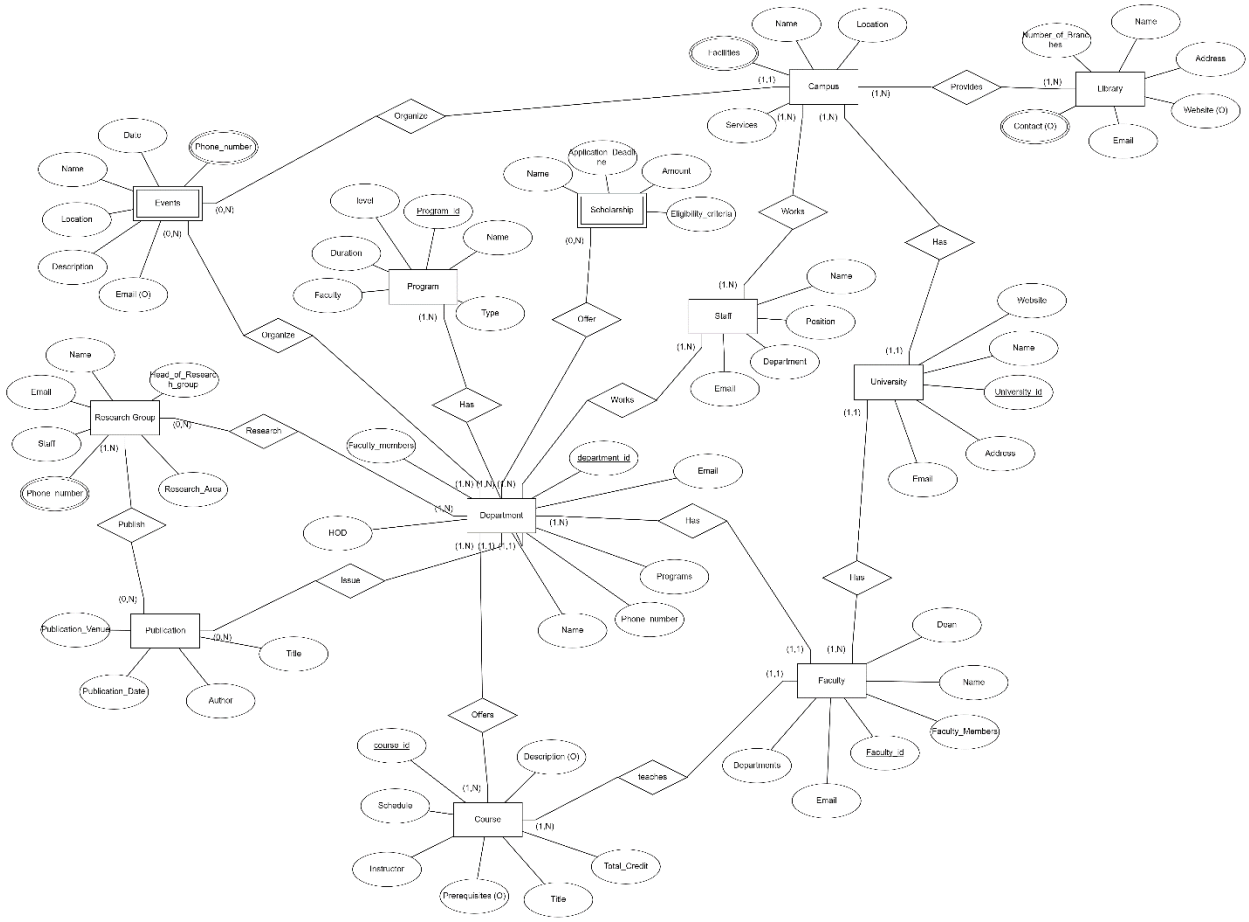
Assignment-1 Report

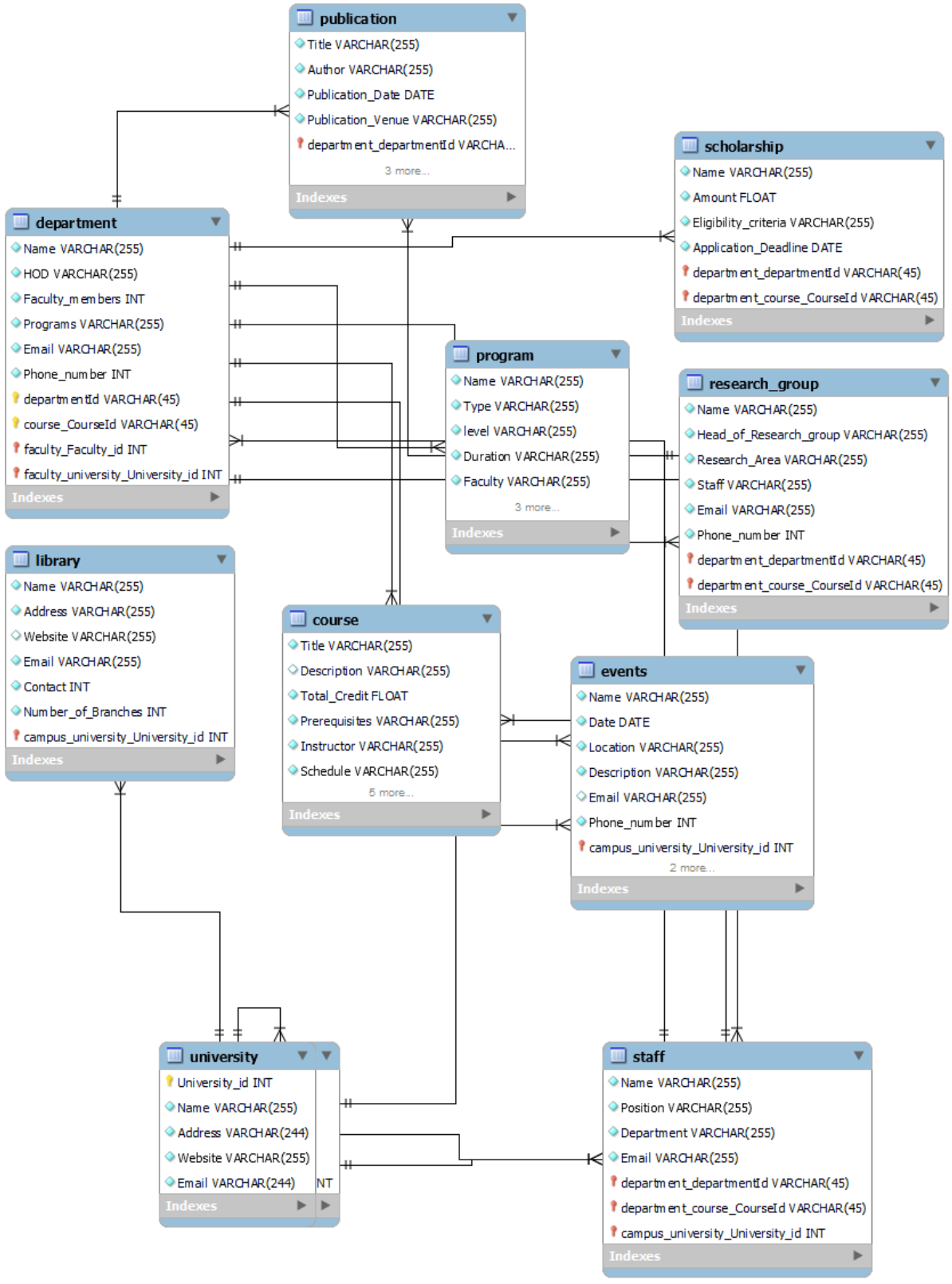
Problem - 1

The conceptual ERD for the 12 entities listed could include the following entities and relationships:

- **University**: This entity have attributes such as Name, Address, Website, Phone number, Email address, and Mission statement. It has a one-to-many relationship with entities such as Faculty and Campus.
- **Faculty**: This entity have attributes such as Name, Dean, Department(s) within the faculty, Number of students, and Academic programs offered. It has a many-to-one relationship with University, and a one-to-many relationship with entities such as Department and Course.
- **Library**: This entity have attributes such as Name, Address, Website, Phone number, and Collection size. It has a one-to-many relationship with Campus.
- **Events**: This entity have attributes such as Name, Date(s), Location, and Description. It has a many-to-one relationship with entities such as Department and Campus. It is Considered as week entity because it does not exists without campus.
- **Department**: This entity have attributes such as Name, Head of department, Faculty members, and Academic programs offered. It has a many-to-one relationship with entities such as Faculty, and a one-to-many relationship with entities such as Course and Research Group.
- **Research Group**: This entity have attributes such as Name, Head of research group, and Research focus/areas. It has a many-to-one relationship with Department.
- **Campus**: This entity have attributes such as Name, Location/address, Facilities, and Services. It has a many-to-one relationship with University and a one-to-many relationship with entities such as Library and Events.
- **Course**: This entity have attributes such as Title, Description, Number of credits, and Prerequisites. It has a many-to-one relationship with Department and a many-to-one relationship with Faculty.
- **Staff**: This entity have attributes such as Name, Job title/position, Department/office, and Work hours/schedule. It has a many-to-one relationship with entities such as Department and Campus.
- **Scholarship**: This entity have attributes such as Name, Amount, Eligibility criteria, and Application deadline. It has a many-to-one relationship with entities such as Program and Department.
- **Publication**: This entity have attributes such as Title, Author(s), Publication date, and Publication venue. It has a many-to-many relationship with entities such as Research Group and Department.
- **Program**: This entity represents the several education programmes that each department offers, such as Bachelor of Computer Science, Master of Chemistry, and so on. entity has a unique identifier that distinguishes it from other entities, so it is considered as a strong entity.



Conceptual Model : Using ERDPlus Website



Logical Model : MySQLWorkBench tool

The model reveals the following dependencies:

- A Faculty entity is associated with a University entity through a foreign key.
- A Library entity is not dependent on any other entities.
- An Event entity is not dependent on any other entities.
- A Department entity is associated with a Faculty entity through a foreign key.
- A Research Group entity is associated with a Department entity through a foreign key.
- A Campus entity is associated with a University entity through a foreign key.
- A Course entity is associated with a Department entity through a foreign key.
- A Staff entity is associated with a Department entity through a foreign key.
- Contact Numbers in each attributes have been defined as multivalued attributes.

Normalization in db

Events
Phone_number (multivaled attribute)
Date
Name

Converting to 1NF ->

unique key multivalue

Email	Phone

1NF Form

CAMPUS
Facilities (Multivalues Attribute)
Name
Location
Services

Converting to 1NF ->

Unique key Multivalue

Location	Facilities

1NF Form

Here Both the tables Campus and Events attributes are not in 1NF Form, so we have to perform some normalization to reduce the redundant value from the table.

I have provided Sql queries for above conceptual erd, which will create Physical ERD. Please find attached sql file for the physical model.

Problem – 2

Problem 2 required the development of a light-weight DBMS prototype using the Java programming language without using any third-party libraries. The application is console-based and accepts user input in the form of SQL queries, once the user has successfully logged in. The application provides functionality for creating one database only.

The following requirements were strictly followed:

- A standard Java IDE was used to develop the application, and any JDK version is acceptable.
- JavaDocs specification for commenting styles was followed while writing the Java code.
- Design principles were used, such as SOLID design principles.
- The application was console-based, and it accepts user input in the form of SQL queries.
- Two-factor authentication was implemented for user authentication, using ID, Password, and question/answer for authentication.
- Custom-designed persistent storage was used to store user information, logs, and data in a file format.
- Implementation of queries (DDL & DML) – CREATE, SELECT, INSERT, UPDATE, DELETE was implemented.

The application architecture is based on the Model-View-Controller (MVC) design pattern, which helps to separate the application's concerns and improve the application's maintainability, flexibility, and extensibility.

user interface was developed using the Java Scanner class to read user input from the console. The application uses Java's standard library to implement hashing functions for user authentication and to create custom-designed persistent storage to store user information, logs, and data.

The application's business logic was implemented using Java classes, including a User class to represent the user's information, a Query class to represent the SQL queries, a Database class to represent the database, and a Controller class to handle the application's logic.

It provides two-factor authentication using ID, Password, and question/answer for authentication. The user's information, logs, and data are stored in a custom-designed persistent storage file format, which uses delimiters to store and access data within a text file.

It supports the implementation of queries (DDL & DML) – CREATE, SELECT, INSERT, UPDATE, DELETE, applied to any number of tables. This was achieved by creating a Query class, and for every query, a separate method was built.

Overall, the application meets the requirements of Problem 2, and the implementation of the application's features was done using Java's standard library. The application's architecture is based on the MVC design pattern, which improves the application's maintainability, flexibility, and extensibility.

Java file in the project:

- Console.java – which will take input from the use for the queries
- Dbms.java – this java file is main file which performs main execution of the java code, and invoke all the function from there
- QueryParser.java – this is the crucial file which will parse all queries and perform validations
- QueryResult.java – which will print the query result from the code
- UserAuthenticationModule.java – the authentication module will perform user authentication and will only allow user to access the db which they are allowed to.

Functionalities

- The code will first perform authentication of the user, with using console as an input
- It will ask for the username and password, if the user is not registered then it will register the user, by asked the security question and answer, performs two factor authentication.
- Then it will take user into the console of the application asking the queries to input

Here user can input basic CRUD queries.

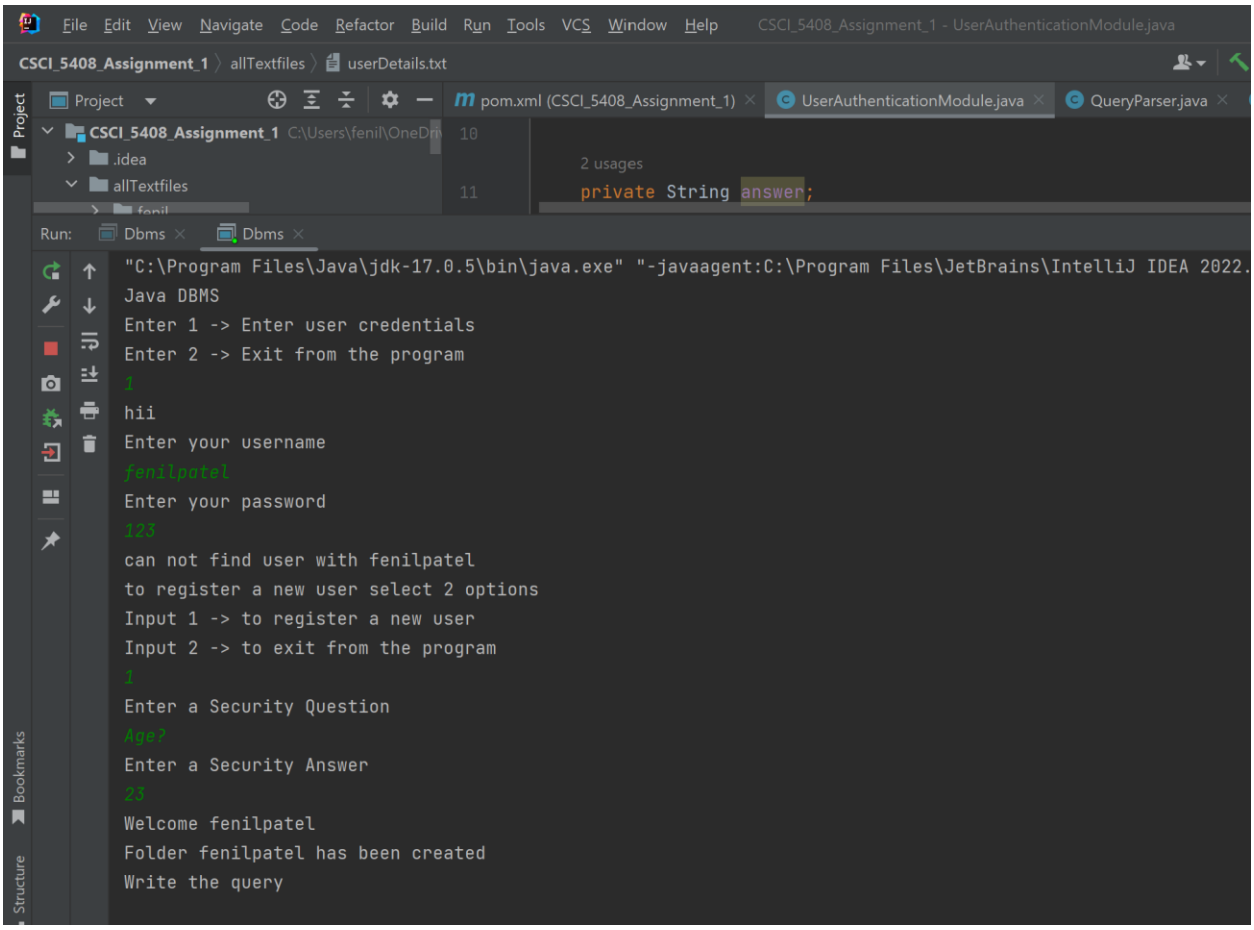
- Some of the user input queries as stated below
- Insert into Persons (PersonID, LastName, FirstName) VALUES ('123','Patel','Fenil');
- SELECT * FROM Persons WHERE PersonID = 123 AND FirstName = 'Fenil';
- Select * from Persons;
- Select FirstName, LastName From Persons WHERE PersonID = 124 AND FirstName = 'Sarthak';
- Select FirstName from Persons where PersonID = 124;
- Select * from Persons where PersonID = 124;
- UPDATE Persons SET FirstName = 'Harsh' WHERE PersonID = 124;
- DELETE FROM Persons WHERE FirstName='Fenil';

The QueryParser method will use the regex to validate the user queries, and after performing the query validation, then and only then it will parse the query. For Parsing the Queries, I have again used Regex and file operations.After performing parsing, application will make some changes into the text file of the user folder,Also I have created new folder for each user to make code more moduler.I have spend majority of the assignment time in writing Regex queries, it performs in most cases.

Limitations of the Code:

- It only performs and operations, it will not work for any other or operator or combination of and,or operator.
- It only creates one database, I.e one folder for the user, and in that folder it will create text files for the tables.
- User can not delete schema.
- It will only store the data in form of String as there is a limitation of using a text file.
- It will not perform any data type validation from console.

Output:



User Login/Sign Up

```
Welcome fenilpatel
Folder fenilpatel has been created
Write the query
create table Students (name, id);
C:\Users\fenil\OneDrive\Desktop\Data\CSCI_5408_Assignment_1\allTextfiles\
Students

Process finished with exit code 0
```

Create Statement

Note – The output is for sample purposes you can use any CRUD query, Most of the time it will work.

References:

[1] Dalhousie University. [Online]. Available: https://www.dal.ca/. [Accessed: 02-Mar-2023].

[2] Info@erdplus.com, *ERDPlus*. [Online]. Available: https://erdplus.com/. [Accessed: 02-Mar-2023].

Tools Used:

MySQLWorkbench