Principles of Database Management

Department of Computer Science and Engineering

International University – Vietnam National University

**Forum Management Database System**

Nguyen Tien Cuong - ITITIU18172

Nguyen Le Nguyen - ITITIU18094 (leader)

Nguyen Van Tri - ITITUN18051



12-05-2020

|  |  |  |
| --- | --- | --- |
| **Student name** | **Student ID** | **Contributions** |
| Nguyen Tien Cuong | ITITIU18172 | Be in charge of ER diagram, writing report, peer review |
| Nguyen Le Nguyen | ITITIU18094 | Create database file, review the ER diagram, create query GUI application |
| Nguyen Van Tri | ITITUN18051 | Create relational schema, support in creating the GUI, make presentation slide |

Contents

[INTRODUCTION 4](#_Toc39835214)

[HOW WE CARRY OUT THE PROJECT 4](#_Toc39835215)

[PROJECT DEMO 4](#_Toc39835216)

[CONCLUSIONS 4](#_Toc39835217)

# INTRODUCTION

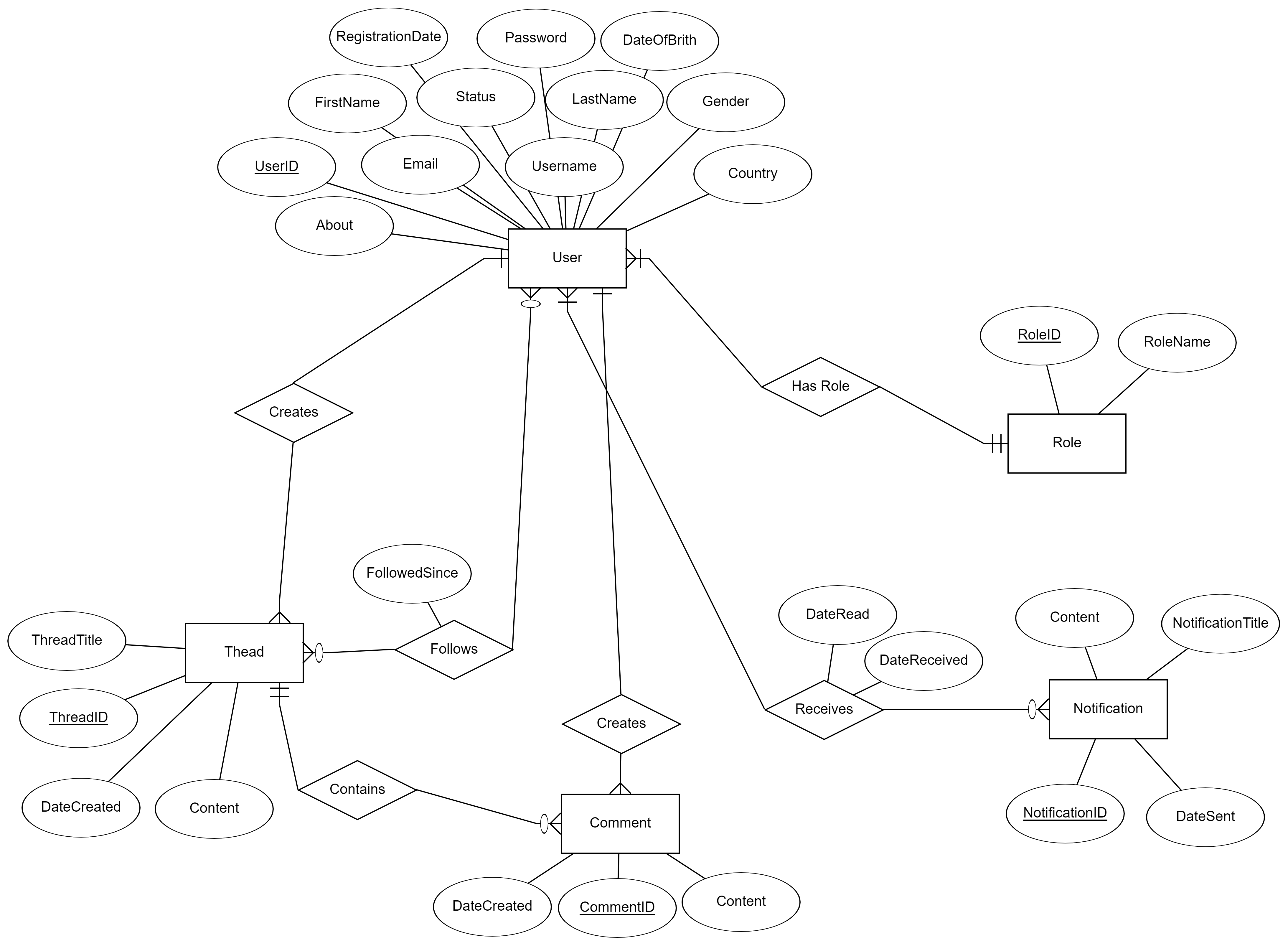
With the explosion of the modern Internet, there are more and more types of social networks is being developed. Social networks are essentially an online website that people can access and connect to many other people including friends, family members, business partner, and so on. One type of social network that is very popular is forums. Forums are the place where people can discuss ideas and views on a particular topic such as machine learning forums, information technology forums, hacking forums, mathematics community forums, etc. Forums have now become an indispensable part for people nowadays as it gives access to knowledge and important information. Also, it connects many people with the same interest together.

In this project, we attempt to develop a small database system for managing such forum at a basic level. In this report, the process in which we analyze and develop the database system is presented in Section 2, a demo is given in Section 3, and the last section, i.e. Section 4, gives a brief conclusion for this report.

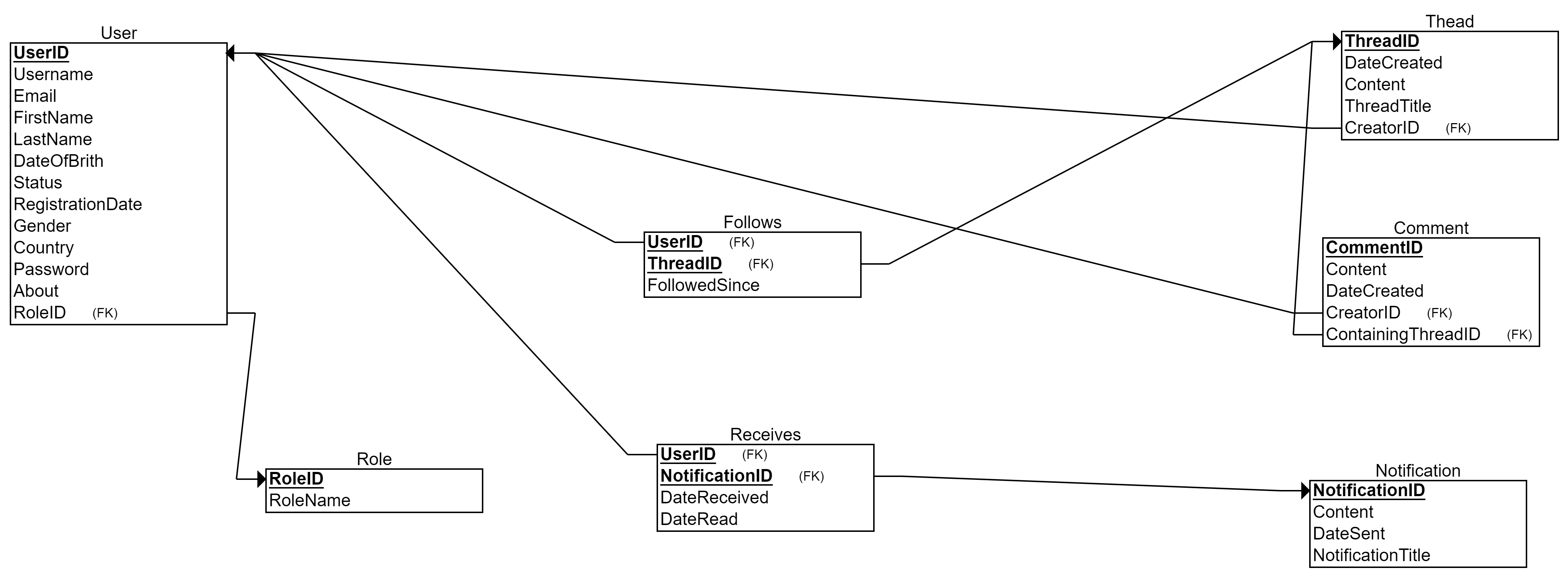
# HOW WE CARRY OUT THE PROJECT

In this project, we have done the following steps:

* In the first stage, we have created the ER diagram, after a period to normalize the database into 3NF, we have finally come up with the following ER diagram:



* + The **User** table captures the essential information of a user. It contains basic information for other user to know about a specific user such as names, gender, the homeland of the user, email, status, etc. Also, the *Status* attribute is to specify the activeness of the user, i.e. to know if that user is banned from the forum, or simply the account was deactivated. The *Password* attribute will store the hashed password of the user to increase the security robustness of our database. We also let the User has the **UserID** as primary key to speed up the querying process in applications which utilize this database.
  + Other tables also contains an ID to enable faster query so that the applications can answer the query as quick as possible. This was done to allow easy extension of this database in the future.
  + Although our database is simple, it captures the most essential elements of a forum database system. That is, the forum cannot be formed without a database to save information of a user, a thread, a comment, and many notifications.
  + The **Role** table was added to facilitate the role division problem. Giving a single table for each type of user (admins, moderators, and normal users) is not advisable because it makes the database becomes too complicated. Therefore, adding an attribute, which specify the role of a user into the User table, will prevent this. However, specifying the role by a description string makes controlling the types of user harder in the future so we added the Role table to separate the role types. This can help in further expand the functionalities of the forum system in the future, e.g. adding new user role.
  + **Thread** tables contains the most vital attributes which are the title, the content, and the creation date to easily check when to delete a thread after a pre-specified period of time to describe a thread.
  + The **Comment** table is used to save the comments in all of the threads. It contains the same attributes as a thread except the *title* because a comment usually does not have a *title*.
  + **Notification** table contains the content of a *system notification* such as title, content, and the date when it is sent so that the system can identify when to delete the notification to reduce memory consumption.
  + The **Receives** relation will be created as a table to manage the many-to-many relationship between **User** and **Notification** because the same notification can be sent to many users so it is not necessary to re-create the same notification for each user. The **Receives** table contains the date which a user read the notification and the date which the user receives it to know when to delete the notification. The deletion process of notifications will occur after different period of time for different cases such as the case that the user had already read the notification, the user have not read it but he/she received it, or when the notification is not sent successfully for some other reasons.
  + If a user follow a thread. The **Follows** table is used for that purpose. The **Follows** table contains the *FollowedSince* attribute to identify if the system have to send a notification to a specific user when there is a new comment, a new update, etc of that thread. Also, due to the many-to-many relationship of the relation between **User** and **Thread**,we created a separate **Follows** table.
* In the second stage, we construct the Relational Schema based on the ER diagram, the schema is given below:



* + Our forum will include several basic functionalities such as creating threads, comments, user registration, and a notification system which will notify the user when the thread they subscribed to has a new comment or an update from the thread author.
  + The notification system will only maintain a basic function which is creating only system notifications sent to user.
  + **User** will have several roles which are admin, moderator, and normal user.
  + Because of the many-to-many relationship of **User** and **Thread**, we created a Follows table to save the threads which a particular user follows.
  + The Receives table is also used to represent many-to-many relationship of User and Notification.
  + As stated in stage one, for one-to-many relationships between **User** and **Comment**, **User** and **Thread**, and, finally, **User** and **Role**, we save the primary key of the one end of the relationship in the many end of the relationship, i.e. the table in which one instance matched with many instances of the other table has the primary key as the foreign key in the other table.
* Finally, we create the database and implement a GUI application which is used to perform some operations on the database such as querying the database, updating, inserting, and deleting the data.

# PROJECT DEMO

# CONCLUSIONS

The explosion of the net has led to the development of a variety of social networks. One of the most common type of social network is a forum. There are many forum types which serves different purposes on the Internet. In this project, we have attempted to construct a basic database to serve some basic functionalities of a real forum. Because of the simplicity of the database, the database needs to be improved to be able to run in a large forum. However, the database shown in this report is so far a complete version of the database in a really basic forum.