**HCMC UNIVERSITY OF TECHNOLOGY AND EDUCATION**

**FACULTY FOR HIGH QUALITY TRAINING**

**INFORMATION TECHNOLOGY**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



**THE FIRST PROJECT REPORT**

**BUILDING AN APPLICATION THAT CONVERTING DATA OF THE DATABASE FROM SQL SERVER TO OTHER DATABASE MANAGEMENT SYSTEM**

**LECTURER NAME : Mr. Nguyen Duc Khoan**

**STUDENT NAME : Nguyen Mai Hung**

**STUDENT ID : 16110098**

**STUDENT NAME : Tra Dang Khoa**

**STUDENT ID : 16110124**

**CLASS : 16110CL4**

*Ho Chi Minh City, December 2018*

# Thanks

We would like to express our sincere thanks to Mr. Nguyen Duc Khoan, we are always excited to attend you lecture, because we are sure that we are going to learn again something new and interesting. We found friendship, guidance, discipline, and almost everything in just one person. Whatever success we might have achieved in our project it’s because you have been a wonderful teacher for us. Again, our team would like to thank the lecture.

The project has been completed but there are many limitations. Besides that, there is much knowledge as well as the time we do through each week is not optimal. Mistakes are unavoidable, we are looking forward to receiving all the comments of our teachers to help our limited knowledge better. We sincerely thank.

# Preface

This project report has been prepared in partial fulfilment of the requirement for the subject: the first project in the academic year 2018-2019. For preparing the project report, we have learned more about the issues related to the project during the suggested duration for the period of 8 weeks, to avail the necessary information. The blend of learning and knowledge acquired during our practical studies in the first project is presented in this project report.

The writing report is a partial fulfillment to complete the course. In the report, we try to repersent all the content which we learnt in a great deal in the program in a systematic. We devided each of the topics as an individual chapter to reflect the entire topic more prominently and clearly. In the reference, we have used citation method in the entire report. Finally, we are very hopeful that the structure and topic of the report, will be a useful print material to all the reader especially to the user.

CONTENT

[Thanks 2](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719729)

[Preface 3](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719730)

[1.List of images 5](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719731)

[2.List of tables 5](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719732)

[3.Contents 6](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719733)

[3.1 .Project Description 6](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719734)

[3.1.1. Objective 6](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719735)

[3.1.2. User benefits 6](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719736)

[3.1.3. Use case diagram 7](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719737)

[3.1.4. Use case description tables 8](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719738)

[3.2 .Software degisn 11](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719739)

[3.2.1. Technology 11](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719740)

[3.2.2. Software architecture 12](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719741)

[3.2.3. Programming paradigm 15](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719742)

[3.2.4. Graphical User Interface (GUI) design 18](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719743)

[3.2.5. Class Diagram 20](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719744)

[3.2.6. Data Sources 31](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719745)

[3.2.7. Configuration 31](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719746)

[3.3 .Project implementation 32](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719747)

[3.3.1. Work assignment 32](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719748)

[3.3.2. Work plan 33](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719749)

[3.4 .Manual instruction 34](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719750)

[3.5.Conclusion 35](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719751)

[3.5.1. Results 35](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719752)

[3.5.2. Difficulties 35](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719753)

[3.5.3. Solutions 35](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719754)

[3.5.4. Advantage 36](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719755)

[3.5.5. Defect 36](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719756)

[3.5.6. Source code 36](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719757)

[3.5.7. Development ideas 36](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719758)

[4.Reference 36](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719759)

[LECTURER COMMENT 38](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719760)

# List of images

[Image 1 - Use Case Diagram 7](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529652034)

[Image 2 - software architecture 12](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529652035)

[Image 3 - 3-Layer Architecture 13](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529652036)

[Image 4 - Project architecture 14](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529652041)

[Image 5 - class inheritance and class of data and behavior. 18](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529652038)

[Image 6 - Login Form 19](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529652041)

[Image 7 - Convert Script Form 19](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529652040)

[Image 8 - Class Diagram Presentation Layer 20](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529652040)

[Image 9 - Class Diagram Data Access Layer 21](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529652040)

Image 10 - [Class Diagram Business Layer 21](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529652040)

[Image 11 - Manual instruction 34](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529652053)

# List of tables

[Table 1 - Use case Login description 8](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719657)

[Table 2 - Use case Choice Databases description 8](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719658)

[Table 3 - Use case Choice Tables description 9](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719659)

[Table 4 - Use case Schema, Data or both description 9](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719660)

[Table 5 - Use case Choice DBMS destination description 9](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719661)

[Table 6 - Use case Convert Script description 10](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719662)

[Table 7 - GUI work assignment 18](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719663)

[Table 8 - List of classes are used in the program 22](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719664)

[Table 9 – Methods in ConvertDB 23](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719669)

[Table 10 – Methods in LoginAndGetDataForMain 25](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719669)

[Table 11 - Methods in MSAccess class 25](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719673)

[Table 12 - Methods in MySQL class 26](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719674)

[Table 13 - Methods in ReadMappingFile class 27](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719675)

[Table 14 - Methods in ConectMSAccess class 27](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719676)

[Table 15 - Methods in ConectSQL class 28](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719677)

[Table 16 - Methods in Column class 30](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719678)

[Table 17 - Methods in DataBase class 30](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719679)

[Table 18 - Methods in ForeignKey class 30](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719680)

[Table 19 – Methods in Mapping class 30](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719681)

[Table 20 – Methods in PrimaryKey class 31](file:///D:\MÔN%20HỌC\ĐỒ%20ÁN%201\DAMH_Dot1_Nhom02\DAMH_Dot1_Nhom02\%5bnhom2%5ddocument.docx#_Toc529719682)

Table 21 – Methods in Table class 31

# Contents

# Project Description

### Objective

* Design an application which helps the user to convert schema and data from Microsoft SQL Server to MySQL(.sql) or Access(.mdb).
* Design user interfaces easy to use.
* Features of the database connection.
* Building appropriate object classes.

### User benefits

User can choose many options like:

* Select the database user want to convert.
* Select tables in the database user want.
* Select the DBMS (Database Management System) the user want to convert.
* Choose the path where user save file (\*.sql or \*.mdb).

### Use case diagram



Image 1 - Use Case Diagram

### Use case description tables

Table 1 - Use case Login description

|  |  |  |  |
| --- | --- | --- | --- |
| Use case name | Login | | |
| Description | Allows user to access the database management system | | |
| Actor | User | | |
| Business event | No. | Agent | System response |
|  | 1 | Access the application |  |
|  | 2 |  | Give login interface |
|  | 3 | Check the workstation information |  |
|  | 4 | Click “Sign in” button |  |
| Preconditions | Have installed DBMS Microsoft SQL Server | | |
| Condition affecting termination outcome | When connection succeeded and User click “Sign in” button  When connection failed and show error reporting system | | |

Table 2 - Use case Choice Databases description

|  |  |  |  |
| --- | --- | --- | --- |
| Use case name | Choice Databases | | |
| Description | Select the database according to user requirements | | |
| Actor | User | | |
| Business event | No. | Agent | System response |
|  | 1 |  | Load the database name for the user select |
|  | 2 | Select 1 database to generate the script |
| Preconditions | Successfully connect to DBMS | | |
| Condition affecting termination outcome | Function performed successfully  User click “Exit” | | |

Table 3 - Use case Choice Tables description

|  |  |  |  |
| --- | --- | --- | --- |
| Use case name | Choice Tables. | | |
| Description | Support for users can choose a few or all tables | | |
| Actor | User | | |
| Business event | No. | Agent | System response |
|  | 1 |  | Load the tables name for the user select |
|  | 2 |  | Display the list of options |
|  | 3 | Select tables user need to convert script |  |
| Preconditions | Successfully connect to DBMS | | |
| Condition affecting termination outcome | User has selected the object to generate the script  User click “Exit” | | |

Table 4 - Use case Schema, Data or both description

|  |  |  |  |
| --- | --- | --- | --- |
| Use case name | Schema, Data or both. | | |
| Description | Support for users can choose types of data as desired | | |
| Actor | User | | |
| Business event | No. | Agent | System response |
|  | 1 | Logged in successfully |  |
|  | 2 |  | Display the list of types of data to script |
|  | 3 | Select a types of data to convert script |  |
| Preconditions | User logged in successfully | | |
| Condition affecting termination outcome | Function performed successfully  User click “Exit” | | |

Table 5 - Use case Choice DBMS destination description

|  |  |  |  |
| --- | --- | --- | --- |
| Use case name | Choice DBMS destination | | |
| Description | Support for user can choose several Tables in a database to generate scripts as desired | | |
| Actor | User | | |
| Business event | No. | Agent | System response |
|  | 1 | Logged in successfully |  |
|  | 2 |  | Display the list of types DBMS destination user want convert |
|  | 3 | Select types DBMS destination user want convert script |  |
|  | 4 |  | Get the selections of the user for the resolution |
| Preconditions | User logged in successfully | | |
| Condition affecting termination outcome | Function performed successfully  User click “Exit” | | |

Table 6 - Use case Convert Script description

|  |  |  |  |
| --- | --- | --- | --- |
| Use case name | Export Cript | | |
| Description | Support for user can save file scripts as \*.sql file or \*.accdb | | |
| Actor | User | | |
| Business event | No. | Agent | System response |
|  | 1 | Logged in successfully |  |
|  | 2 |  | Display 1 options as CheckBox Control |
|  | 4 | Create file scripts as \*.sql file or \*.accdb on user requests. |  |
| Preconditions | User logged in successfully | | |
| Condition affecting termination outcome | Function performed successfully  User click “Exit” | | |

## Software degisn

### Technology

In project we used ADO.NET technology as the main technology for our our project.

ADO.NET provides consistent access to data sources such as SQL Server and XML, and to data sources exposed through OLE DB and ODBC. Data-sharing consumer applications can use ADO.NET to connect to these data sources and retrieve, handle, and update the data that they contain.

ADO.NET separates data access from data manipulation into discrete components that can be used separately or in tandem. ADO.NET includes .NET Framework data providers for connecting to a database, executing commands, and retrieving results. Those results are either processed directly, placed in an ADO.NET [DataSet](https://docs.microsoft.com/en-us/dotnet/api/system.data.dataset) object in order to be exposed to the user in an ad hoc manner, combined with data from multiple sources, or passed between tiers. The DataSet object can also be used independently of a .NET Framework data provider to manage data local to the application or sourced from XML.

The ADO.NET classes are found in System.Data.dll, and are integrated with the XML classes found in System.Xml.dll. For sample code that connects to a database, retrieves data from it, and then displays that data in a console window, see [ADO.NET Code Examples](https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ado-net-code-examples).

ADO.NET provides functionality to developers who write managed code similar to the functionality provided to native component object model (COM) developers by ActiveX Data Objects (ADO). We recommend that you use ADO.NET, not ADO, for accessing data in your .NET applications.

ADO.NET provides the most direct method of data access within the .NET Framework. For a higher-level abstraction that allows applications to work against a conceptual model instead of the underlying storage model, see the [ADO.NET Entity Framework](https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/index).

### Software architecture

We use software architecture is three-layer architecture to interact with UI (user interface), functional process logic (BR-business rule or BL-business logic), computer data storage and data access. The three-layer architecture is a software design pattern and well-established software architecture.

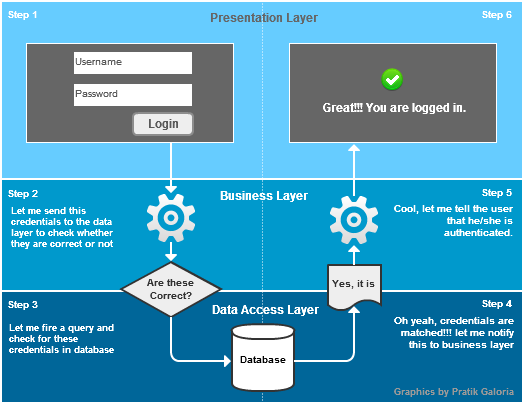
[](https://www.google.com.vn/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwjBmZa018PeAhXKc94KHaOKDGkQjRx6BAgBEAU&url=https://www.codeproject.com/Articles/667431/3-Tier-Architecture-in-Csharp-Web-Application&psig=AOvVaw0WCUSv6rgB_J3LKQpJJ9vb&ust=1541728153754008)

Image 2 - software architecture

Advantages of 3-layer Architecture

* Reusability: Moving code-behind libraries. It is possible to make changes in the presentation level without effecting the other two (business or data access layer).
* Decoupling of UI, Business logic and Data Access Layer are done.
* Maintainability: When we change one layer due to the client's needs it doesn't affect the other layers and we need to do less changes in another layer.

Our project apply 3-layer architecture indicates logical separation of components, such as having distinct classes for the Database Access Layer, Business Logic Layer and User Interface Layer.

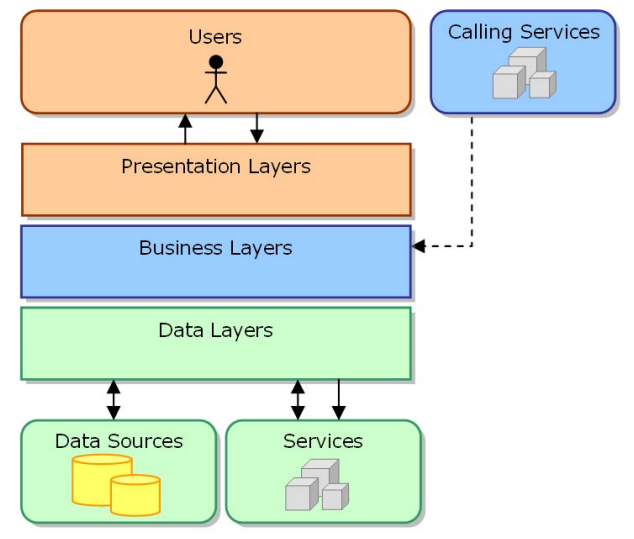


Image 3 - 3-Layer Architecture

It has a few rule in the 3-layer architecture:

The code for each layer must be contained with separate files which can be maintained separately.

Each layer may only contain code which belongs in that layer. Thus business logic can only reside in the Business layer, presentation logic in the Presentation layer, and data access logic in the Data Access layer.

Each layer should be totally unaware of the inner workings of the other layers. The Business layer, for example, must be database-agnostic and not know or care about the inner workings of the Data Access layer. It must also be presentation-agnostic and not know or care how its data will be handled. It should not process its data differently based on what the receiving component will do with that data. The presentation layer may take the data (in our case – treeview) and construct a user interface, or process it in some other way, but that should be totally irrelevant to the Business layer.

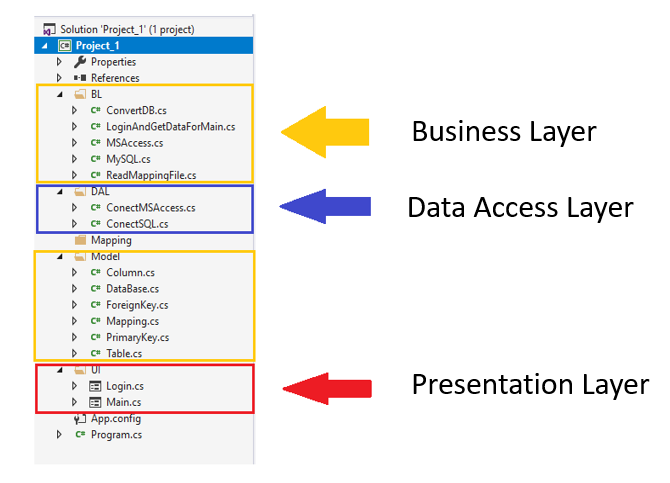


Image 4 – Project architecture

* The Presentation layer can only receive requests from Form, and return responses to form.
* The Presentation layer can only send requests to, and receive responses from the Business layer. It cannot have direct access to either the database or the Data Access layer.
* The Business layer can only receive requests from, and return response to the Presentation layer.
* The Business layer can only send requests to, and receive responses from the Data Access layer. It cannot access the database directly.
* The Data Access layer can only receive requests from, and return responses to the Business layer. It cannot issue requests to anything other than the DBMS which it supports.

### Programming paradigm

After realizing the advantage of Object Oriented Programming paradigm, our team decided to design the application in that paradigm.

Historically, in the early days, computer programmers executed commands like this:

function main()

{

// All code is here

}

Then, in the next evolutionary step, people split the code into more specialized parts, each of which performs a function, and they are retrieved from the main function:

function main()

{

a();

b();

}

function a()

{

// Your code here

}

function b()

{

// Your code here

}

And it is called “Procedure Oriented Programming – POP”

And in the next evolutionary stage, in the 2000s, people were programming Object Oriented Programming (OOP).

Principles of object-oriented programming:

* class inheritance.
* abstraction of data and behavior.
* encapsulation of data and class implementation.
* polymorphism methods.
* virtual methods.

Object-oriented programming (OOP) is a programming paradigm based on the concept of "*objects*", which may contain data, in the form of fields, often known as *attributes;* and code, in the form of procedures, often known as *methods*.

Classes – the definitions for the data format and available procedures for a given type or class of object; may also contain data and procedures (known as class methods) themselves, i.e. classes contain the data members and member functions

Objects – instances of classes

Object Oriented Programming has great advantages over other programming styles:

* Code Reuse and Recycling:
* Objects created for Object Oriented Programs can easily be reused in other programs.
* Encapsulation:
* Once an Object is created, knowledge of its implementation is not necessary for its use. In older programs, coders needed understand the details of a piece of code before using it (in this or another program).
* Objects have the ability to hide certain parts of themselves from programmers. This prevents programmers from tampering with values they shouldn't. Additionally, the object controls how one interacts with it, preventing other kinds of errors.
* Design Benefits:
* Large programs are very difficult to write. Object Oriented Programs force designers to go through an extensive planning phase, which makes for better designs with less flaws. In addition, once a program reaches a certain size, Object Oriented Programs are actually easier to program than non-Object Oriented ones.
* Software Maintenance:
* Programs are not disposable. Legacy code must be dealt with on a daily basis, either to be improved upon (for a new version of an exist piece of software) or made to work with newer computers and software. An Object Oriented Program is much easier to modify and maintain than a non-Object Oriented Program. So although a lot of work is spent before the program is written, less work is needed to maintain it over time.

In our project, we apply OOP coding style in many ways.

Since all of the columns, tables, and databases have the same “Name” attribute and the “CreateScript” method, we design relationships between these classes like this:

public class MySQL:ConvertDB{…}

public class MSAccess:ConvertDB{…}

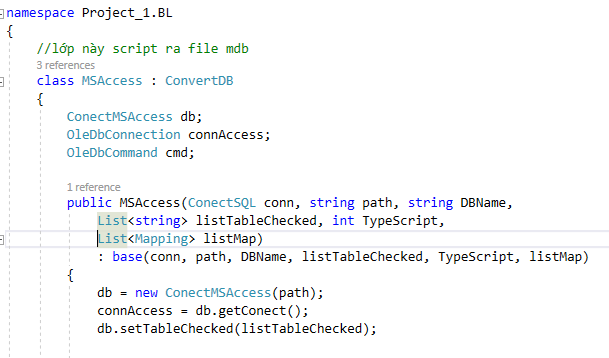


Image 5 - class inheritance and class of data and behavior.

### Graphical User Interface (GUI) design

As all applications are used, the interface is one of the key success factors of a program.

|  |  |  |  |
| --- | --- | --- | --- |
| No | Forms | Designer &Briefly explain design decisions | Purpose |
| 1 | Login | Nguyen Mai Hung  The login screen is familiar and simple. | Users can connect to any available database corresponding to the login account in the computer. |
| 2 | Convert Script | Nguyen Mai Hung and Tra Dang Khoa  The design is satisfying and simple. | Users can select multiple options to create the script they want to convert. They can also choose the table and where they want to save, in the end, they can select the DBMS they want to convert. |

Table 7 – GUI work assignment

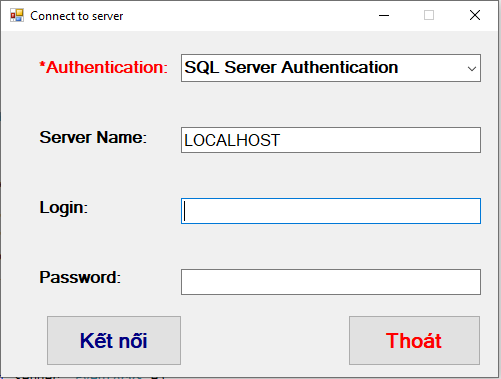
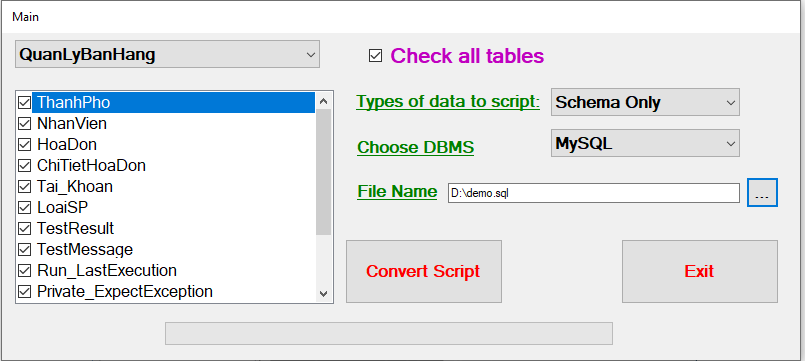


Image 6 - Login Form

Image 7 - Convert Script Form

### Class Diagram

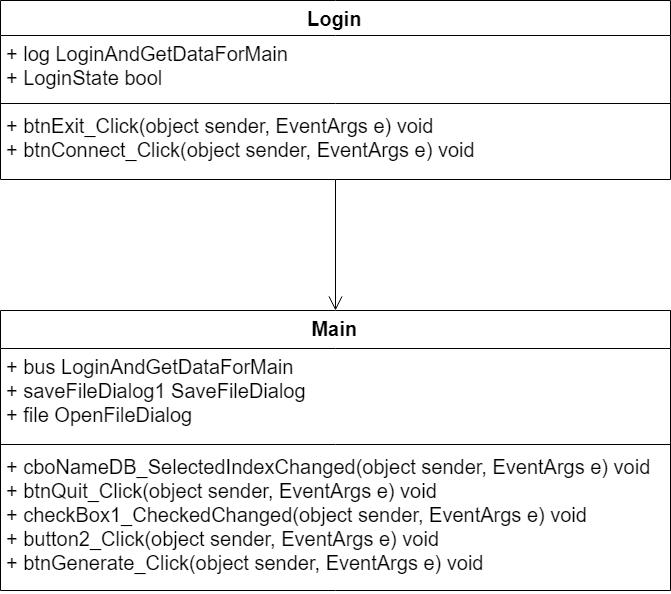


Image 8 – Class Diagram Presentation Layer

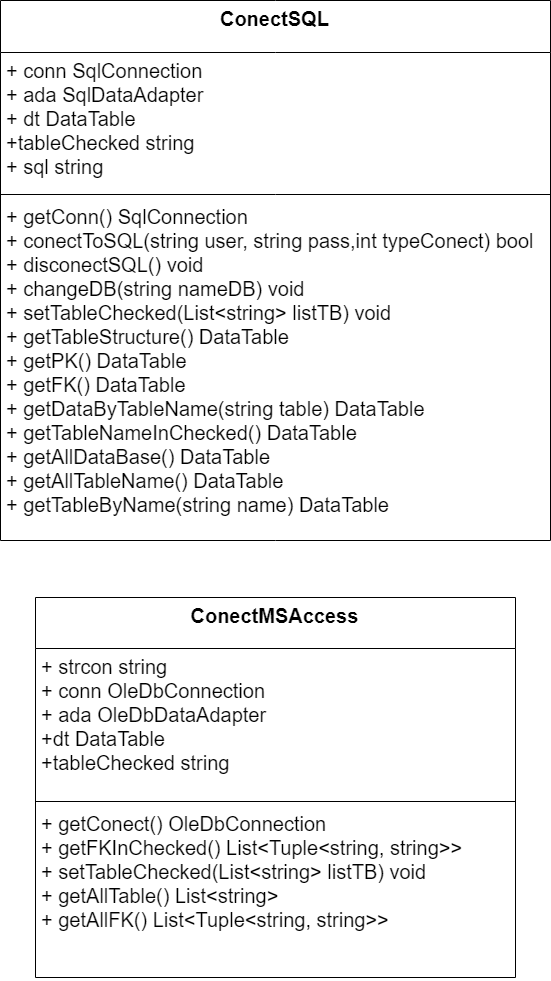


Image 9 – Class Diagram Data Access Layer

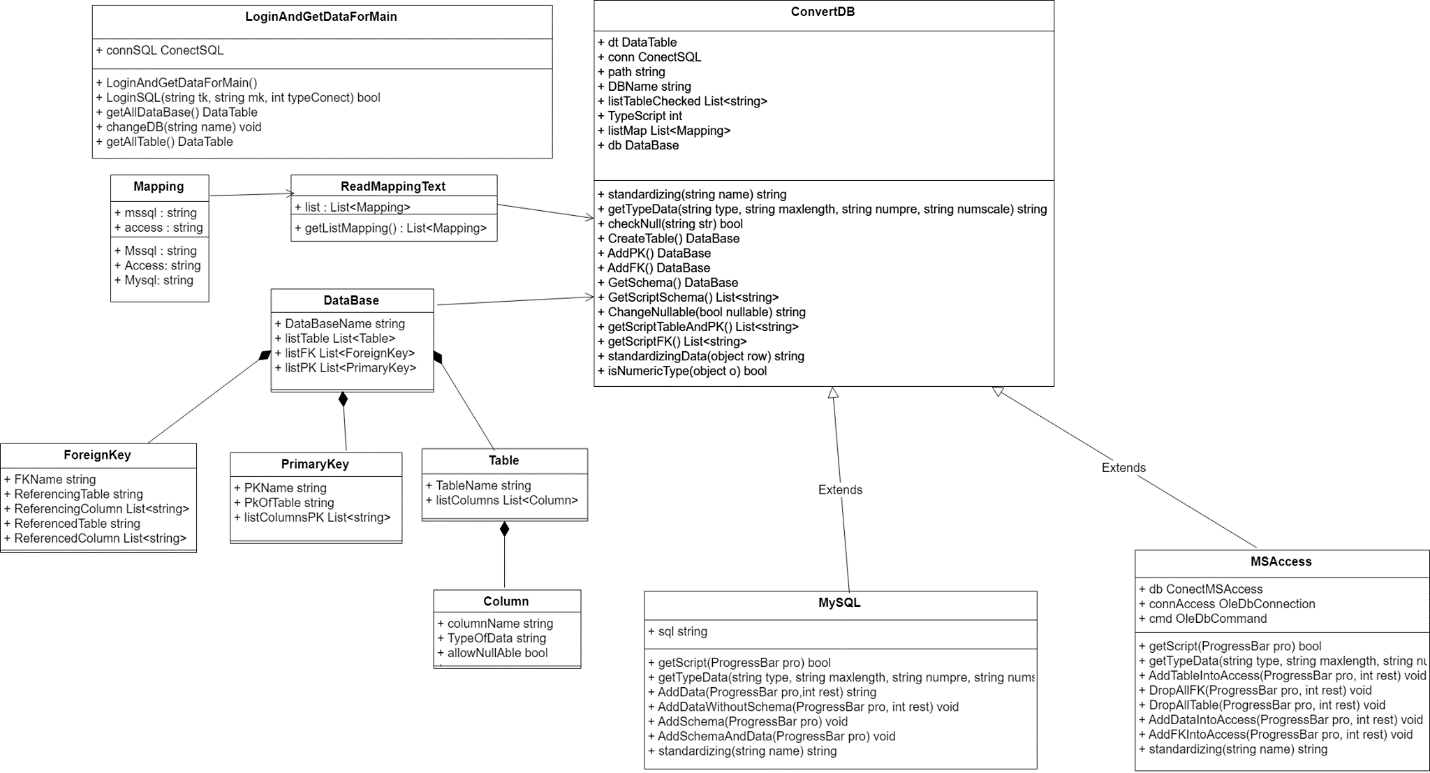


Image 10 – Class Diagram Business Layer

#### List of classes are used in the program

Table 8 - List of classes are used in the program

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Class Name | Responsible | Purpose |
| 1 | ConvertDB | Tra Dang Khoa | An extensible program-code-template for convert objects (get initialization, get primary key, foreign key, constraint,.....). |
| 2 | LoginAndGetDataForMain | Tra Dang Khoa | Takes all data to the user interface and check login. |
| 3 | MS Access | Tra Dang Khoa | Inherit ConvertDB Class.  Add data to database access and convert the data and structure through the ConvertDB class into data and the structure of the MS Access. |
| 4 | MySQL | Tra Dang Khoa, Nguyen Mai Hung | Inherit ConvertDB Class.  Add data to database access and convert the data and structure through the ConvertDB class into data and the structure of the MySQL. |
| 5 | ReadMappingFile | Tra Dang Khoa | Reads file mapping to used to convert data types. |
| 6 | ConectMSAccess | Tra Dang Khoa | This is a data access layer used to connect to MS Access.  Provides simplified access to data stored. Return objects containing the requested values. |
| 7 | ConectSQL | Nguyen Mai Hung | This is a data access layer used to connect to SQL Server. |
| 8 | Column | Nguyen Mai Hung | This is the layer that contains the structure of a column |
| 9 | DataBase | Nguyen Mai Hung | This is the layer that contains the structure of a database |
| 10 | ForeignKey | Nguyen Mai Hung | This is the layer that holds the structure of the references foreign key |
| 11 | Mapping | Tra Dang Khoa, Nguyen Mai Hung | This class contains the structure to convert the corresponding data type between the three DBMS |
| 12 | PrimaryKey | Nguyen Mai Hung | This is the layer that holds the structure of the references primary key |
| 13 | Table | Nguyen Mai Hung | This class contains the structure of a table |

#### Methods in ConvertDB class

Table 9 - Methods in ConvertDB

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Method | Purpose | File name,  Line | Responsible |
| 1 | **ConvertDB()**  Input: conn, path, DBName, ListTableChecked, TypeScript,listMap)  Output: none.  Pseudo code: none. | Initialize Column object with specify arguments – its attributes | ConvertDB.cs  (73) | Tra Dang Khoa |
| 2 | **Standardizing()**  Input: name  Output: string.  Pseudo code: none. | Standard functions for table and column names | ConvertDB.cs  (85) | Tra Dang Khoa |
| 3 | **getTypeData()**  Input: type, maxlength, numpre, numscale  Output: string.  Pseudo code: none. | Get data type with both data length | ConvertDB.cs  (89) | Tra Dang Khoa |
| 4 | **checkNull()**  Input: str  Output: booleans.  Pseudo code: none. | Check null returns | ConvertDB.cs  (95) | Tra Dang Khoa |
| 5 | **CreateTable()**  Input: str  Output: database.  Pseudo code: none. | Mapping the table structure into the object | ConvertDB.cs  (100) | Tra Dang Khoa |
| 6 | **AddPK()**  Input: none  Output: database.  Pseudo code: none. | Map the primary key to the object | ConvertDB.cs  (133) | Tra Dang Khoa |
| 7 | **AddFK()**  Input: none  Output: database.  Pseudo code: none. | Map foreign key to object | ConvertDB.cs  (164) | Tra Dang Khoa |
| 8 | **GetSchema()**  Input: none  Output: database.  Pseudo code: none. | The method used to gather methods CreateTable(),CreateTable(),CreateTable(). | ConvertDB.cs  (218) | Tra Dang Khoa |
| 9 | **GetScriptSchema()**  Input: none  Output: listResult.  Pseudo code: none. | Convert schema to string format including table name, column, primary key, foreign key | ConvertDB.cs  (227) | Tra Dang Khoa |
| 10 | **getScriptTableAndPK()** Input: none  Output: listResult.  Pseudo code: none. | Convert table and PK to string format including table name, column, primary key, foreign key | ConvertDB.cs  (291) | Tra Dang Khoa |
| 11 | **getScriptFK()**  Input: none  Output: listResult.  Pseudo code: none. | Convert FK to string format. | ConvertDB.cs  (328) | Tra Dang Khoa |
| 12 | **standardizingData()**  Input: row  Output: string.  Pseudo code: none. | Standardizing Data | ConvertDB.cs  (355) | Tra Dang Khoa |
| 13 | **isNumericType()**  Input: object  Output: booleans.  Pseudo code: none. | Returns the data type of the object | ConvertDB.cs  (414) | Tra Dang Khoa |

#### Methods in LoginAndGetDataForMain class

Table 10 - Methods in LoginAndGetDataForMain

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Method | Purpose | File name, Line | Responsible |
| 1 | **LoginAndGetDataForMain()**  Input: none.  Output: none.  Pseudo code: none. | connecting to SQL Sever | LoginAndGetDataForMain.cs (23) | Tra Dang Khoa |
| 2 | **LoginSQL()**  Input: tk, mk, typeConect.  Output: booleans.  Pseudo code: none. | Check login | LoginAndGetDataForMain.cs (28) | Tra Dang Khoa |
| 3 | **getAllDataBase(), getAllTable()**  Input: none.  Output: datatable  Pseudo code: none. | Get all table names | LoginAndGetDataForMain.cs (41,50) | Tra Dang Khoa |

#### Methods in MSAccess class

Table 11 - Methods in MSAccess

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Method | Purpose | File name, Line | Responsible |
| 1 | **MSAccess()**  Input: conn, path, DBName, listTableChecked,  TypeScript, listMap)  Output: none.  Pseudo code: none. | Initialize a MS Access object without argument. | MSAccess.cs  (22) | Tra Dang Khoa |
| 2 | **getScript()**  Input: pro.  Output: boolean.  Pseudo code: none. | Pour data into MS Access | MSAccess.cs  (34) | Tra Dang Khoa |
| 3 | **getTypeData()**  Input: type, maxlength, numpre, numscale.  Output: datatype.  Pseudo code: none. | Override only for access | MSAccess.cs  (83) | Tra Dang Khoa |
| 4 | **DropAllFK()**  Input: pro, rest.  Output: none.  Pseudo code: none. | Get list FK in MS Access. | MSAccess.cs  (139) | Tra Dang Khoa |
| 5 | **AddDataIntoAccess()**  Input: pro, rest.  Output: none.  Pseudo code: none. | Add data into access | MSAccess.cs  (177) | Tra Dang Khoa |
| 6 | **AddFKIntoAccess()**  Input: pro, rest.  Output: none.  Pseudo code: none. | Add FK into access | MSAccess.cs  (211) | Tra Dang Khoa |
| 7 | **standardizing()**  Input: name.  Output: none.  Pseudo code: none. | Standardizing name | MSAccess.cs  (228) | Tra Dang Khoa |

#### Methods in MySQL class

Table 12 - Methods in MySQL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Method | Purpose | File name,  Line | Responsible |
| 1 | **MySQL()**  Input: conn, path, DBName, listTableChecked, TypeScript, listMap.  Output: none.  Pseudo code: none. | Initialize a MySQL object without argument. | MySQL.cs  (17) | Nguyen Mai Hung |
| 2 | **getScript()**  Input: pro.  Output: boolean.  Pseudo code: none. | Pour data into MS Access | MySQL.cs  (22) | Nguyen Mai Hung |
| 3 | **getTypeData()**  Input: type, maxlength, numpre, numscale.  Output: datatype.  Pseudo code: none. | Override only for access | MySQL.cs  (48) | Nguyen Mai Hung |
| 4 | **AddData()**  Input: pro, rest.  Output: sql.  Pseudo code: none. | Just add data | MySQL.cs  (81) | Nguyen Mai Hung |
| 5 | **AddDataWithoutSchema()**  Input: pro, rest.  Output: sql.  Pseudo code: none. | Add data without schema | MySQL.cs  (127) | Nguyen Mai Hung |
| 6 | **AddSchema()**  Input: pro.  Output: sql.  Pseudo code: none. | Just add Schema | MySQL.cs  (137) | Nguyen Mai Hung |
| 7 | **AddSchemaAndData()**  Input: pro.  Output: sql.  Pseudo code: none. | Add data schema | MySQL.cs  (155) | Nguyen Mai Hung |
| 8 | **standardizing()**  Input: name.  Output: none.  Pseudo code: none. | Standardizing name | MySQL.cs  (176) | Nguyen Mai Hung |

#### Methods in ReadMappingFile class

Table 13 - Methods in ReadMappingFile

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Method | Purpose | File name, Line | Responsible |
| 1 | **getListMapping()**  Input: none  Output: list  Pseudo code: none | Get list in file mapping | ReadMappingFile.cs  (20) | Tra Dang Khoa |

#### Methods in ConectMSAccess class

Table 14 - Methods in ConectMSAccess

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Method | Purpose | File name, Line | Responsible |
| 1 | **ConectMSAccess()**  Input: datasource  Output: none  Pseudo code: none | Connect function to MS Access | ConectMSAccess.cs  (18) | Tra Dang Khoa |
| 2 | **getConect()**  Input: none  Output: none  Pseudo code: none | Connect function to MS Access via OLEDB | ConectMSAccess.cs  (26) | Tra Dang Khoa |
| 3 | **getFKInChecked()** Input: none  Output: DataTable  Pseudo code: none | Get FK in checked | ConectMSAccess.cs  (30) | Tra Dang Khoa |
| 4 | **setTableChecked()**  Input:listTB  Output: DataTable  Pseudo code: none | Set table checked | ConectMSAccess.cs  (48) | Tra Dang Khoa |
| 5 | **getAllTable()**  Input: none  Output: none  Pseudo code: none | Get all table | ConectMSAccess.cs  (57) | Tra Dang Khoa |
| 6 | **getAllFK()**  Input: none  Output: List<string>  Pseudo code: none | Get all FK | BLData.cs  (112) | Nguyen Luong My Ha |

#### Methods in ConectSQL class

Table 15 - Methods in ConectSQL class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Method | Purpose | File name,  Line | Responsible |
| 1 | **getConn()**  Input: none  Output: none  Pseudo code: none | Connect function to SQL Server | ConectSQL.cs  (22) | Nguyen Mai Hung |
| 2 | **conectToSQL()**  Input: user, pass, typeConect.  Output: none.  Pseudo code: none | Function check connection SQL Server | ConectSQL.cs  (26) | Nguyen Mai Hung |
| 3 | **disconectSQL()**  Input:none.  Output: none.  Pseudo code: none | Function disconect SQL Server | ConectSQL.cs  (55) | Nguyen Mai Hung |
| 4 | **changeDB()**  Input: nameDB.  Output: none.  Pseudo code: none. | Function change name database | ConectSQL.cs  (60) | Nguyen Mai Hung |
| 5 | **setTableChecked()**  Input:listTB.  Output: none.  Pseudo code: none | Function use to set table user check | ConectSQL.cs  (65) | Nguyen Mai Hung |
| 6 | **getTableStructure()**  Input:none.  Output: none.  Pseudo code: none | Function use to get table structure | ConectSQL.cs  (74) | Nguyen Mai Hung |
| 7 | **getPK()**  Input:none.  Output: datatable.  Pseudo code: none | Function use to get PK structure | ConectSQL.cs  (86) | Nguyen Mai Hung |
| 8 | **getFK()**  Input:none.  Output: datatable.  Pseudo code: none | Function use to get FK structure | ConectSQL.cs  (99) | Nguyen Mai Hung |
| 9 | **getDataByTableName()**  Input: table.  Output: datatable.  Pseudo code: none | Function use to get data by table name | ConectSQL.cs  (117) | Nguyen Mai Hung |
| 10 | **getTableNameInChecked()**  Input:none.  Output: datatable.  Pseudo code: none | Function use to get table name in checked | ConectSQL.cs  (128) | Nguyen Mai Hung |
| 11 | **getAllDataBase()**  Input:none.  Output: datatable.  Pseudo code: none | Function use to get all database | ConectSQL.cs  (137) | Nguyen Mai Hung |
| 12 | **getAllTableName()**  Input:none.  Output: datatable.  Pseudo code: none | Function use to get all table name | ConectSQL.cs  (145) | Nguyen Mai Hung |
| 13 | **getTableByName()**  Input:name.  Output: datatable.  Pseudo code: none | Function use to get table by name | ConectSQL.cs  (153) | Nguyen Mai Hung |

#### Methods in Column class

Table 16 - Methods in Column class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Method | Purpose | File name,  Line | Responsible |
| 1 | **Column()**  Input: name, TypeOfData, AllowNullAble  Output: none  Pseudo code: none | Initialize a Column object. | Column.cs  (34) | Nguyen Mai Hung |

#### Methods in DataBase class

Table 17 - Methods in DataBase class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Method | Purpose | File name,  Line | Responsible |
| 1 | **DataBase()**  Input: none  Output: none  Pseudo code: none | Initialize a Database object. | DataBase.cs  (42) | Nguyen Mai Hung |

#### Methods in ForeignKey class

Table 18 - Methods in ForeignKey class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Method | Purpose | File name,  Line | Responsible |
| 1 | **ForeignKey()**  Input: none  Output: none  Pseudo code: none | Initialize a ForeignKey object. | ForeignKey.cs  (46) | Nguyen Mai Hung |

#### Methods in Mapping class

Table 19 - Methods in Mapping class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Method | Purpose | File name,  Line | Responsible |
| 1 | **Mapping()**  Input: string a,b,c  Output: none  Pseudo code: none | Mapping between data types | Mapping.cs  (15) | Nguyen Mai Hung |

#### Methods in PrimaryKey class

Table 20 - Methods in PrimaryKey class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Method | Purpose | File name,  Line | Responsible |
| 1 | **PrimaryKey()**  Input: none  Output: none  Pseudo code: none | Initialize a list Primary Key | PrimaryKey.cs  (34) | Nguyen Mai Hung |

#### Methods in Table class

Table 21 - Methods in Table class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Method | Purpose | File name,  Line | Responsible |
| 1 | **Table()**  Input: none  Output: none  Pseudo code: none | Initialize a table | Table.cs  (27) | Nguyen Mai Hung |

### Data Sources

* All available databases in Microsoft SQL server.
* SQL Server DBMS
* MySQL DBMS
* MS Access DBMS

### Configuration

* Recommend Database Management System MSQL Server in your computer.
* Make sure SQL Server Services is started.
* Make sure MS Accsess is installed

## Project implementation

### 3.3.1 Work assignment

Table 22 – Work plan

|  |  |  |
| --- | --- | --- |
| Student’s name | Evaluate contribution | Taskwork |
| Nguyen Mai Hung | 100% | Design UI and Use Case Diagram |
| Tra Dang Khoa | 100% | Design Class Diagram |
| Tra Dang Khoa,  Nguyen Mai Hung | 100% | Coding |
| Tra Dang Khoa,  Nguyen Mai Hung | 100% | Test app and Debug |

### 3.3.2 Work plan

Table 23 – Work assignment

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Build application to export SQL Server Database to SQL Script | | | | | | | | | | | |
| No. | Goal | Schedule | | | | | | | | Khoa | Hung |
| 1 | Understand Requirement. | o | o |  |  |  |  |  |  | 🗸 | 🗸 |
| 2 | Describe the requirements of the project. | o | o |  |  |  |  |  |  | 🗸 | 🗸 |
| 3 | Use case diagram |  | o | o |  |  |  |  |  |  | 🗸 |
| 4 | Learn About Schema and ADO.NET. |  | o | o | o |  |  |  |  | 🗸 | 🗸 |
| 5 | Three Layer Architecture in C# .NET. |  | o | o | o |  |  |  |  | 🗸 |  |
| 6 | Review Object Oriented Programming |  |  | o | o |  |  |  |  | 🗸 | 🗸 |
| 7 | User interface design. |  |  | o | o | o |  |  |  |  | 🗸 |
| 8 | Logical algorithm design (algorithm flowchart) |  |  |  | o | o |  |  |  | 🗸 | 🗸 |
| 9 | Load Database from MS SQL SERVER to C# Winform Application. |  |  |  | o | o |  |  |  |  | 🗸 |
| 10 | Learn how to Convert Script. |  |  |  | o | o |  |  |  | 🗸 | 🗸 |
| 11 | Class Diagram Design. |  |  |  | o | o | o |  |  | 🗸 |  |
| 12 | Raise Events for Components. |  |  |  |  | o |  |  |  | 🗸 | 🗸 |
| 13 | Program Implementation. |  |  |  |  | o | o | o |  | 🗸 | 🗸 |
| 14 | Optimize Code. |  |  |  |  |  |  | o |  | 🗸 | 🗸 |
| 15 | Testing. |  |  |  |  |  |  | o |  | 🗸 | 🗸 |
| 16 | Report. |  |  |  |  |  |  | o | o | 🗸 | 🗸 |
| Day | | 2/11/2018 | 9/11/2018 | 16/11/2018 | 23/11/2018 | 30/11/2018 | 7/12/2018 | 14/12/2018 | 21/12/2018 |  |  |
| Week | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  |  |
| Note | | o – Begin  o – Complete 50%  o – Complete 100% | | | | | | | | | |

## Manual instruction

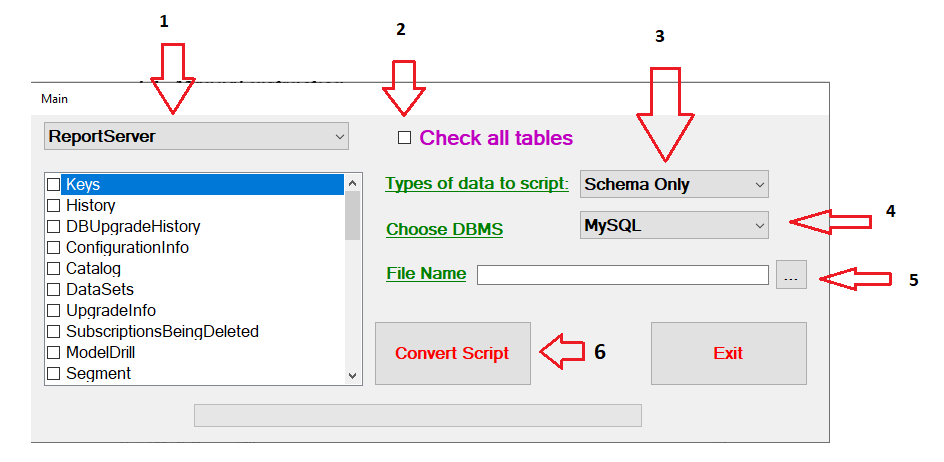


Image 11 - Manual instruction

Step 1: Select a database, some tables in a database you want to convert the script.

Step 2: Select all tables in a database you want to convert the script.

Step 3: Select type of data to script you want to convert (three options support is Schema Only, Schema And Data, Data Only ).

Step 4: Select DBMS destination you want to convert (tow options support is MySQL and Access ).

Step 5: Select location to save the folder when you convert.

Step 6: Click Convert Button (save the script as file \*.sql with MySQL and \*.accdb with Access).

## Conclusion

### Results

* Almost requirements are met.
* Apply three-layer architecture is software architecture.
* Design the application with Object Oriented Programming paradigm.
* Simple design GUI for easy using.
* The code is quite clean and reusable.
* Our application can have get relationship between tables (SQL Foreign key Constraint).

### Difficulties

* Initially, we did not understand about access so it took time to implement the program.
* There are many new concept that we have never known before (Mapping between DBMS and OLEDB in MS Access)
* It is difficult to combine the knowledge learned from previous semesters into the same research topic and apply them proficiency.

### Solutions

* Review the knowledge learned from the previous semesters about databases, data structures and algorithms, OOP, Windows programming, programming techniques,…
* Refer to a variety of sources in books and internet to answer the issues involved.
* Ask lecturer to get specific instruction to solve the problem.

### Advantage

* Converts most types of data
* Meets the requirements of the project
* Simple and user-friendly GUI
* Reuse and Recycling, Maintainability
* The application size is small.

### Defect

* Users are required to install and start services Microsoft SQL Server.
* Our application needs to create a database in \* .accdb format before converting the data

### Source code

<https://github.com/khoatd152/Project_1/tree/master/Project_1>

### Development ideas

* Support for converting databases directly from computers that have installed SQL Server DBMSs in the same server.

# Reference

* Nguyen Duc Khoan, ADO.NET Tutorial part 1

<https://www.youtube.com/watch?v=sDhuKpcoepk>, accessed 15th November, 2018

* Nguyen Duc Khoan, ADO.NET Tutorial part 2

<https://www.youtube.com/watch?v=QSwr98p8ohk>, accessed 18th November, 2018

* StackOverflow, show the table structure in sql server,

<https://stackoverflow.com/questions/18298433/how-can-i-show-the-table-structure-in-sql-server-query>

* StackOverflow, get table primary key using sql,

<https://stackoverflow.com/questions/3930338/sql-server-get-table-primary-key-using-sql-query>

* StackOverflow, list all foreign keys referencing a given table table in sql server,

<https://stackoverflow.com/questions/483193/how-can-i-list-all-foreign-keys-referencing-a-given-table-in-sql-server>

* StackOverflow, connect access database in C#,

<https://stackoverflow.com/questions/17023861/how-to-connect-access-database-in-c-sharp>

* StackOverflow, connect database sql server use C#,

<https://o7planning.org/vi/10511/ket-noi-database-sql-server-su-dung-csharp>

# LECTURER COMMENT

Lecturer

(Name, signature)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_