The ClassNet case study

Suppose you need to develop a database to support a social network (like Facebook) that allows users to exchange messages and comment on content posts. The simplified database contains the tables attached in the script 'PostgreSQL_Relations_No_Constraints'. Kindly note that the script contains no constraints, which is something you will have to work out as part of assignment 1. Based on the script, the tables produced have the following meaning:

- The {USERS} table stores the personal information of your network users. For simplicity, it is assumed that no special privacy settings are available, that is, it is assumed that all stored personal information of users is visible to all network users (therefore, if a user wishes to protect some of his personal data, he simply leaves the corresponding fields of the {USER} table blank). Also, for simplicity, other information, such as the user's password, which should normally be recorded, is omitted.
- The {FRIENDS} table stores the pairs of users who have established friendship through the network.
- The {MESSAGE} table stores the messages posted by users on the 'walls' of the network users. A user can post messages both on his own wall and on the walls of other users.
- The {LIKINGS} table stores the pairs of users with the messages they read and indicated that they liked them.
- The {REMARK} table stores the remarks posted by users under messages written on walls, thus participating in asynchronous electronic discussions on related topics. (Note: to simplify the DB schema, we assume that the remarks of users in the {REMARK} table do not receive 'Like' reports, as, for example, the messages of users posted in the {MESSAGE} table do).
- The {GROUPING} table stores the data of the social groups formed by users in the network, based on their common interests.
- The {GROUP_ADMINISTRATION} table stores the usernames of the administrators of each social group (note: a social group may have more than one administrator). This specific relational table was derived from a N:M association between 'USERS' and 'GROUPS'.
- The { GROUP_MEMBERSHIP} table stores the usernames of the members of each social group. This specific relational table was derived from another N:M association between 'USERS' and 'GROUPS'.

Assignment 1 (Functional dependencies and joinless decomposition)

Now consider the relation R = {UserName, GroupCode, RegistrationDate, Gender, FirstName, LastName, Description, City, Country, GroupName, GroupDescription, CreationDate} and the following functional dependencies:

{UserName} → {FirstName, LastName, Description, City, Country}

{GroupName} → { GroupDescription, CreationDate}

Investigate whether the attribute pair {UserName, GroupCode} is a candidate key of the relation R and produce a decomposition in 3NF/BCNF. Then, implement the relational schema (of the decomposition you propose) in PostgreSQL by updating the script you have been provided with and adding the corresponding sample data.

Additionally, declare yourself as a user and declare your classmates as 'friends' by making appropriate modifications/extensions to the relational schema. Then, create groups corresponding to the four case studies examined during the semester:

/* Insert records into the { GROUPING } Table */

INSERT INTO GROUPING VALUES (4, 'Group 4', 'Appstore', TO_DATE('17/03/2026', 'dd/mm/yyyy'));

INSERT INTO GROUPING VALUES (5, 'Group 5', 'Auctions', TO_DATE('17/03/2026', 'dd/mm/yyyy'));

INSERT INTO GROUPING VALUES (6, 'Group 6', 'GrandPrix', TO_DATE('17/03/2026', 'dd/mm/yyyy'));

INSERT INTO GROUPING VALUES (7, 'Group 7', 'ClassNet', TO_DATE('17/03/2026', 'dd/mm/yyyy'));

Finally, define the users you created as members of the above groups as well as the administrators of each group depending on the case study that each one chooses. Use indicative data of your choice where you deem appropriate.

Assignment 2 (New data types)

The current phase includes relational schema improvements using new data types. The task consists of two parts. Specifically, in the first part you will need to modify your schema and data as follows:

- Add a new attribute 'telephone' of type multivalued to the USERS table and insert appropriate tuples
- Modify the existing attribute GENRE of the USERS table so that it is of type enumerated and insert appropriate tuples
- Modify the USERS table so that it contains a new attribute ACCOMMODATION of type ADDRESS specified as composite including the already existing CITY and COUNTRY and insert appropriate tuples

Then, after the above has been completed, formulate appropriate SQL statements for the queries:

1. Print the details of users who are male (or female respectively)
2. Print the personal details of users from Heraklion who are male and have established friendship ties with other users
3. Print the surname, first name and second phone number of users who declare 2810321040 as their first phone number
4. Print the personal details of users who have declared Heraklion as their city and Greece as their country.
5. Print the personal details of users from Heraklion or Chania who have established friendship ties with other users

Assignment 3 (Specialization hierarchies)

The current exercise is dedicated to the management of special problems and data such as specialization hierarchies and inheritance. Specifically, you will need to implement the following:

- Modify the existing schema so as to create a specialization hierarchy that separates professors from students. Specifically, it is required to distinguish members (USERS) into professors (PROFESSOR) and students (STUDENT) with the friendship relationship only involving members of the student category. The two tables should inherit from the USERS table. The PROFESSOR table will also have the attribute 'Specialty'. The NOVICE table will also have the attribute 'date_of_registration' of type DATE which records the date of registration in the university department.
- Next, import/modify your database data to conform to the new schema. You should import at least one PROFESSOR and your fellow students as STUDENT.
- Once you have completed the above, formulate the SQL statements for the queries:
    1. Print the types of projects for which the expert has expertise under a pseudonym of your choice
    2. Print the pseudonyms of the students (STUDENT)
    3. Print the pseudonym, first name and last name of all teachers (PROFESSOR)
    4. Calculate the total number of members (USERS) of the network
    5. Print the personal information of users who are teachers and declare Rethymno as their city and have not yet established friendship ties with other users.

Assignment 4 (Graphs in Postgres and recursive SQL)

The goal of this phase of the project is (a) to familiarize you with current issues in data management and (b) to introduce you to how relational technology addresses complex issues such as graphs, recursive queries, etc.

4A. Creating a property graph and implementing it in PostgreSQL

You will start by developing (initially on paper or in a tool) a property graph (see theory) that will capture the following:

- Friendship relationships between users of the network that you have already implemented
- Participation relationships (i.e. registration) of users in case studies (note that this relationship is captured in the current version of your implementation)

4A (Creating a property graph and implementing it in PostgreSQL): You will start by developing (initially on paper or using a tool) a property graph (see theory) that will capture: (a) Relationships between drivers and manufacturers (i.e., Luis Hamilton is a member of the Ferrari team) and (b) the driver making the fastest lap in a grand prix (i.e., Luis Hamilton made the fastest lap at 2025 Bahrain grand prix). You should aim to include sufficient data to make querying the graph meaningful. Having designed the property graph, you should then represent the property graph using relational technology. It is at your discretion to choose a technique from those you have learned so far, i.e. the use of normalized relations, unnormalized relations, adjacency lists, etc., while you should study the way in which the alternative nodes and edges that will exist in your graph will be supported (see PostgreSQL inheritance mechanism).

4B (Trajectory queries with recursion): Having represented the property graph with relationships, you should become familiar with modern graph traversal techniques using recursive SQL calls. Specifically, in the current version of the database you have implemented, you should answer the following questions:

1. Count all your friends
2. Count the mutual friends you have with another user of your choice
3. Count all participants in a case study of your choice
4. For each node count the previous ones (in a direction of your choice)
5. Count the case studies declared by students participating in the course

Assignment 5 (No SQL)

The last exercise requires you to choose a NoSQL system of your choice and re-examine (a subset of) the SocialNet database in order to gain hands-on experience with a system of a different type and philosophy than the one you used to develop the database. Your goal should not be to create yet another different implementation of the database in another computing environment (hence the reference to the subset above) but to migrate a portion of your data to a NoSQL environment in order to create a new data source complementary to PostgreSQL. The data you choose should be realistic and reflect real-world conditions. For example, it could be bibliography for the course being examined, encoded in XML/JSON format. Students who manage to present an indicative scenario of data utilization from all the sources they create will receive an additional bonus during

the final grading. I kinldly ask each student to notify me (by email at da@hmu.gr) as to the NoSQL system to be used and a tentative deadline for presenting the work in class.