Text

Description automatically generated

**CZ4031: DATABASE SYSTEM PRINCIPLES**

**Assignment 1**

***3 October 2021***

**Group 20**

Tok Jing Xian

Chan Zhao Hui

Leow Wei Thou, Samuel

Soham Bhadra (U1822379K)

Table of Contents

[INTRODUCTION 3](#_Toc83923818)

[Description 3](#_Toc83923819)

[Implementation overview 3](#_Toc83923820)

[Dataset attributes 3](#_Toc83923821)

[STORAGE COMPONENT 4](#_Toc83923822)

[Record 4](#_Toc83923823)

[Disk Block 4](#_Toc83923824)

[EXPERIMENTS 5](#_Toc83923825)

[Experiment 1 5](#_Toc83923826)

# INTRODUCTION

## Description

In this project, we design and implement a simple storage and database system using C++ that uses B+ trees for indexing records. We support inserting, searching for and deleting records. We use a single C++ file containing all the functions.

## Implementation overview

We have organised our program into the following structures and classes:

* struct Record, containing record attributes and methods and a function toString for printing record values.
* struct Disk\_Block and struct Bucket, containing (de)initialisation methods.
* class BPlusTree, which contains all the methods related to the B+ tree as explained in the B+ Tree Implementation section.

### Dataset attributes

The dataset (data.tsv) used for this project contains IMDb IDs, ratings and votes for movies. The following are the attributes in the dataset:

* tconst: alphanumeric unique identifier of the title
* averageRating : weighted average of all the individual user ratings
* numVotes: number of votes the title has received

The following experiments are written in the C++ programming language to design the storage of data and the B+ tree.

Sample record in data.tsv:

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data type** | **Data example** |
| tconst | String | tt0000001 |
| averageRating | float | 5.6 |
| numVotes | int | 1645 |

Data types used in this project:

|  |  |
| --- | --- |
| **Data Type** | **Storage** |
| Integer / Unsigned Integer | 4 bytes |
| Float | 4 bytes |

# STORAGE DESIGN AND STRUCTURE

As per the project requirements, we have defined the disk size as 108 bytes or 100 MB and the block size as 100 bytes.

### Record

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Information |
| id | int | tconst (only the numeric value is use) |
| avg\_rating | float | Average rating |
| num\_of\_votes | int | Number of votes |

Total size of 1 record = 12 bytes.

### Disk Block

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Information |
| id | int | Header of the disk block |
| Record | Object | Records size |

To get number of records stored in a disk block, we use the following calculation:

**For block size = 100 bytes:**

Number of records per block = (Block size - size of Integer) / Record size = **8**

**For block size = 100 bytes:**

Number of records per block = (Block size - size of Integer) / Record size = **41**

# B+ TREE DESIGN AND IMPLEMENTATION

class BplusTree (insertIntoBucket, insertIntoLeaf, searchForLeafNodeWithKey, searchAndPrintLeafNode, searchForLeftLeafSiblingOfKey, searchAndPrintExperimentFour, insertChildNode, splitFullLeafNodeForInsert, insertIntoFullNonleafNode, insertParentUpdate, addRecord, getBucket, changeKeyParentUpdate, getKeyPositionInNode, getNodePositionInParent, getLeafSiblings, deleteParentUpdate, deleteFullBucket, deleteRecord, getParentNode, getLeafParent, displayTree)

getTotalRecordCount, retrieveData,

# EXPERIMENTS

## Experiment 1

Block size = 100 bytes

Number of blocks utilized: 133790

Size of database: 12.7592MB

Block size = 500 bytes

Number of blocks utilized: 26106

Size of database: 12.4483MB