pointer. In a leaf node, the first *n* pointers point to a 17 byte record which contains a 9 byte tconst, 4 bytes of numVotes and 4 bytes of averageRating. The last pointer of each node then points to the head of the next node.

Maximum number of keys in a node

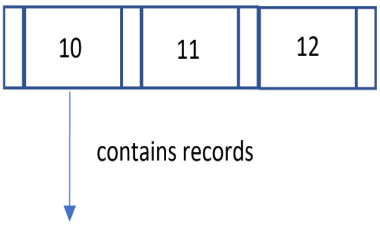
When designing the B+ Tree, the maximum number of keys a node can contain is denoted by the parameter *n*. Each node of the B+ Tree would either have 200 or 500 bytes. Considering that each node would have *n* number of keys and *n*+1 pointers, we can calculate the parameter *n* with the following formula.

$$n = (block\ size - pointer\ size)/(pointer\ size + key\ size)$$

where *block size* = 200 or 500 bytes. In the case of a 64-bit operating system, the *pointer size* would be 8 bytes, and *key size* would be 4 bytes due to keys declared as the Integer data type. Following this formula, we can calculate that a B+ Tree with a block size of 200 bytes would maintain a maximum of 16 keys per node, and a B+ Tree with a block size of 500 bytes would maintain a maximum of 41 keys per node.

Duplicate Records

In order to deal with the insertion of multiple records with the same key value in the dataset, a list was chosen to store the records.




```
[[RecordBlock(new Record("10", 1, 1), blk 1)],[RecordBlock(new Record("10", 3, 2), blk 5)]]
```

During the insertion of a duplicate record, the appropriate location will be identified in the leaf node.

If there are records with the same key that are already stored within the B+tree, the new record will be appended into the same key value. If no existing key is found, a new leaf node entry will be made.

3. Results

200B Blocks

Experiment 1

Total Memory Size: 500.0 MB
Size of Database: 18.195406 MB
Block Size: 200 bytes
Record Size: 17 bytes
Number of Blocks Allocated: 97302 blocks
Number of Blocks Remaining: 2402698 blocks
Total Number of Records: 1070318

Experiment 2

Parameter *n* of the B+ Tree: 16
Number of nodes in the B+ Tree: 1747
Height of the B+ Tree: 4

Contents of the root node:

Root Node									
1342	2631	3897	5970	8878	12134	20151	30034	49802	125979

Contents of the first child node:

First Child Node								
106	267	448	573	726	837	960	1122	1224

Experiment 3

Number of index nodes accessed: 5

Contents of first five index nodes:

Index node	Contents
1	1342, 2631, 3897, 5970, 8878, 12134, 20151, 30034, 49802, 125979
2	106, 267, 448, 573, 726, 837, 960, 1122, 1224
3	458, 468, 478, 487, 501, 514, 525, 538, 552, 561
4	487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500
5	501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513

Number of data blocks accessed: 110

Contents of first five data blocks:

Data block	Contents
1	tt0013624, tt0013626, tt0013627, tt0013629, tt0013631, tt0013658, tt0013662, tt0013668, tt0013672, tt0013674, tt0013679
2	tt0024549, tt0024550, tt0024551, tt0024553, tt0024554, tt0024555, tt0024558, tt0024559, tt0024560, tt0024561, tt0024562
3	tt0028276, tt0028277, tt0028278, tt0028279, tt0028280, tt0028281, tt0028282, tt0028283, tt0028284, tt0028285, tt0028286
4	tt0041946, tt0041947, tt0041948, tt0041949, tt0041951, tt0041952, tt0041953, tt0041954, tt0041955, tt0041956, tt0041957
5	tt0047356, tt0047357, tt0047358, tt0047359, tt0047360, tt0047361, tt0047362, tt0047363, tt0047364, tt0047365, tt0047366

The accessed data blocks refer to the blocks created by the program in Experiment 1 on the disk. The highlighted records in each data block contain movies with “numVotes” equal to 500.

Average of “averageRating’s” of records returned: 6.732

Experiment 4

Number of index nodes accessed: 85

Contents of first five index nodes:

Index node	Contents
1	1342, 2631, 3897, 5970, 8878, 12134, 20151, 30034, 49802, 125979
2	20937, 21812, 22631, 23444, 24703, 26119, 27317, 28769
3	28916, 29004, 29116, 29268, 29387, 29594, 29633, 29743, 29848, 29959
4	29959, 29962, 29974, 29975, 29978, 29982, 29988, 29996, 30022
5	30034, 30037, 30041, 30049, 30053, 30056, 30078, 30081, 30085, 30090

Number of data blocks accessed: 953

Contents of first five data blocks:

Data block	Contents
1	tt0054166, tt0054167, tt0054168, tt0054169, tt0054170, tt0054171, tt0054172, tt0054173, tt0054174, tt0054175, tt0054176
2	tt0026774, tt0026775, tt0026776, tt0026777, tt0026778, tt0026779, tt0026781, tt0026783, tt0026784, tt0026785, tt0026786
3	tt0091826, tt0091827, tt0091828, tt0091829, tt0091830, tt0091831, tt0091832, tt0091833, tt0091834, tt0091835, tt0091836
4	tt3361702, tt3361726, tt3361740, tt3361784, tt3361786, tt3361792, tt3361794, tt3361812, tt3361814, tt3361834, tt3361856
5	tt1456939, tt1456941, tt1456944, tt1456946, tt1456947, tt1456948, tt1456949, tt1456950, tt1456953, tt1456957, tt1456958

The accessed data blocks refer to the blocks created by the program in Experiment 1 on the disk. The highlighted records in each data block contain movies with “numVotes” between 30,000 and 40,000 inclusively.

Average of “averageRating’s” of records returned: 6.728

Experiment 5

Number of deleted nodes = 0
Number of nodes = 1747
Height of B+ Tree= 4

Contents of the root node:

Root Node									
1342	2631	3897	5970	8878	12134	20151	30034	49802	125979

Contents of the first child node:

First Child Node								
106	267	448	573	726	837	960	1122	1224

500B Blocks

Experiment 1

Total Memory Size: 500.0 MB
Size of Database: 18.195406 MB
Block Size: 500 bytes
Record Size: 17 bytes
Number of Blocks Allocated: 39642 blocks
Number of Blocks Remaining: 960358 blocks
Total Number of Records: 1070318

Experiment 2

Parameter *n* of the B+ Tree: 41
Number of nodes in the B+ Tree: 658
Height of the B+ Tree: 3

Contents of the root node:

Root Node
810, 1970, 2998, 3824, 4754, 5795, 6963, 8596, 10593, 12944, 15653, 19079, 23444, 28454, 33701, 41156, 50539, 72884, 106475, 209225 (20 keys)

Contents of the first child node:

First Child Node
26, 59, 88, 120, 160, 188, 222, 253, 291, 316, 338, 360, 395, 418, 444, 484, 506, 528, 553, 575, 611, 652, 676, 699, 727, 754, 780 (27 keys)

Experiment 3

Number of index nodes accessed: 3

Contents of the three index nodes:

Index node	Contents
1	810, 1970, 2998, 3824, 4754, 5795, 6963, 8596, 10593, 12944, 15653, 19079, 23444, 28454, 33701, 41156, 50539, 72884, 106475, 209225
2	26, 59, 88, 120, 160, 188, 222, 253, 291, 316, 338, 360, 395, 418, 444, 484, 506, 528, 553, 575, 611, 652, 676, 699, 727, 754, 780
3	484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505

Number of data blocks accessed: 110

Contents of first five data blocks:

Data block	Contents
1	tt0013658, tt0013662, tt0013668, tt0013672, tt0013674, tt0013679, tt0013681, tt0013682, tt0013687, tt0013688, tt0013690, tt0013704, tt0013705, tt0013710, tt0013716, tt0013719, tt0013724, tt0013727, tt0013728, tt0013730, tt0013735, tt0013736, tt0013739, tt0013741, tt0013750, tt0013766, tt0013773
2	tt0024561, tt0024562, tt0024563, tt0024564, tt0024567, tt0024568, tt0024569, tt0024570, tt0024571, tt0024573, tt0024574, tt0024576, tt0024577, tt0024578, tt0024579, tt0024580, tt0024581, tt0024582, tt0024586, tt0024589, tt0024590, tt0024592, tt0024593, tt0024594, tt0024595, tt0024596, tt0024597
3	tt0028277, tt0028278, tt0028279, tt0028280, tt0028281, tt0028282, tt0028283, tt0028284, tt0028285, tt0028286, tt0028287, tt0028288, tt0028289, tt0028290, tt0028291, tt0028292, tt0028294, tt0028296, tt0028297, tt0028298, tt0028299, tt0028300, tt0028301, tt0028302, tt0028303, tt0028304, tt0028305
4	tt0041933, tt0041934, tt0041935, tt0041938, tt0041939, tt0041940, tt0041943, tt0041944, tt0041945, tt0041946, tt0041947, tt0041948, tt0041949, tt0041951, tt0041952, tt0041953, tt0041954, tt0041955, tt0041956, tt0041957, tt0041958, tt0041959, tt0041961, tt0041962, tt0041963, tt0041966, tt0041967
5	tt0047330, tt0047331, tt0047333, tt0047334, tt0047335, tt0047336, tt0047337, tt0047338, tt0047339, tt0047340, tt0047341, tt0047342, tt0047343, tt0047345, tt0047348, tt0047349,

	tt0047351, tt0047353, tt0047355, tt0047356, tt0047357, tt0047358, tt0047359, tt0047360, tt0047361, tt0047362, tt0047363

The accessed data blocks refer to the blocks created by the program in Experiment 1 on the disk. The highlighted records in each data block contain movies with “numVotes” equal to 500.

Average of “averageRating’s” of records returned: 6.732

Experiment 4

Number of index nodes accessed: 37

Contents of first five index nodes:

Index node	Contents
1	810, 1970, 2998, 3824, 4754, 5795, 6963, 8596, 10593, 12944, 15653, 19079, 23444, 28454, 33701, 41156, 50539, 72884, 106475, 209225
2	28603, 28890, 29099, 29297, 29730, 29956, 30158, 30457, 30658, 30865, 31144, 31440, 31607, 31768, 32028, 32195, 32470, 32644, 33044, 33357
3	29956, 29959, 29962, 29974, 29975, 29978, 29982, 29988, 29996, 30022, 30034, 30037, 30041, 30049, 30053, 30056, 30078, 30081, 30085, 30090, 30136, 30144, 30149
4	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
5	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, 30639

Number of data blocks accessed: 953

Contents of first five data blocks:

Data block	Contents
1	tt0054166, tt0054167, tt0054168, tt0054169, tt0054170, tt0054171, tt0054172, tt0054173, tt0054174, tt0054175, tt0054176, tt0054177, tt0054178, tt0054179, tt0054180, tt0054181, tt0054182, tt0054183, tt0054184, tt0054185, tt0054186, tt0054187, tt0054188, tt0054189, tt0054190, tt0054191, tt0054192
2	tt0026759, tt0026760, tt0026761, tt0026762, tt0026766, tt0026768, tt0026769, tt0026771, tt0026772, tt0026773, tt0026774, tt0026775, tt0026776, tt0026777, tt0026778, tt0026779, tt0026781, tt0026783, tt0026784, tt0026785, tt0026786, tt0026787, tt0026788, tt0026789, tt0026790, tt0026791, tt0026792
3	tt0091801, tt0091802, tt0091804, tt0091805, tt0091806, tt0091807, tt0091810, tt0091813, tt0091814, tt0091815, tt0091816, tt0091817, tt0091818, tt0091819, tt0091820, tt0091821, tt0091823, tt0091824, tt0091825, tt0091826, tt0091827, tt0091828, tt0091829, tt0091830, tt0091831, tt0091832, tt0091833

4	tt3361428, tt3361436, tt3361490, tt3361532, tt3361556, tt3361572, tt3361576, tt3361578, tt3361580, tt3361584, tt3361586, tt3361588, tt3361590, tt3361614, tt3361618, tt3361630, tt3361638, tt3361644, tt3361702, tt3361726, tt3361740, tt3361784, tt3361786, tt3361792, tt3361794, tt3361812, tt3361814
5	tt1456875, tt1456876, tt1456881, tt1456894, tt1456896, tt1456902, tt1456903, tt1456912, tt1456913, tt1456915, tt1456931, tt1456937, tt1456939, tt1456941, tt1456944, tt1456946, tt1456947, tt1456948, tt1456949, tt1456950, tt1456953, tt1456957, tt1456958, tt1456961, tt1456963, tt1456964, tt1456966

The accessed data blocks refer to the blocks created by the program in Experiment 1 on the disk. The highlighted records in each data block contain movies with “numVotes” between 30,000 and 40,000 inclusively.

Average of “averageRating’s” of records returned: 6.728

Experiment 5

Number of deleted nodes: 0
Number of nodes: 658
Height of B+ Tree: 3

Contents of the root node:

Root Node
810, 1970, 2998, 3824, 4754, 5795, 6963, 8596, 10593, 12944, 15653, 19079, 23444, 28454, 33701, 41156, 50539, 72884, 106475, 209225

Contents of the first child node:

First Child Node
26, 59, 88, 120, 160, 188, 222, 253, 291, 316, 338, 360, 395, 418, 444, 484, 506, 528, 553, 575, 611, 652, 676, 699, 727, 754, 780