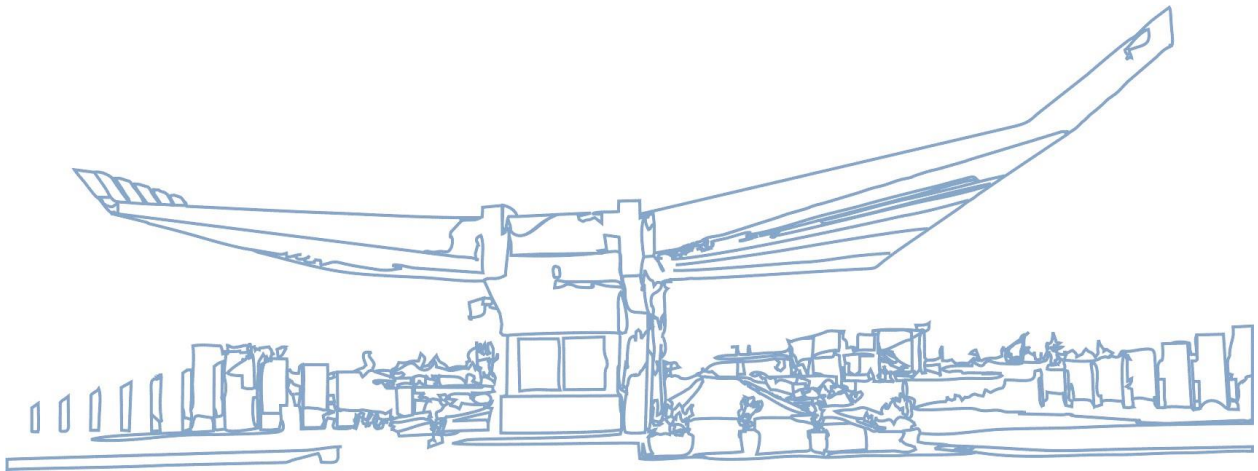


COURSE: DATABASE MANAGEMENT SYSTEM

PROJECT: EPOKA CLUBS DATABASE SYSTEM



Team members:

Aldrin Çifliku

Brinaldo Vorfi

Erlis Këndezi

Fabiona Tafçiu

Juljan Losha

Table of contents:

1. Introduction	3
2. Project outline and Planning	3
3. Requirements	4
4. Research and Idea Consolidation	5
5. Explanation regarding ERD	7
5.1 What the database keeps track of	7
5.2 Relationships	8
5.3 Solution and choices explanation	8
5.4 Solution and choices explanation	9
6. Explanation regarding RS	10
6.1 One to one relationship	10
6.2 One to many relationships	11
6.3 Many to many relationships	11
6.4 Normalization done	11
7. SQL visualization	12

1. Introduction:

Epoka University is home to a variety of clubs. A large number of clubs are already formed and it also gives the students the opportunity to form their own clubs. Therefore, a lot of data needs to be stored, retrieved, or manipulated. This project for the Database management system course requires the creation of a database system to manage the clubs of Epoka University. Currently, these actions are carried out in Excel or using pen and paper. This is not efficient and a lot of time-consuming, also minimizing this way the usage of a database. This project intends the creation of a database system from scratch to fulfill the club board's requirements while keeping as our main goal its efficiency and its wide usage of it. The final purpose of our work would be, why not, to even implement the system into the Epoka database and make it fully functional.

2. Project outline and Planning

One of the most important parts of a project is organizing it and managing the workload so that the work goes smoothly. This is why we specified the steps we are going to take for the development of this project. After being introduced to the topic, we started by forming our group consisting of 5 students and choosing the leader of it. Next, in order to get updated during different stages, it was beneficial to set up a time and day to meet, at least twice a week for brief meetings with the intent to discuss the development of our project, and what we had achieved and even ask for help when needed. Besides this, we also planned to constantly communicate throughout the week for any new discovery or idea that we could implement immediately. This way of working and communicating seemed beneficial for us. We also divided what part of the project everyone wanted to work on (ER diagram, relational schema, creating, inserting the data, writing the managerial queries and documentation) so it would be easier to divide the work among us when the project would be more concrete than just an idea. When the project would be finished, we agreed to all be part of the presentation and explanation to the audience.

To be able to work with the other two steps of the project, we have to first specify the requirements in a written document and implement the other two steps making sure the requirements are fulfilled. So, we intend on gathering the requirements by asking different sources that are going to use the database such as the president of the clubs and the board of each club. After specifying the requirements, our aim is the modeling of the ER diagram, while keeping in mind the constraints of its integrity. After having a final ERD, the next step would be the translation of the ERD to relational schema, which must be normalized and be in accordance with the constraints of integrity, too. Then, we have to create the database using SQL and insert data into the tables that we created. It is also crucial to retrieve the information inserted, filtering it in any way you may need it, therefore it is necessary the creation of managerial queries and implementation of them in the database.

3. Requirements:

We had a meeting with our client, the Epoka clubs' president, who explained to us how the clubs and their management works regarding data storage. She also stated some requirements as to what she needed from this project and how we could make this project useful for the clubs.

The general requirement and purpose of the project is the creation of a database for the Epoka Clubs.

- For more specific requirements:

1. It should be able to store personal information for the members of the group such as name, surname, Epoka mailing address, and phone number.
2. It should store information about the clubs' board members and the president: personal info as in members
3. It should store information about the clubs such as name, the number of members, description, and their links to social networks.
4. Also, storing information about the events held by every club: the location, name, date, description, and attendance that decides according to the club rules if the member gets a certificate.
5. It is important to save the attendance of every member of the club to a certain event in a database and automatically calculate if they get a certificate at the end of the year.

-Except for the requirements we got from our client, the clubs' president, we also did our research. Epoka holds annually the Club Fest, in which the clubs' boards make an appearance, giving information about the club and encouraging students to sign up and become members of a certain club. During the Club Fest, we asked different clubs how we could help them with our project and what was missing from their current method of working. From our research, we gathered these additional requirements:

1. Sports club required that the sports teams created between members be stored in the database.
2. Many clubs offer certificates for every workshop, in addition to Epoka's certificate and it needed to be taken into consideration to give out certificates for these certain events, without having to fulfill the condition that Epoka has regarding this part.
3. Multimedia asked for the passwords of the social networks to be stored in the database since with the change of the president, the new president loses access to them. After further discussions with other clubs, it turned out that it was a general problem, that's why we added the password as an attribute to the club's board.

4. Research and Idea Consolidation

Requirement 1 - The Epoka clubs currently do not have a proper database, but all the clubs work in Excel sheets to save the information that they need. Therefore, when creating our database, the efficiency and improvement of the way information was inserted, processed, and modified were the main purposes. Our core concept of improvement relied on the avoidance of data redundancy and the normalization of the database. Both of these approaches were very beneficial regarding data retrieval and manipulation.

Normalization goes beyond simply standardizing data, improves workflow, increases security, and lessens costs. For instance, if the file size is reduced, data storage and processors won't need to be as large. Additionally, increased workflow due to consistency and organization will ensure that all clubs' boards are able to access the database information as quickly as possible, saving time for other necessary tasks. After normalization, our database will be structured and arranged in a way that is logical for all entities. With increased organization, duplication, and location, errors will be minimized, and outdated versions of data can be more easily updated. Because normalization requires that data is more accurately located and uniformly organized, security is significantly increased.

There are a number of factors that play important roles in the optimization of database performance and speed,

But we could not implement all of them since we were limited in knowledge. However, in most cases, performance issues are caused by poor SQL queries performance. When trying to optimize those queries, we often run into many dilemmas, such as whether to use IN or EXISTS, or whether to write a subquery or a join. One of the most reliable tells that some of the queries are in need of optimization is slow-running queries. There are some specific signs that helped us know when the time to reevaluate the queries was:

- The query now performs adequately.
- The resources needed to optimize further are cost prohibitive.
- We have reached a point of diminishing returns for any further optimization.
- We discover a different solution that renders this unnecessary.

Good database performance is important because it helps to maintain high availability, keeps business-critical systems running smoothly both for customers and internal business users, and saves money, which is always a bonus.

Requirement 2 - Regarding the specifics of our improvements, normalization played a huge part in creating the database much more efficiently. Since we tried to avoid data redundancy and inconsistency from the moment we designed the ERD, it was much easier to control the database and make it work in the most useful way. When we created the ERD schema, we used four relations between entities called "is" that contributed to the advantageous creation of the database. The role of every member of the club inherited the personal information from the Student entity. For example, in SQL, when creating the entities for club board members, such as president, treasurer, etc., instead of creating attributes for name, surname, and email in each of these entities, we used the primary key of the Student entity (StudentID) as a foreign key in

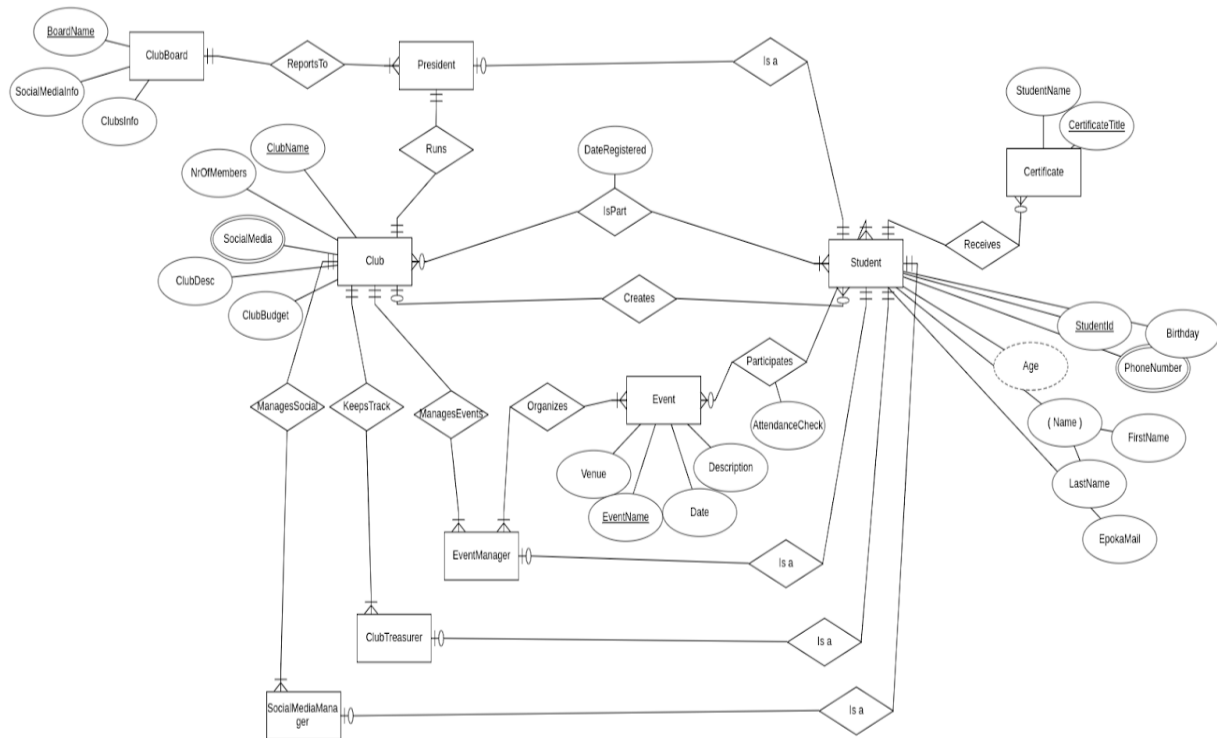
these tables, therefore, we could easily get all the information needed using queries, without having to insert them twice. This change made our database a lot faster and beneficial for the retrieval and manipulation of the data.

Another improvement is the creation of an entity for the phone number. We considered phone numbers as multivalued since there are students who have more than one phone number and they want us to contact them using both numbers, therefore we could save all the values of the phone number for every student attribute in an entity. This way, we were able to avoid multivalued columns that would result in tables that were not considered relational schema.

The second normalization is again regarding multivalues, this time at the club table. That is why we have created a table that contains the social media accounts and the club's name as a foreign key. The club's name and the social media accounts form the primary key of this table. For our purposes, let's call optimal the point at which a query performs acceptably and will continue to do so for a reasonable amount of time in the future. In relation to the increase of speed for the database, there were some steps we followed to make the queries as efficient and fast as possible. We tried to:

- Identify our requirements. By doing so, we had a clear vision about what we wanted to retrieve and how we could more efficiently achieve that.
- Avoid SELECT*
- Create joins with INNER JOIN clauses, not WHERE clauses.
- Define filters using WHERE.
- DELETE and UPDATE in batches.

5. Explanation regarding ERD



Firstly, we will provide the requirements of the database as we retrieved them from the client (club board) and from actors of interest such as current club presidents or event managers. Then we will explain why we chose specific ways to express these requirements such as the “is a” relationship between presidents and students or the attendance check we keep track of in the student-event relationship etc.

5.1 What the database keeps track of

- The database should keep track of clubs. It should be able to store basic information about the club such as the unique club’s name, club description, club budget, number of students registered, and information about the social media (multiple accounts in different social networks).
- We should be able to keep track of students that are related to epoka clubs, either as participants in clubs, as club creators, or as having a role in the clubs (e.g., president). For each student, the database keeps track of the unique student id, the name which is composed of the first name and the last name, the birthday, his/her Epoka mail, and multiple phone numbers.
- The database should keep track of events that are organized. For each event, it should keep track of the venue, date, description, and a unique event name.
- The database should keep track of the certificates issued. For each certificate, it should store the title of the certificate and the name of the student.
- The database should keep track of the presidents.
- The database should keep track of each club’s treasurer.

- The database should keep track of each club's event manager.
- The database should keep track of each club's social media manager.
- Finally, we should be able to store some of the information for the board that is related to Epoka clubs such as clubs' info and clubs' social media info (a password that should be passed to the new social media managers each year).

5.2 Relationships

- Each student can take part in many clubs, but they are not obligated to. Each club must have at least one participant student, but it can have many students. For each student registered the system keeps track of the date of registration.
- Each student can create at most one club but is not obligated to. Each club can be created by many students, but it can also be an existent club or a "default club" (clubs such as programming club that were created with the initiative of the university itself)
- A student may receive many certificates, but it is not sure that he/she will receive one. A certificate can be received by strictly one student.
- A president runs only one club, and each club is run only by one president.
- The president reports only to the club board and the board has many presidents reporting to it
- An event manager manages events for only one club. A club may have one or many event managers managing events for them.
- A social media manager manages the social media accounts for only one club. A club may have one or many social media managers managing their social media accounts.
- A treasurer can keep track of the budget for only one club. A club may have one or many club treasurers to keep track of their expenses.
- An event manager organizes one or many events and each event can be organized by one or many event managers.
- An event manager is a student (clubs are run entirely by students). A student may or may not be a manager.
- A social media manager is a student. A student may or may not be a social media manager.
- A club treasurer is a student. A student may or may not be a club treasurer.
- A president is a student. A student may or may not be a club president.
- A student may participate in many events however he/she is not obligated to. An event should have at least one student, but it can of course have many students participating. For each student that takes part in an event, the database keeps track of an attendance check.

5.3 Solution and choices explanation

When we designed the ER diagram, we felt the need to explain some of our choices as they may seem uncanny at first glance. Behind any of these choices, there is a reasoning that is either related to a possible user interface that might implement the database or to the efficiency of the database in terms of memory consumption and speed of execution. So, let us explain them one by one.

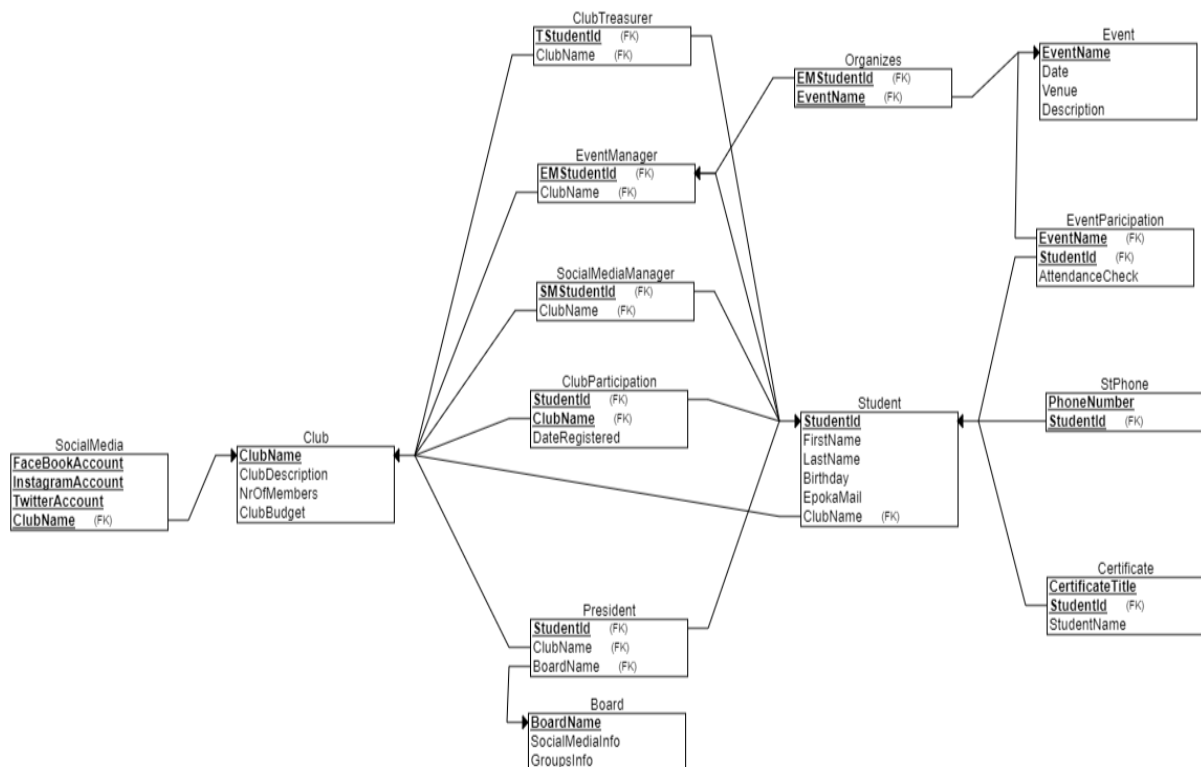
The first detail that catches the eye is of course the inclusion of the club board as an entity. On first impression having the club board as an entity might not seem like an optimal choice. After all the club board is the one monitoring all clubs so one might think that it is not necessary to include it in the database itself. However, the club board is the only constant in the equation of the Epoka clubs. This entity never changes and will exist as long as the Epoka clubs exist. This

is the reason why the club board is crucial to keep the database consistent year after year. How do we achieve this? It is actually pretty simple. The club board does not keep track of generic data such as names, leaders, etc. This data is redundant because it is always changing. The club board however serves as a container for data that doesn't necessarily need to be deleted or altered each year, such as social media accounts and passwords for each club or general club information. This information can be retrieved in the form of UI by officials of the club board and be shared with the next generation of club participants and/or "governors" (presidents, managers, etc).

Secondly, it is quite noticeable that for each student that participates in an event, we also keep track of an attendance check. One has the right to ask why the database should keep track of another value that confirms the attendance when the database is already keeping track of participation as a relationship. Now, this idea comes from the database of the google developer's club whose leader we interviewed during the requirements gathering process. So, the goal is to keep track of all the students that have confirmed they will take part in an event whether they show up or not. For each student that shows up the attendance check takes a positive value. This is quite similar to attendance in a classroom where the list of students has all the students registered for the course, but the teacher marks present only those that have shown up. While the idea was inspired by google the implementation is original and it provides a much easier and more understandable implementation of a possible future interface.

And last but definitely not least is the relationship between the president, event manager, social manager, treasurer, and student. From the ER diagram we can see that they form an "is a" relationship with each other. Why? During the requirements gathering process, we were able to take a closer look into the structure of the club and we noticed that everything within a club is done by students. And the best part of it all is that the information that is stored for the president is the same as that stored for a simple club participant. Now there are cases where the president of one club might just be a participant in another club. In this case, we would have to keep track of the same information twice: the first time as president and the second time as a student. That's why we came up with this solution where the president (or any of the other roles) has an "is a" relationship with the student entity. So, we can retrieve the data for them from the table that keeps the data for students. A natural question that might come to mind is: wouldn't it be better to create relationships between students and clubs such as "is a manager" or "is president"? The simple answer is no, not because this is impossible or wrong but simply because the data would become too intricate for anyone who would try to implement the database in SQL (or any other format). As a result of this normalization that we have done to the database, we have avoided the duplication of the information of an average of 150 persons each year.

6. Explanation regarding RS



In this section, we will explain briefly how we have converted the ERD shown earlier in this RS. The process was not complicated since we made those choices that would lead us to less normalization needed for the database.

6.1 One-to-one relationship:

- The only one-to-one relationships that we have are those between the student and the club roles (social manager, president, event manager, treasurer). In the relational schema, we decided that the student id goes as a foreign key to these respective tables where it becomes a primary key also. This is done for efficiency since the student entity is the mandatory side of the relationship meaning there will be no null fields for this foreign key. But most importantly it is done for the logical proficiency of the database. Later when writing queries it is important to have the foreign key at these tables (social manager, president, event manager, treasurer) since these tables are the ones receiving the information.

6.2 One-to-many relationships:

- The first one-to-many relationship is when students create a club. In such cases, the club's name is stored as a foreign key in the student table.

- Another one-to-many relationship is that of students and certificates where the student id is set as the foreign key in the table certificate. (so as to be able to find the information for the student who received the certificate)
- The relationship between roles (president, event manager, etc) and the club is also one-to-many. The club's name is sent as a foreign key in all those tables.
- The last one-to-many relationship is that of the president and the board. The club board name is stored as a foreign key at the president's table.

6.3 Many-to-many relationships:

- The first many-to-many relationship is that of the participation of students in clubs. In this case, we have created the club participation table which has a composed primary key that is made of the student id and the club's name which come to this table as foreign keys from their respective tables. The table has also a column that keeps the date on which a student was registered.
- Another many-to-many relationship is that of event organization between event managers and events which is depicted by the table organizes which only takes as foreign key both the primary key of the event manager and of the event.
- Participation in events is a many-to-many relationship depicted in the table event participation which also keeps track of attendance.

6.4 Normalization done:

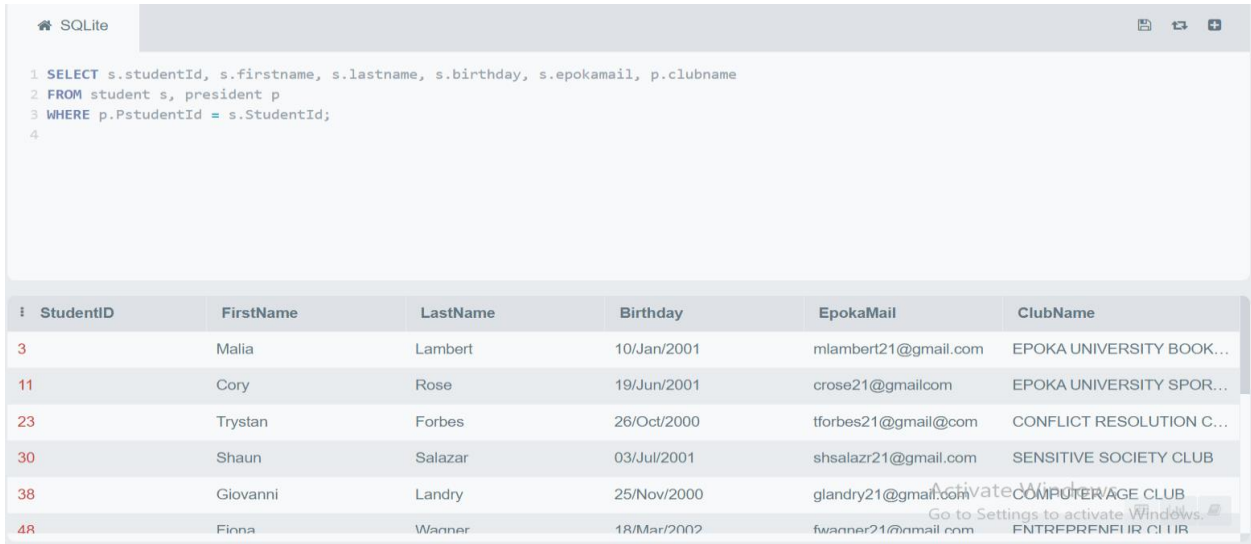
From the beginning, we planned the database in such a way that normalization would not be needed on large scale. However, this is a process that we were conscious we could not avoid. The normalization we have done here is related to the club and student tables (the core tables).

- Firstly, the student table contains multivalues (phone numbers) so we cannot just have a single table. That is why we have created the table StPhone which takes the student id as a foreign key which alongside the phone number it forms a composed primary key for this new table
- The second normalization is again regarding multivalues, this time at the club table. That is why we have created a table that contains the social media accounts and the club's name as a foreign key. The club's name and social media accounts form the primary key of this table.

6. SQL visualization

For a better visualization of the database and how it looks like from the user point of view, we need to use managerial queries. Here are some of many examples (the managerial queries are in the SQL pdf) of how the tables for certain needs of retrieval of information look like.

1- /* Get all info for each club's president */



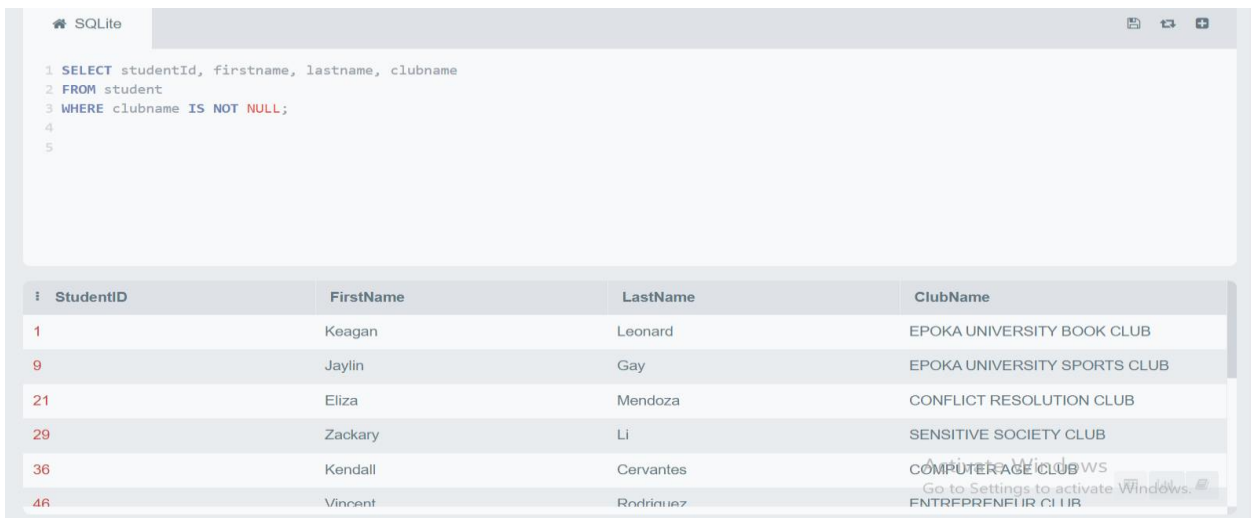
```

1 SELECT s.studentId, s.firstname, s.lastname, s.birthday, s.epokamail, p.clubname
2 FROM student s, president p
3 WHERE p.PstudentId = s.StudentId;
4

```

StudentID	FirstName	LastName	Birthday	EpokaMail	ClubName
3	Malia	Lambert	10/Jan/2001	mlambert21@gmail.com	EPOKA UNIVERSITY BOOK...
11	Cory	Rose	19/Jun/2001	crose21@gmail.com	EPOKA UNIVERSITY SPOR...
23	Trystan	Forbes	26/Oct/2000	tforbes21@gmail.com	CONFLICT RESOLUTION C...
30	Shaun	Salazar	03/Jul/2001	shsalazr21@gmail.com	SENSITIVE SOCIETY CLUB
38	Giovanni	Landry	25/Nov/2000	glandry21@gmail.com	COMPUTER AGE CLUB
48	Fiona	Wanner	18/Mar/2002	fwanner21@gmail.com	ENTREPRENEUR CLUB

2-/* Find the information for all the creators of each club */



```

1 SELECT studentId, firstname, lastname, clubname
2 FROM student
3 WHERE clubname IS NOT NULL;
4
5

```

StudentID	FirstName	LastName	ClubName
1	Keagan	Leonard	EPOKA UNIVERSITY BOOK CLUB
9	Jaylin	Gay	EPOKA UNIVERSITY SPORTS CLUB
21	Eliza	Mendoza	CONFLICT RESOLUTION CLUB
29	Zackary	Li	SENSITIVE SOCIETY CLUB
36	Kendall	Cervantes	COMPUTER AGE CLUB
46	Vincent	Rodriguez	ENTREPRENEUR CLUB

3-/* Find the information for the event managers given the club's name */

SQLite

```

1 SELECT Student.StudentID, Student.firstname, Student.lastname, Student.Birthday
2 FROM Student, EventManager
3 WHERE Student.studentid = EventManager.emstudentid AND EventManager.clubname = 'EPOKA UNIVERSITY BOOK CLUB';
4
5

```

StudentID	FirstName	LastName	Birthday
3	Malia	Lambert	10/Jan/2001

Activate Windows
Go to Settings to activate Windows.

4-/* Find the information for all the event managers in all clubs*/

SQLite

```

1 SELECT Student.StudentID, Student.firstname, Student.lastname, Student.Birthday
2 FROM Student, EventManager
3 WHERE Student.studentid = EventManager.emstudentid;
4

```

StudentID	FirstName	LastName	Birthday
3	Malia	Lambert	10/Jan/2001
11	Cory	Rose	19/Jun/2001
23	Trystan	Forbes	26/Oct/2000
71	Kash	Tapia	28/May/2001
111	Ethan	Lowery	20/Jul/2001
123	Railee	Anderson	20/Jul/2002

Activate Windows
Go to Settings to activate Windows.

5-/* Show the name of the club with the least nr of members */

SQLite

```

1 SELECT clubname
2 FROM Club
3 WHERE clubnrofmembers = (SELECT MIN(clubnrofmembers) FROM club);
4

```

ClubName
CONFLICT RESOLUTION CLUB

Activate Windows
Go to Settings to activate Windows.

6-/* Get information about the club's name and club budget of the clubs with the highest budget*/

SQLite

```

1 SELECT clubname, clubbudget
2 FROM Club
3 WHERE clubbudget = (SELECT MAX(clubbudget) FROM Club);
4

```

ClubName	ClubBudget
PROGRAMMING CLUB	400

Activate Windows
Go to Settings to activate Windows.

7-/*Find when a specific student registered in a specific club*/

SQLite

```

1 SELECT dateregistered
2 FROM ClubParticipation
3 WHERE studentid = (SELECT studentid
4                     FROM Student
5                     WHERE firstname = 'Orion' AND lastname = 'Robles')
6 AND clubname = 'PROGRAMMING CLUB';
7

```

DateRegistered
16/Dec/2022

Activate Windows
Go to Settings to activate Windows.

8-/* Show the names and the certificate awarded for all students that earned a certificate*/

SQLite

```

1 SELECT studentname, certificatetitle
2 FROM Certificate;
3

```

StudentName	CertificateTitle
Alejandro Cox	Book Night Certificate
Daphne Vincent	POKA Football League Certificate
Kolten Barton	Poetry Competition Certificate
Avery Barrera	Training Program in Insurance Certificate
Carlee Solomon	Epoka Talks Certificate
Ashly Mneas	Film Production and Art in Albania Certificate

Activate Windows
Go to Settings to activate Windows.

9-/* Show the name of the least attended event and its attendance*/

SQLite

```

1 SELECT eventname, Min(attendance)
2 FROM (SELECT eventname AS eventname, COUNT(*) AS attendance
3 FROM eventparticipation
4 WHERE attendancecheck = 'True'
5 GROUP BY eventname);
6
7

```

eventname	Min(attendance)
Book Night	4

Activate Windows
Go to Settings to activate Windows.

10-/*Find the student with most club participations*/

SQLite

```

1 SELECT firstname, lastname
2 FROM Student
3 WHERE studentid = (SELECT studentid FROM (SELECT studentid, Max(participations)
4 FROM (SELECT studentid, COUNT(*) AS participations
5 FROM ClubParticipation
6 GROUP BY studentid)));
7
8

```

FirstName	LastName
Keagan	Leonard

Activate Windows
Go to Settings to activate Windows.

11- /* Show student id and date of the student that was first registered in a specific club*/

SQLite

```

1 SELECT studentid, dateregistered, clubname
2 FROM ClubParticipation
3 WHERE dateregistered = (SELECT MIN(dateregistered) FROM ClubParticipation WHERE clubname = "PROGRAMMING CLUB" );
4
5

```

StudentID	DateRegistered	ClubName
51	10/Dec/2022	ENTREPRENEUR CLUB
58	10/Dec/2022	PROGRAMMING CLUB

Activate Windows
Go to Settings to activate Windows.

12-/* Select information for students of a specific club */

SQLite

```

1 SELECT firstname, lastname, epokamail, birthday, p.phonenumber
2 FROM student s, stphone p
3 WHERE s.studentid
4 IN (SELECT studentid FROM ClubParticipation WHERE clubname = 'PROGRAMMING CLUB')
5 AND s.studentid = p.studentid;
6

```

i	FirstName	LastName	EpokaMail	Birthday	PhoneNumber
	Journey	Barker	jbarker21@gmail.com	19/Oct/2001	(296) 559-0841
	Orion	Robles	orobles21@gmail.com	31/Dec/1999	(694) 950-0015
	Triston	Lowe	tlowe21@gmail.com	20/Jan/2000	(853) 989-2053
	Lydia	Rodgers	lrodgers21@gmail.com	25/May/2000	(703) 406-2846
	Hector	Sampson	hsampson21@gmail.com	18/Dec/2001	(498) 783-4384
	Isabella	Rubio	irubio21@gmail.com	10/Jan/2001	(921) 983-0733
	Anaya	Montes	amontes21@gmail.com	17/Mar/2001	(463) 419-1323
	Dashawn	Carson	dcarson21@gmail.com	12/Mar/2001	(737) 313-8660
	Maria	Mcclain	mmclain21@gmail.com	19/May/2000	(923) 287-1941
	Randall	Donaldson	rdonaldson21@gmail.com	27/Jun/2000	(947) 489-4793
	Sebastian	Velez	svelez21@gmail.com	14/Jun/2001	(599) 299-7138
	Kolten	Ortega	kortega21@gmail.com	16/Aug/2000	(224) 261-4356
	Deenan	Mahonev	dmahonev21@gmail.com	11/Jul/2001	(868) 831-6881

13-/* Select attendance list for an event*/

SQLite

```

1 SELECT s.firstname, s.lastname, e.attendancecheck AS Attendance
2 FROM Student s, EventParticipation e
3 WHERE s.StudentID = e.studentid AND e.eventname = 'EPOKA Football League';
4

```

i	FirstName	LastName	Attendance
	Daphne	Vincent	True
	Savanna	Shannon	True
	Kaleb	Downs	True
	Chaya	Wolfe	True
	Parker	Hamilton	True
	Konnor	Richards	False
	Kadyn	Strong	True
	Judah	Farmer	True
	Lorelei	Franklin	False

14-/* Select events that have more than 5 students attending*/

SQLite

```

1 SELECT eventname, COUNT(studentid) AS Attended
2 FROM EventParticipation
3 WHERE attendancecheck = 'True'
4 GROUP BY eventname
5 HAVING COUNT(studentid) > 5;
6

```

EventName	Attended
EPOKA Football League	7
Poetry Competition	6
Training Program in Insurance	6

15-/* Get all info for a club */

SQLite

```

1 SELECT c.clubname, c.clubdescription, c.clubbudget, c.clubnrofmembers, s.facebookaccount AS Facebook,
2 s.instagramaccount AS Instagram,
3 s.twitteraccount AS Twitter
4 FROM club c, SocialMedia s
5 WHERE s.clubname = c.ClubName;

```

ClubName	ClubDescription	ClubBudget	ClubNrOfMem...	Facebook	Instagram	Twitter
EPOKA UNIVE...	The Epoka Univ...	300	8	EPOKA UNIVE...	EPOKA_UNIVE...	EPOKAUNIVERSITY...
EPOKA UNIVE...	The Epoka Spor...	200	12	EPOKA UNIVE...	EPOKA_UNIVE...	EPOKAUNIVERSITY...
CONFLICT RE...	Epoka Conflict ...	150	7	CONFLICT RE...	CONFLICT_RE...	CONFLICTRESOLU...
SENSITIVE SO...	Sensitive Societ...	300	8	SENSITIVE SO...	SENSITIVE_SO...	SENSITIVESOCIET...
COMPUTER A...	Epoka Compute...	250	10	COMPUTER A...	COMPUTER_A...	COMPUTERAGECLUB
ENTREPRENE...	The mission of ...	200	8	ENTREPRENE...	ENTREPRENE...	ENTREPRENEURC...
PROGRAMMIN...	If YOU can write...	400	15	PROGRAMMIN...	PROGRAMMIN...	PROGRAMMINGCLUB
THEATER CLUB	Epoka THEATE...	350	10	THEATER CLUB	THEATER_CLUB	THEATERCLUB
ARCHISPACE ...	ArchiSpace esht...	250	10	ARCHISPACE ...	ARCHISPACE_...	ARCHISPACECLUB
FUTURE ENGI...	The mission of t...	200	12	FUTURE ENGI...	FUTURE_ENGI...	FUTUREENGINEER...
MUSIC CLUB	You will what ev...	150	8	MUSIC CLUB	MUSIC_CLUB	MUSICCLUB
YOUNG ECON...	The Young Eco...	350	12	YOUNG ECON...	YOUNG_ECON...	YOUNGECONOMIS...
NEW GENERA...	Epoka NEW GE...	300	10	NEW GENERA...	NEW_GENERA...	NEWGENERATION...

16- /*Show the name of the most attended event*/

SQLite

```
1 SELECT eventname, Max(attendance)
2 FROM (SELECT eventname AS eventname, COUNT(*) AS attendance
3 FROM eventparticipation
4 WHERE attendancecheck = 'True'
5 GROUP BY eventname);
6
```

eventname	Max(attendance)
EPOKA Football League	7

Activate Windows
Go to Settings to activate Windows.

17- /* Show the name of the earliest event*/

PostgreSQL

```
1 SELECT eventname
2 FROM EVENT
3 WHERE eventdate = (SELECT MIN(eventdate) FROM EVENT);
4
```

eventname
EPOKA Football League

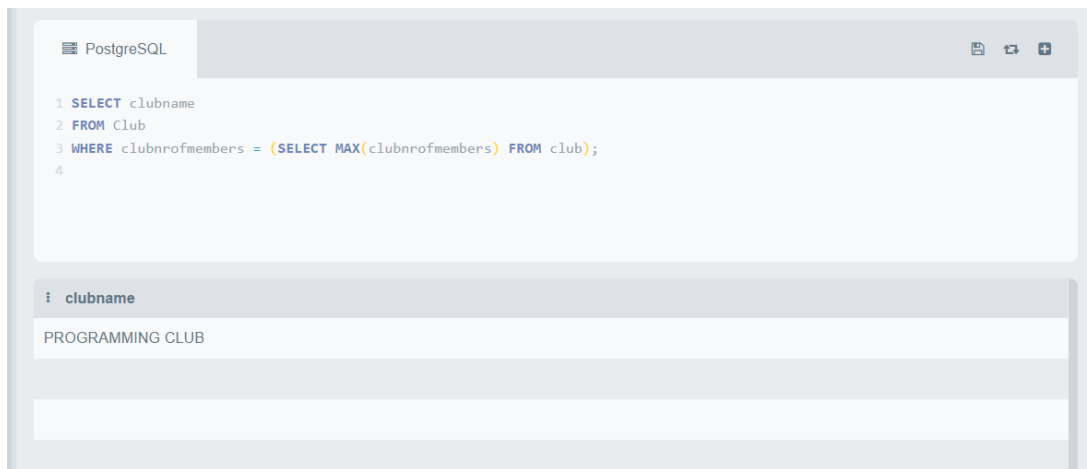
18-/*Show the name of the latest event*/

PostgreSQL	
<pre>1 SELECT eventname 2 FROM EVENT 3 WHERE eventdate = (SELECT MAX(eventdate) FROM EVENT); 4</pre>	
eventname	
Epoka Talks	

19-/* Check the events of a specific club has and show their name*/

PostgreSQL	
<pre>1 SELECT eventname 2 FROM Organizes o 3 WHERE emstudentid IN (SELECT emstudentid 4 FROM EventManager 5 WHERE clubname = 'EPOKA UNIVERSITY BOOK CLUB'); 6</pre>	
eventname	
Book fest	
Book Night	

20- /* Show the name of the club with the most nr of members */



-Other SQL managerial queries:

21 - /* Get all phone numbers of a specific student given name NAME and surname SURNAME */

```
SELECT p.phonenumber
FROM StPhone p
WHERE p.studentId = (SELECT studentId FROM student WHERE firstname = 'Fiona' AND
Lastname = 'Wagner');
```

22- /* Find the information for the treasurer given the club's name */

```
SELECT Student.StudentID, Student.firstname, Student.lastname, Student.Birthday
From Student, Treasurer
WHERE Student.studentid = Treasurer.TStudentID And Treasurer.clubname = 'PROGRAMMING
CLUB';
```

23-/* Find the information for the social media managers given the club's name */

```
SELECT Student.StudentID, Student.firstname, Student.lastname, Student.Birthday
From Student, SocialManager
WHERE Student.studentid = SocialManager.smstudentid And SocialManager.clubname =
'PROGRAMMING CLUB';
```

24- /* Find the information for all the social media managers in all clubs*/

```
SELECT Student.StudentID, Student.firstname, Student.lastname, Student.Birthday
From Student, SocialManager
WHERE Student.studentid = SocialManager.smstudentid;
```

25- /* Find the information for all the treasurers in all clubs*/

```
SELECT Student.StudentID, Student.firstname, Student.lastname, Student.Birthday
From Student, Treasurer
WHERE Student.studentid = Treasurer.TStudentID;
```

```
26-/* Find the information for all clubs with more than a certain number of members*/  
SELECT *  
FROM Club  
WHERE clubnrofmembers >= 10;
```

```
27- /* Find the average number of participants in a club*/  
SELECT AVG(clubnrofmembers)  
FROM Club;
```

```
28- /* Show the name of the club with the most nr of members */  
SELECT clubname  
FROM Club  
WHERE clubnrofmembers = (SELECT MAX(clubnrofmembers) FROM club);
```

```
29- /*Show the name of all students and their contact info(email and phone number)*/  
select firstname,lastname, stphone, epokamail  
from student , stphone  
WHERE student.studentid = stphone.studentid;
```

```
30- /* Show social media accounts for all clubs */  
SELECT *  
FROM socialmedia;
```

```
31 /* Get information about the club name and club budget of the clubs with the lowest budget*/  
SELECT clubname, clubbudget  
FROM Club  
WHERE clubbudget = (SELECT MIN(clubbudget) FROM Club);
```

```
32- /*Show the information of the event manager who is organizing a specific event*/  
select studentid, em.clubname as Organizesfor, s.firstname, s.lastname, s.epokamail, s.birthday  
from EventManager em, Student s  
where em.EMStudentID = s.StudentID and emstudentid = (select emstudentid  
from Organizes  
where eventname = 'EPOKA Football League');
```

```
33-/*Show the board information*/  
Select boardname,socialmediainfo,groupsinfo  
from Board;
```

```
34-/* Show the club descriptions for all clubs */  
Select clubname,clubdescription  
FROM Club;
```

```
35- /*Show the students who have won the "Book Night Certificate" certificate.*/  
SELECT studentname, certificatetitle  
FROM Certificate  
WHERE certificatetitle = "Book Night Certificate";
```

```
36-/*Show the students who have won the “Poetry Competition Certificate” certificate.*/  
SELECT studentname, certificatetitle  
FROM Certificate  
WHERE certificatetitle = "Poetry Competition Certificate";
```

```
37- /*Show the students who have won the “Training Program in Insurance Certificate” certificate.*/  
SELECT studentname, certificatetitle  
FROM Certificate  
WHERE certificatetitle = "Training Program in Insurance Certificate";
```

```
38- /*Show the students who have won the “EPOKA Football League Certificate” certificate.*/  
SELECT studentname, certificatetitle  
FROM Certificate  
WHERE certificatetitle = "EPOKA Football League Certificate";
```

```
39- /*Show the students who have won the “Epoka Talks Certificate” certificate.*/  
SELECT studentname, certificatetitle  
FROM Certificate  
WHERE certificatetitle = "Epoka Talks Certificate";
```

```
40- /*Show the students who have won the “Film Production and Art in Albania Certificate”  
certificate.*/  
SELECT studentname, certificatetitle  
FROM Certificate  
WHERE certificatetitle = "Film Production and Art in Albania Certificate";
```

```
41- /*Show the students who have won the “Peace Forum | Debating Europe 100 Years after World  
War I Certificate” certificate.*/  
SELECT studentname, certificatetitle  
FROM Certificate  
WHERE certificatetitle = "Peace Forum | Debating Europe 100 Years after World War I Certificate";
```

```
42- /* Show participants of a specific event that are also club treasurers*/  
select firstname, lastname  
from Student s, EventParticipation e  
where s.StudentID = e.StudentID and  
s.studentid in (select Treasurer.tstudentid  
from Treasurer) and e.eventname = 'Poetry Competition';
```

```
43- /* Show participants of a specific event that are also club event manager*/  
select firstname, lastname  
from Student s, EventParticipation e  
where s.StudentID = e.StudentID and  
s.studentid in (select EventManager.emstudentid  
from EventManager) and e.eventname = 'Poetry Competition';
```

```
44-/* Show participants of a specific event that are also club event manager*/  
select firstname, lastname  
from Student s, EventParticipation e  
where s.StudentID = e.StudentID and  
s.studentid in (select SocialManager.smstudentid  
from SocialManager) and e.eventname = 'Poetry Competition';
```

45- /*Find the student with least club participations*/

```
SELECT firstname, lastname
```

```
from Student
```

```
where studentid = (SELECT studentid from(select studentid, Min(participations)
```

```
    From (SELECT studentid, Count(*) as participations
```

```
    from ClubParticipation
```

```
    group by studentid))));
```

46- /* Show student id and date of the student that was first registered in a specific club*/

```
SELECT studentid, dateregistered, clubname
```

```
FROM ClubParticipation
```

```
Where dateregistered = (SELECT MIN(dateregistered) FROM ClubParticipation WHERE clubname  
= "PROGRAMMING CLUB" );
```

47- /*Show clubs that have more members than the avg of members per club */

```
SELECT clubname, clubnrofmembers
```

```
FROM Club
```

```
WHERE clubnrofmembers > (SELECT AVG(clubnrofmembers) FROM Club);
```

48- /*Show clubs that have less members than the avg of members per club */

```
SELECT clubname, clubnrofmembers
```

```
FROM Club
```

```
WHERE clubnrofmembers < (SELECT AVG(clubnrofmembers) FROM Club);
```