**Database Management System – Phase 1**

**SID**: 202005204 **STUDENT NAME:** Salma Eletreby **Email**: Se2005204@qu.edu.qa

**SID**: 202003532 **STUDENT NAME:** Aya Muhanad Hassan **Email**: ah2003532@qu.edu.qa

**SID**: 202001942 **STUDENT NAME:** Hissa Al Muhannadi **Email**: ha2001942@qu.edu.qa

**Course number**: CMPS 451

**Submission date**: 12th November 2023

**Class section:** L51

Table of Contents

[Design Choices 3](#_Toc150628457)

[Backend Design 3](#_Toc150628458)

[Frontend design 3](#_Toc150628459)

[Summary of Metadata 3](#_Toc150628460)

[Queries Supported 3](#_Toc150628461)

[GitHub Link 3](#_Toc150628462)

[Metadata Screenshots 3](#_Toc150628463)

[Table metadata 3](#_Toc150628464)

[Attribute metadata 4](#_Toc150628465)

[Index attribute 4](#_Toc150628466)

[Application Screenshots 4](#_Toc150628467)

[Main Screen 4](#_Toc150628468)

[Statistics Screen 5](#_Toc150628469)

[Statistics for dept table 5](#_Toc150628470)

[Statistics for course table 5](#_Toc150628471)

[Cost Screen 5](#_Toc150628472)

[Equijoin 5](#_Toc150628473)

[Selection on primary key 6](#_Toc150628474)

[Selection on secondary key 6](#_Toc150628475)

# Design Choices

## Backend Design

For our backend design, we created 3 relations in our MySQL database to act as our catalogue. We connected this database with Java using connection JDK.

We chose creating relations in MySQL to represent our catalogue as we agreed that this is similar approach to the actual way catalogue is stored and viewed in actual MySQL databases and it is easier than populating the actual tables with enough records to give accurate numbers.

For our cost algorithms, we implemented it in Java as a class with multiple methods and we called this class and its methods in the screens.

## Frontend design

For front design, we went with Swing interface in Java because it’s easier, faster and cleaner of Java’s UI interface design options.

# Summary of Metadata

For each relation, we recorded meta data of the field that is primary index, the field that is secondary index, the field that is clustering index with null if any of those indexes are not available. We stored average row(record length) in bytes, table size in bytes, number of blocks allocated and the blocking factor as well as number of records.

For each attribute, we record the data type, if it’s a key its type otherwise null and if its unique or not. Additionally, we store if the relation is sorted on that attribute, if it’s used for an index and if its nullable. Furthermore, the number of distinct values, the minimum value if it’s a numeric data type and maximum if its numeric type.

For each index, we recorded the attribute its built on, its type (primary, secondary, clustering) and the structure of the index file. Additionally, we store if its unique or not as well, height of tree if file structure is a tree and its distinct values as well as first level index.

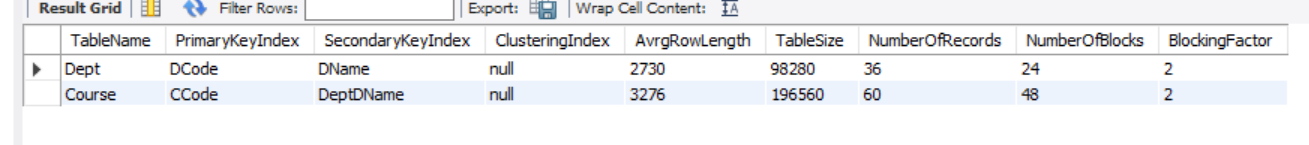
# Queries Supported

# GitHub Link

<https://github.com/salma-se2005204/DBMS-Project.git>

# Metadata Screenshots

## Table metadata



## Attribute metadata

A screenshot of a computer

Description automatically generated

## Index attribute

A screenshot of a computer

Description automatically generated

# Application Screenshots

## Main Screen

A screenshot of a computer

Description automatically generated

## Statistics Screen

### Statistics for dept table

A screenshot of a computer

Description automatically generated

### Statistics for course table

A screenshot of a computer

Description automatically generated

## Cost Screen

### Equijoin

|  |  |
| --- | --- |
| Returns all columns | Returns specific columns |
|  |  |

### Selection on primary key

|  |  |
| --- | --- |
| selection using a primary key and equality operator | selection using a primary key with range operator |
|  |  |

### Selection on secondary key

A screenshot of a computer

Description automatically generated

### Selection on non-key

|  |  |
| --- | --- |
| Selection on non-key on and equality operator | Selection on non-key on and range operator |
|  |  |
| Selection on non-key using AND | Selection on non-key using OR |
|  |  |