Databases - Final Project (MAY.2020)

Teacher:
Andrea Corradini

Group Members:
Constantin Razvan Tarau
Marius Daniel Munteanu
Paul Panaitescu

Table of Contents

I. Report

A. Miniworld Description

App Description

Socialbook is a social application similar to Facebook or Instagram. The purpose of the app is to stimulate human interaction and facilitate communication between people all over the world. In order to join the application you must have an account. After you have created an account you can make posts or stories, write comments and replies or be part of different groups.

Scenario

Socialbook consists of posts that can be published by users either on their profile, or on a group. A post is made up of a unique identifier, its content, the original poster, the date the post was published as well as an indicator if the post is monetarily sponsored.

Each account has a unique identifier, person's first name and last name, personal email, a chosen password, person's gender, the date when the user joined the community. Other than that, consider a way to remove an account without losing the account's data.

On a post, users can leave comments, and each comment will have its unique identifier, the actual content of the post, and the date the comment was made.

Users can respond to other users' comments under the form of a reply, that has a unique identifier, the content of the reply and the date.

Users can react to posts, comments and replies. In addition to reacting to a post, the user can also share it.

As mentioned above, users also have the option to publish posts inside a group, and the group has its own identifier, a name and a short description. In order to post, the user must be a member of that group.

Lastly, users can also publish posts that get archived a day after it's published, and those are known as stories. A story will have its unique identifier, the content under the form of a picture, and the date it was posted.

B. Modeling and Design

Our database is called SocialBook and it is organized in 2 major categories:

- Base Tables
- Bridge Tables

In total there are 13 tables, of which 6 are base tables and 7 are bridge tables, the latter being used to make the relations between the base tables.

The Base Tables are the following:

- Account (**id**, firstName, lastName, email, password, birthDate, gender, joinDate, active)
- Post (**id**, ***accountId***, content, sponsored, date)
- Comment (**id**, ***postId***, ***accountId***, content, date)
- Reply (**id**, ***commentId***, ***accountId***, content, date)
- Group (**id**, name, description)
- Story (**id**, ***accountId***, photoLink, postDate)

The Bridge Tables are the following

- PostLikes (***postId***, ***accountId***)
- PostShares (***postId***, ***accountId***)
- CommentLikes (***commentId***, ***accountId***)
- ReplyLikes (***replyid***, ***accountId***)
- Friendship (***receiverAccountId***, ***senderAccountId***, pending, denied)
- GroupPosts (***socialGroupId***, ***postId***)
- GroupParticipants (***socialGroupId***, ***accountId***)

***Note**: For more details about the table structure, check the included DB_Structure.pdf document.*

C. Entities and Relationships

We have implemented 2 different versions of ER Diagrams to have a better understanding of the relationships between tables, to illustrate the logical structure of the database, and to have an overview of how the tables should look like. These diagrams illustrate the entities, attributes, relationships, and the cardinality ratios.

Image C.1. - ER Diagram

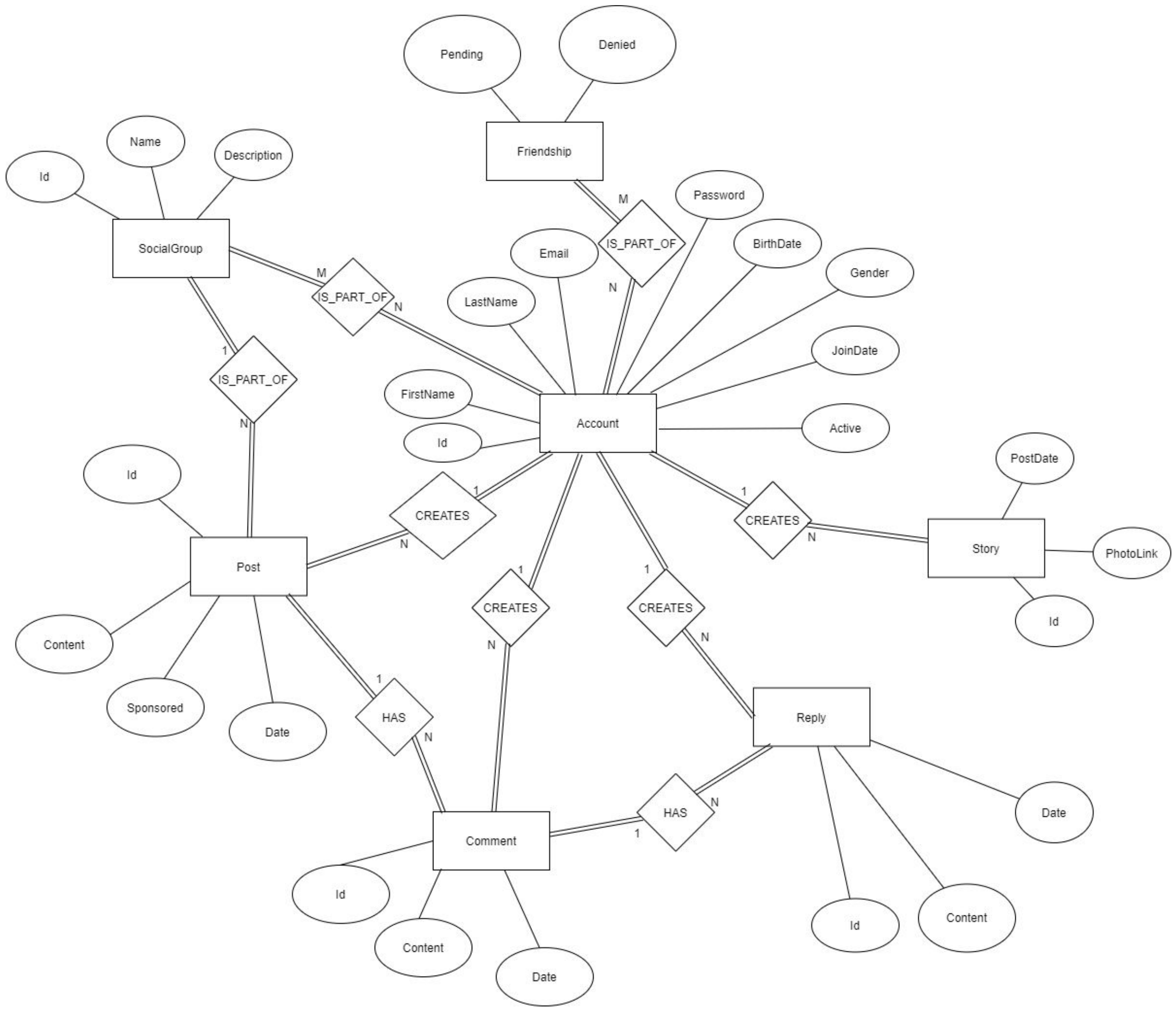


Image C.2. - ER Diagram with PK and FK

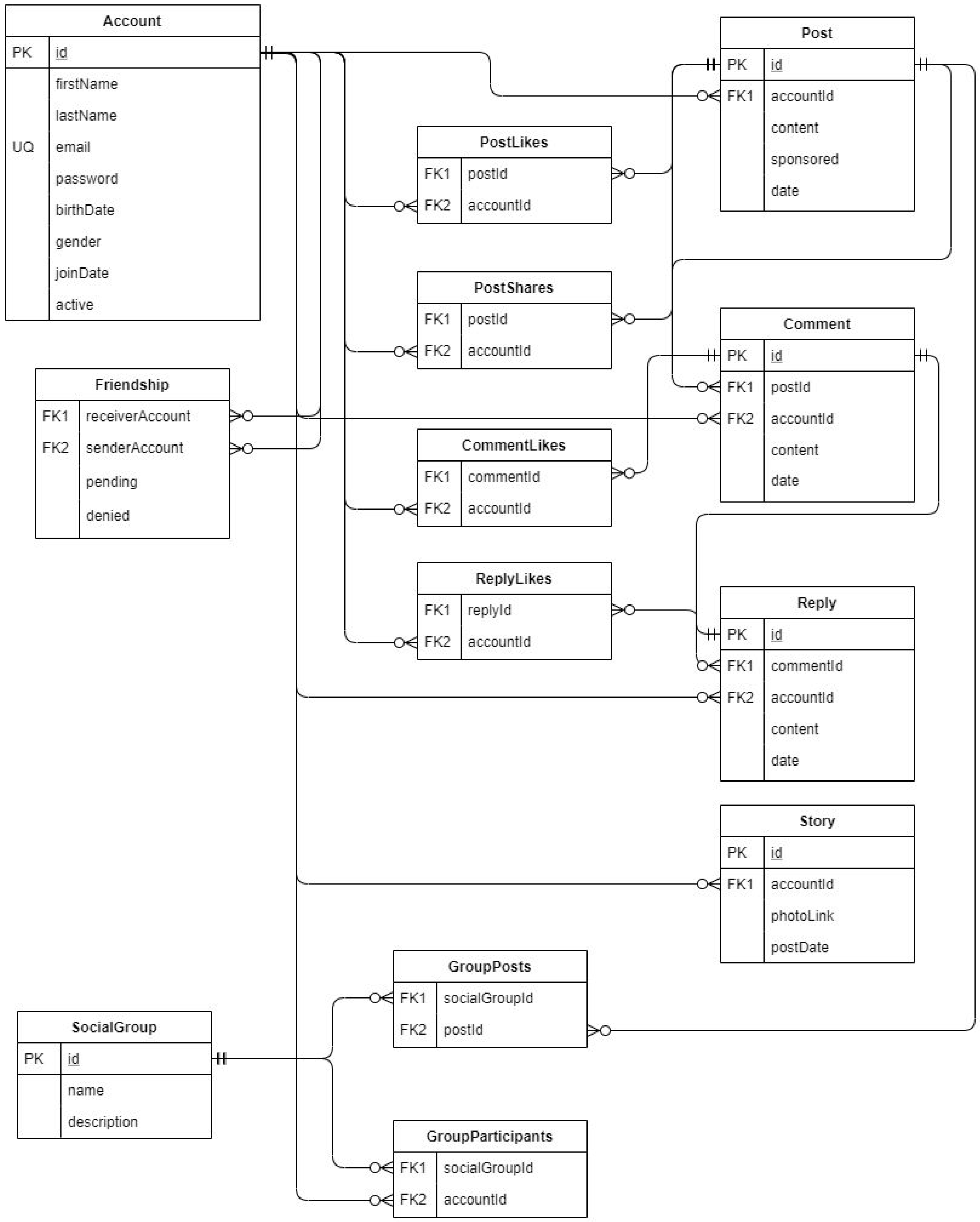
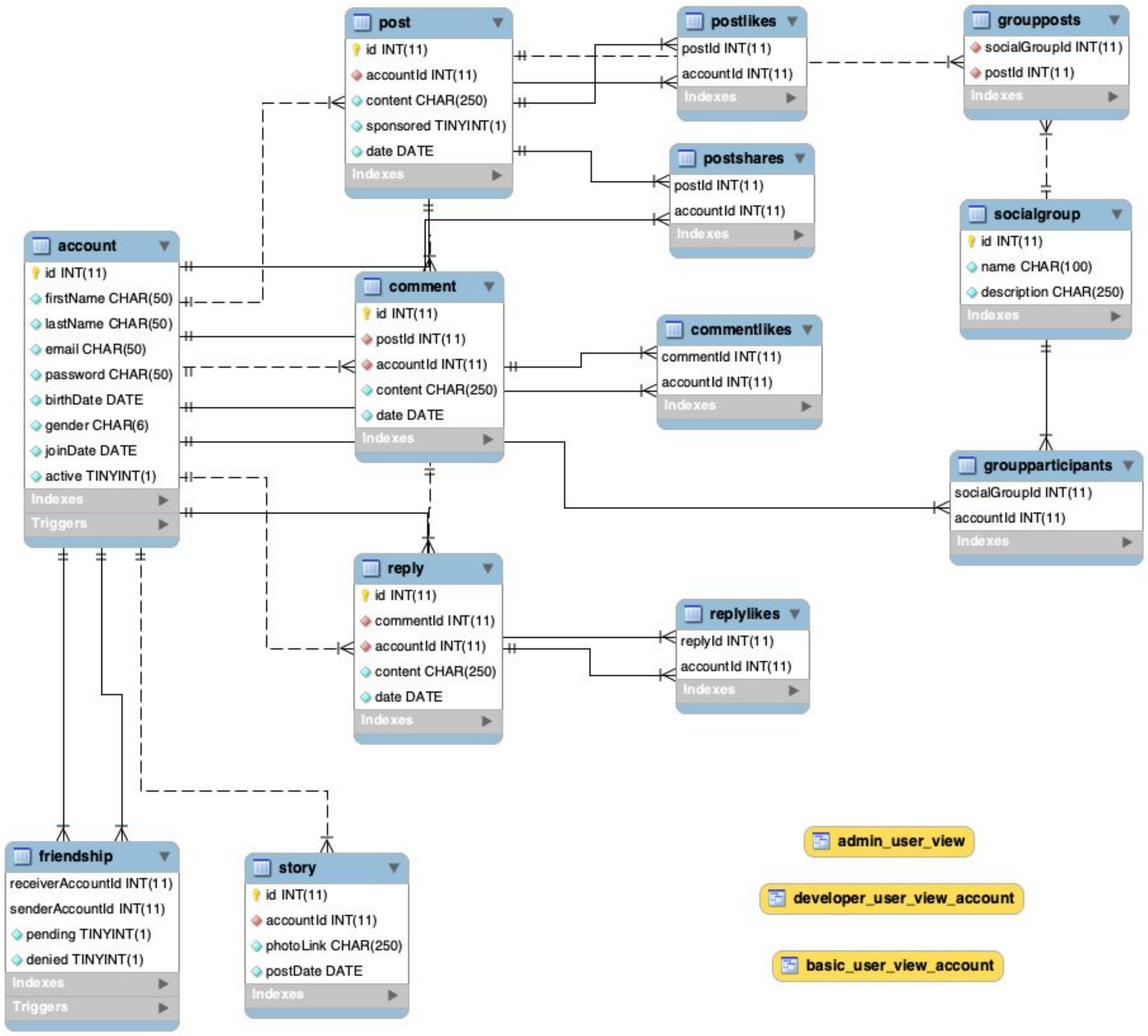


Image C.2. - Exported EER Diagram



D. Normalization

We have applied the data normalization rules, in order to optimize the database, minimize the redundancy, avoid data inconsistency and referential integrity problems.

First Normal Form

We have applied the first normal form, meaning that the attributes of all the tables have no repeating groups of data, since the very beginning, when we were designing the table. We made sure that the attributes contain only single and unique values.

Second Normal Form

The second normal form was also applied in our database, from the very early stages of the development. We designed the table in such a way that all the non-primary key attributes are fully-dependent on the primary key.

For example, columns like `firstName` or `lastName` in the Account table are not primary keys and they depend only on the `id` of the account, meaning we could have multiple accounts that share the same values on the aforementioned columns, but we are still able to differentiate between those rows.

Third Normal Form

In order to make sure we get rid of more possible anomalies we had to make sure that our tables are also in the third normal form, meaning that our tables are in the first 2 normal forms, but all non-primary key attributes are not dependent on any other non-primary keys.

For example, we have one occasion where 3NF is not respected, namely the `pending` and `denied` columns in the Friendship table. In this case, we can argue that while a friendship request is pending (as in the receiver has yet to accept or deny it), the `denied` column should not have any meaningful value, therefore making it dependent on the `pending` column. One way to go about this would be to merge the two columns into one, and have that column hold the state of the request, and it could have limited possible values, like “pending”, “accepted” and “denied”.

E. Physical Database

We used MySQL RDBMS for this project, because we wanted to have a relational database. Why did we want a relational database? Well, because we knew that we wanted to make a social application and we would have some relations between the accounts and the posts, with comments and replies etc. MySQL was also the database we had the most experience with, so it was easier for us to use MySQL than other relational databases like PostgreSQL, or MariaDB.

Referential Integrity

We used referential integrity with all the foreign keys to ensure some constraints take place.

```
CREATE TABLE Account
(
    id         INT AUTO_INCREMENT,
    ...
);

CREATE TABLE Post
(
    ...
    FOREIGN KEY (accountId) REFERENCES Account (id)
        ON UPDATE CASCADE ON DELETE NO ACTION
);
```

For example we used `ON UPDATE CASCADE` so that when we update an account, the posts created by that account will have the `accountId` changed accordingly. Other than that, we used `ON DELETE NO ACTION` so that when we delete an account, the posts created by that account will not change, in order to maintain that post, even after account deletion.

With MySQL, we used different database objects throughout our project like views, indexes, temporary tables, stored procedures and functions, triggers, scheduled events, and transactions.

View

We used virtual tables, namely the views, to only show specific columns of the Account table for each of the different types of users available in the system.

We created 3 types of users:

1. **Basic**
2. **Developer**
3. **Admin**

The **Basic User** can only see the data and cannot modify it and we made a view for the basic user with the account details, but without any personal information like `email` or `password`.

The **Developer User** can also modify the data from the fields, so we created a view with all the account details, including the personal information and the `active` field.

The **Admin User** can do what the developer user does, and can grant permission to other users. For the view of the admin user we decided to show all the privileges of the existing users.

Index

We used the indexes for the Account table to speed up the search of a specific account based on the first and last name.

Temporary Table

We use a temporary table in a stored procedure that makes a statistic for the number of likes, comments and shares of a specific post based on the given account id. After the creation, the temporary table remains active only for the current session.

Stored Procedure and Function

The stored procedures and functions are objects that we used multiple times to reuse some code. We used stored functions to get the number of likes, shares and comments of a specific post, where we returned a variable that counts each appearance of the aforementioned elements for a specific post. The stored procedures were used in multiple places, like at the creation of the welcome post, when an account is created, or to disable an account by making the account's `active` column from `true` into `false`, when an account was deactivated by the user.

Trigger

We used triggers for a welcome post that is automatically created when a new account is created. Basically, we insert a post immediately after inserting a new account. In addition, we also used a trigger for friend request validation, in order to make sure that if someone sent a friend request, the receiver can only accept or reject it, but not send also a friend request to the sender. If that was the case, we would send an error message. Moreover, we have also checked that the friend request sender id and the friend request receiver id cannot be the same.

Scheduled Event

For scheduled events, we came up with the idea to make a birthday post. Therefore, we chose to make the scheduled event to check daily for the birth dates of the active accounts. In order to find the birth date of the account we used a formula where we checked if the current date equals the date of birth to which we modified only the year by adding the difference between the current year and the date of birth year of each account.

Transaction

We used a transaction for the add friend request validation, to cancel the error messages that may occur if the request receiver wants to send a friend request back to the request sender or if the request receiver and sender account id is the same. Therefore, if error messages occurs, then the transaction cancels the error message and rollbacks the creation of the friend request.

F. Operations, Use Cases, Code

The possible operations identified in our database include, but are not limited, to: basic CRUD operations for all the base tables, procedures for more complex operations like adding a friend request, disabling an account, but not deleting it, viewing the number of likes, shares and comments a post has, as well as an overview of the posts of an account.

The following are a number of use cases we have created out of these operations, and below them the corresponding code is written:

1. Making an Account

```
INSERT INTO Account (firstName, lastName, email, password,
birthDate, gender)
VALUES ('Paul', 'Panaitescu', 'paul000@gmail.com', 'pass12',
'1996-10-7', 'male');
```

2. Making a Post

```
INSERT INTO Post (accountId, content, sponsored)
VALUES (1, 'Hello everyone!', 1);
```

3. Sending a Friend Request

```
CALL add_friend_request(0, 1);
```

4. Accepting a Friend Request

```
UPDATE Friendship SET pending = false WHERE receiverAccountId = 1
AND senderAccountId = 0;
```

5. Rejecting a Friend Request

```
UPDATE Friendship SET pending = false, denied = true WHERE
receiverAccountId = 1 AND senderAccountId = 0;
```

6. **Commenting on a Friend’s Post**

```
INSERT INTO Comment(postId, accountId, content) VALUES (1, 1,
'Hello there!');
```

7. **Liking a Friend’s Post**

```
INSERT INTO PostLikes(postId, accountId) VALUES (1, 1);
```

8. **Replying to a Friend’s Comment**

```
INSERT INTO Reply(commentId, accountId, content) VALUES (1, 1,
'Hello again!');
```

9. **Joining a Group**

```
INSERT INTO GroupParticipants(socialGroupId, accountId) VALUES (0,
4);
```

10. **Viewing Posting Activity and Statistics of an Account**

```
CALL get_posts_of(1);
```

G. Rationale for Choices

In this project we had many choices to make, many of them were already described in this file:

1. Normalization
    - We made sure to convert the tables to cover the 3 normalization forms.

2. Bridge Tables
    - In order to represent the many-to-many relationship, we used a bridge table. For example: GroupParticipants table has `socialGroupId`, and `accountId` as foreign keys, to show that an account is part of a specific social group.

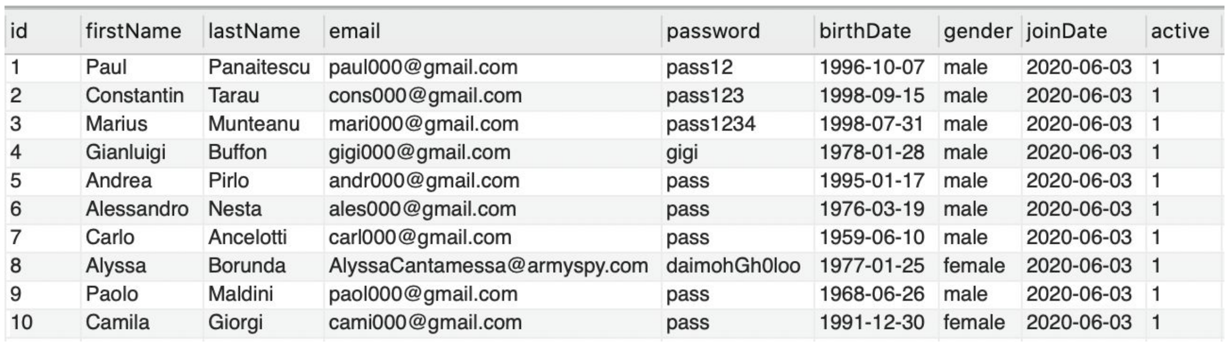
3. Referential Integrity
    - Update. For Example: when updating an account, the `accountId` of the posts that were created by that account may suffer changes.
    - Delete. For Example: when deleting an account, the posts that were created by that account will remain the same, with no change to their `accountId.`

4. Users
    - Privileges. We assigned privileges like `SELECT, INSERT, UPDATE, DELETE`, or `ALL`, to certain roles. Other than that, we used `WITH GRANT OPTION` to allow a user to grant permissions to other users.
    - Roles. We assigned roles to users, so that the users can have the proper privileges.

II. Screenshots of the Database

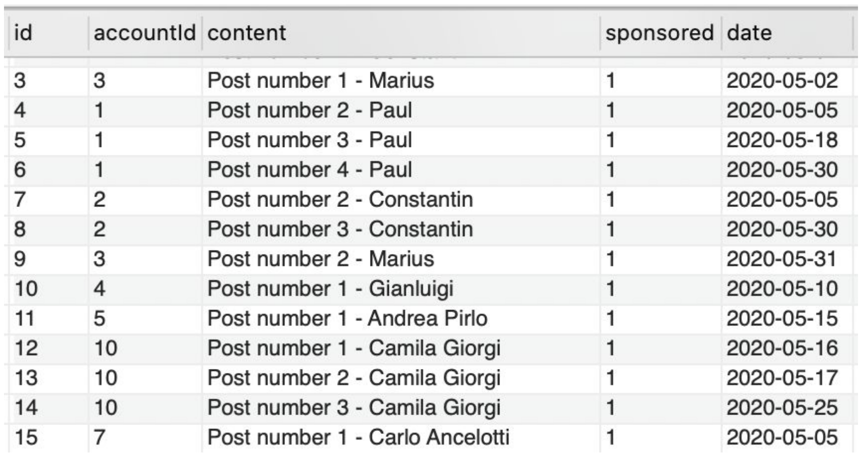
Base Tables

Account



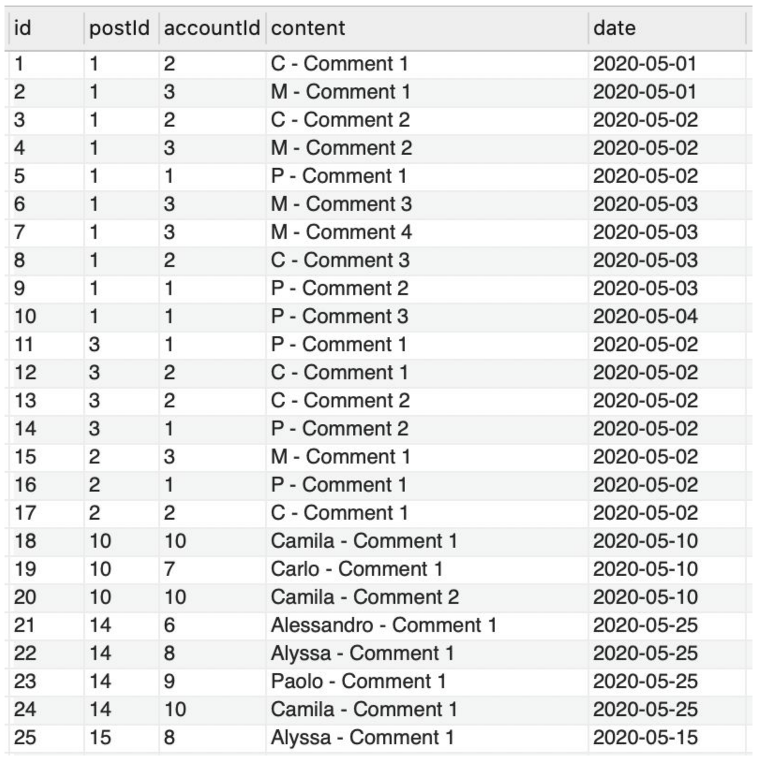
id	firstName	lastName	email	password	birthDate	gender	joinDate	active
1	Paul	Panaitescu	paul000@gmail.com	pass12	1996-10-07	male	2020-06-03	1
2	Constantin	Tarau	cons000@gmail.com	pass123	1998-09-15	male	2020-06-03	1
3	Marius	Munteanu	mari000@gmail.com	pass1234	1998-07-31	male	2020-06-03	1
4	Gianluigi	Buffon	gigi000@gmail.com	gigi	1978-01-28	male	2020-06-03	1
5	Andrea	Pirlo	andr000@gmail.com	pass	1995-01-17	male	2020-06-03	1
6	Alessandro	Nesta	ales000@gmail.com	pass	1976-03-19	male	2020-06-03	1
7	Carlo	Ancelotti	carl000@gmail.com	pass	1959-06-10	male	2020-06-03	1
8	Alyssa	Borunda	AlyssaCantamessa@armyspy.com	daimohGh0loo	1977-01-25	female	2020-06-03	1
9	Paolo	Maldini	paol000@gmail.com	pass	1968-06-26	male	2020-06-03	1
10	Camila	Giorgi	cami000@gmail.com	pass	1991-12-30	female	2020-06-03	1

Post



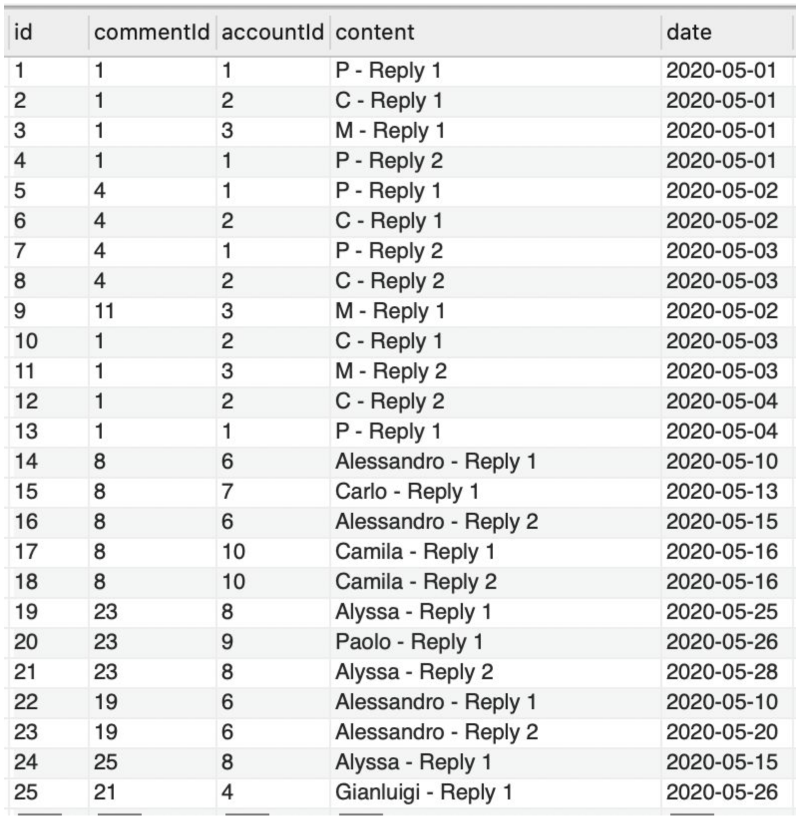
id	accountId	content	sponsored	date
3	3	Post number 1 - Marius	1	2020-05-02
4	1	Post number 2 - Paul	1	2020-05-05
5	1	Post number 3 - Paul	1	2020-05-18
6	1	Post number 4 - Paul	1	2020-05-30
7	2	Post number 2 - Constantin	1	2020-05-05
8	2	Post number 3 - Constantin	1	2020-05-30
9	3	Post number 2 - Marius	1	2020-05-31
10	4	Post number 1 - Gianluigi	1	2020-05-10
11	5	Post number 1 - Andrea Pirlo	1	2020-05-15
12	10	Post number 1 - Camila Giorgi	1	2020-05-16
13	10	Post number 2 - Camila Giorgi	1	2020-05-17
14	10	Post number 3 - Camila Giorgi	1	2020-05-25
15	7	Post number 1 - Carlo Ancelotti	1	2020-05-05

Comment



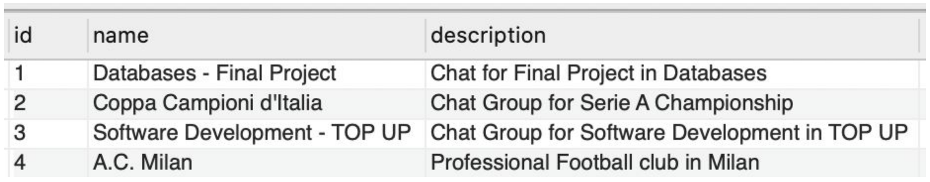
id	postId	accountId	content	date
1	1	2	C - Comment 1	2020-05-01
2	1	3	M - Comment 1	2020-05-01
3	1	2	C - Comment 2	2020-05-02
4	1	3	M - Comment 2	2020-05-02
5	1	1	P - Comment 1	2020-05-02
6	1	3	M - Comment 3	2020-05-03
7	1	3	M - Comment 4	2020-05-03
8	1	2	C - Comment 3	2020-05-03
9	1	1	P - Comment 2	2020-05-03
10	1	1	P - Comment 3	2020-05-04
11	3	1	P - Comment 1	2020-05-02
12	3	2	C - Comment 1	2020-05-02
13	3	2	C - Comment 2	2020-05-02
14	3	1	P - Comment 2	2020-05-02
15	2	3	M - Comment 1	2020-05-02
16	2	1	P - Comment 1	2020-05-02
17	2	2	C - Comment 1	2020-05-02
18	10	10	Camila - Comment 1	2020-05-10
19	10	7	Carlo - Comment 1	2020-05-10
20	10	10	Camila - Comment 2	2020-05-10
21	14	6	Alessandro - Comment 1	2020-05-25
22	14	8	Alyssa - Comment 1	2020-05-25
23	14	9	Paolo - Comment 1	2020-05-25
24	14	10	Camila - Comment 1	2020-05-25
25	15	8	Alyssa - Comment 1	2020-05-15

Reply



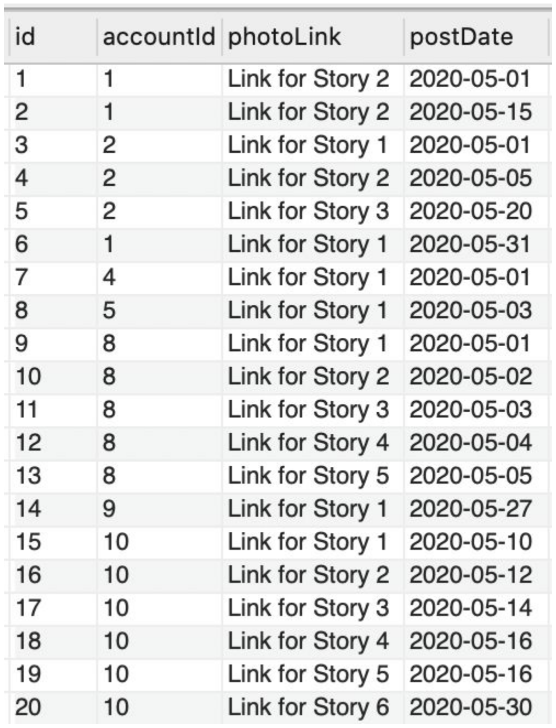
id	commentId	accountId	content	date
1	1	1	P - Reply 1	2020-05-01
2	1	2	C - Reply 1	2020-05-01
3	1	3	M - Reply 1	2020-05-01
4	1	1	P - Reply 2	2020-05-01
5	4	1	P - Reply 1	2020-05-02
6	4	2	C - Reply 1	2020-05-02
7	4	1	P - Reply 2	2020-05-03
8	4	2	C - Reply 2	2020-05-03
9	11	3	M - Reply 1	2020-05-02
10	1	2	C - Reply 1	2020-05-03
11	1	3	M - Reply 2	2020-05-03
12	1	2	C - Reply 2	2020-05-04
13	1	1	P - Reply 1	2020-05-04
14	8	6	Alessandro - Reply 1	2020-05-10
15	8	7	Carlo - Reply 1	2020-05-13
16	8	6	Alessandro - Reply 2	2020-05-15
17	8	10	Camila - Reply 1	2020-05-16
18	8	10	Camila - Reply 2	2020-05-16
19	23	8	Alyssa - Reply 1	2020-05-25
20	23	9	Paolo - Reply 1	2020-05-26
21	23	8	Alyssa - Reply 2	2020-05-28
22	19	6	Alessandro - Reply 1	2020-05-10
23	19	6	Alessandro - Reply 2	2020-05-20
24	25	8	Alyssa - Reply 1	2020-05-15
25	21	4	Gianluigi - Reply 1	2020-05-26

SocialGroup



id	name	description
1	Databases - Final Project	Chat for Final Project in Databases
2	Coppa Campioni d'Italia	Chat Group for Serie A Championship
3	Software Development - TOP UP	Chat Group for Software Development in TOP UP
4	A.C. Milan	Professional Football club in Milan

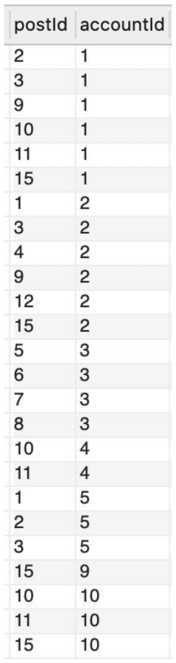
Story



id	accountId	photoLink	postDate
1	1	Link for Story 2	2020-05-01
2	1	Link for Story 2	2020-05-15
3	2	Link for Story 1	2020-05-01
4	2	Link for Story 2	2020-05-05
5	2	Link for Story 3	2020-05-20
6	1	Link for Story 1	2020-05-31
7	4	Link for Story 1	2020-05-01
8	5	Link for Story 1	2020-05-03
9	8	Link for Story 1	2020-05-01
10	8	Link for Story 2	2020-05-02
11	8	Link for Story 3	2020-05-03
12	8	Link for Story 4	2020-05-04
13	8	Link for Story 5	2020-05-05
14	9	Link for Story 1	2020-05-27
15	10	Link for Story 1	2020-05-10
16	10	Link for Story 2	2020-05-12
17	10	Link for Story 3	2020-05-14
18	10	Link for Story 4	2020-05-16
19	10	Link for Story 5	2020-05-16
20	10	Link for Story 6	2020-05-30

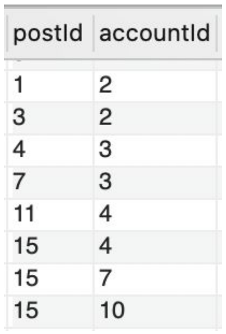
Bridge Tables

PostLikes



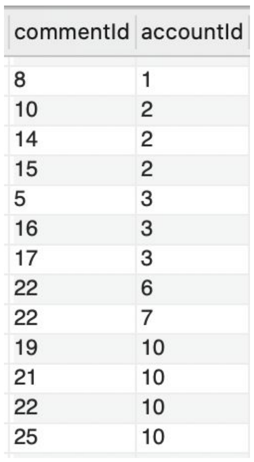
postId	accountId
2	1
3	1
9	1
10	1
11	1
15	1
1	2
3	2
4	2
9	2
12	2
15	2
5	3
6	3
7	3
8	3
10	4
11	4
1	5
2	5
3	5
15	9
10	10
11	10
15	10

PostShares



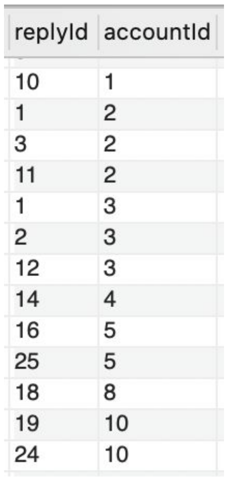
postId	accountId
1	2
3	2
4	3
7	3
11	4
15	4
15	7
15	10

CommentLikes



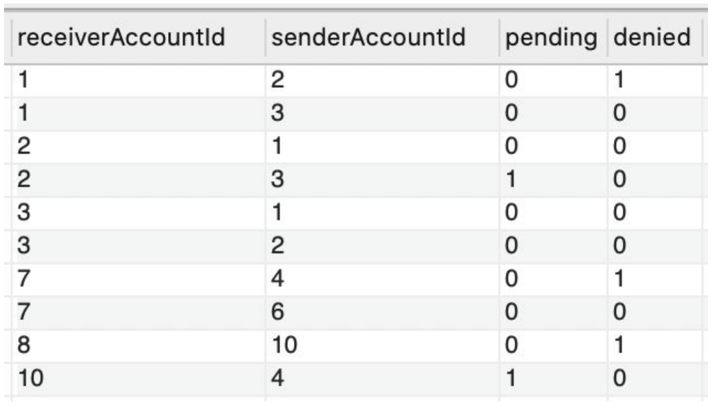
commentId	accountId
8	1
10	2
14	2
15	2
5	3
16	3
17	3
22	6
22	7
19	10
21	10
22	10
25	10

ReplyLikes



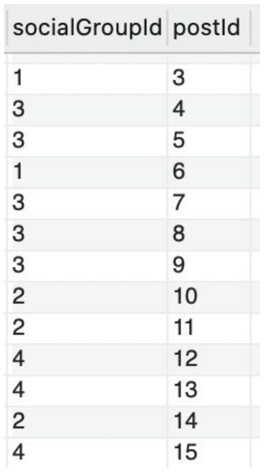
replyId	accountId
10	1
1	2
3	2
11	2
1	3
2	3
12	3
14	4
16	5
25	5
18	8
19	10
24	10

Friendship



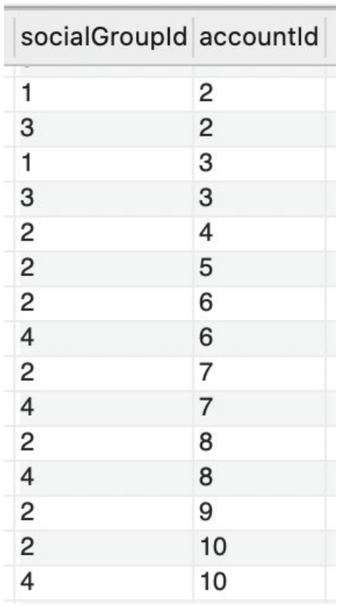
receiverAccountId	senderAccountId	pending	denied
1	2	0	1
1	3	0	0
2	1	0	0
2	3	1	0
3	1	0	0
3	2	0	0
7	4	0	1
7	6	0	0
8	10	0	1
10	4	1	0

GroupPosts



socialGroupId	postId
1	3
3	4
3	5
1	6
3	7
3	8
3	9
2	10
2	11
4	12
4	13
2	14
4	15

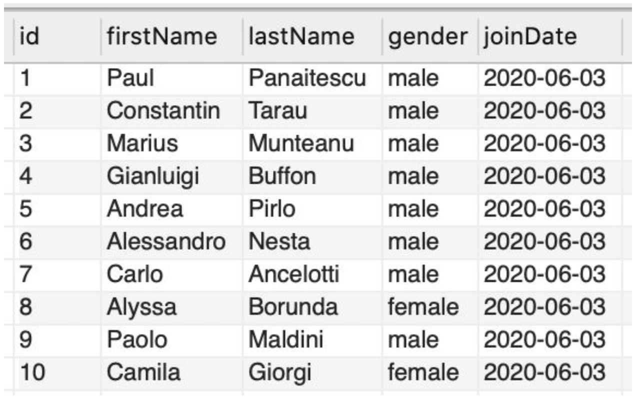
GroupParticipants



socialGroupId	accountId
1	2
3	2
1	3
3	3
2	4
2	5
2	6
4	6
2	7
4	7
2	8
4	8
2	9
2	10
4	10

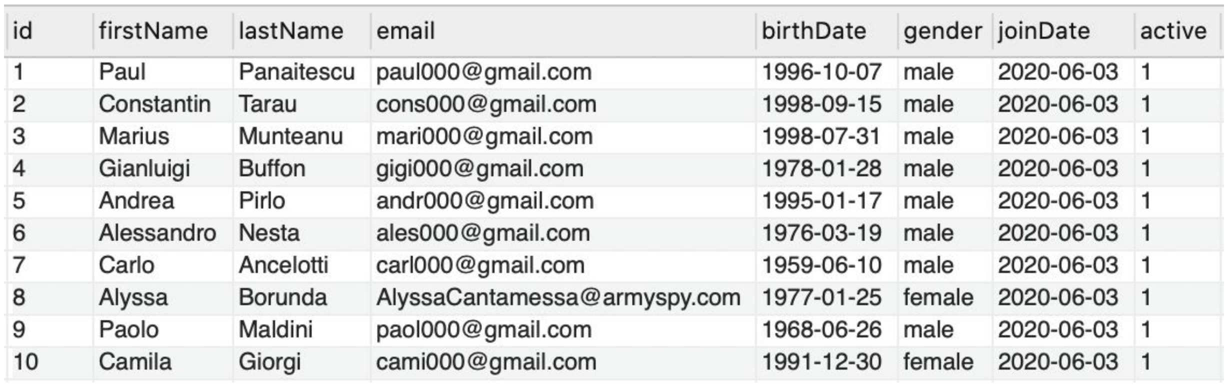
Views

Basic User



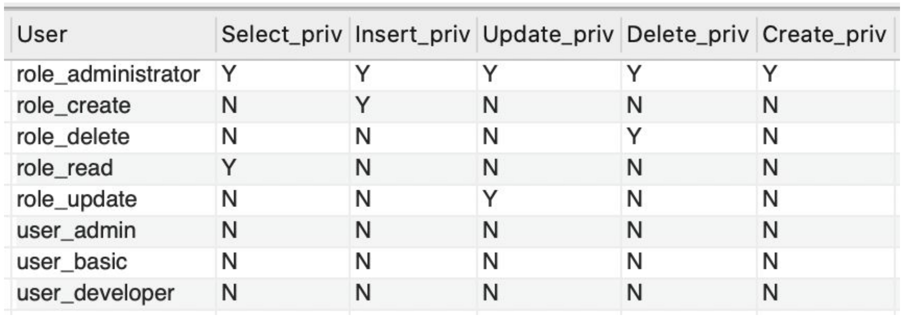
id	firstName	lastName	gender	joinDate
1	Paul	Panaitescu	male	2020-06-03
2	Constantin	Tarau	male	2020-06-03
3	Marius	Munteanu	male	2020-06-03
4	Gianluigi	Buffon	male	2020-06-03
5	Andrea	Pirlo	male	2020-06-03
6	Alessandro	Nesta	male	2020-06-03
7	Carlo	Ancelotti	male	2020-06-03
8	Alyssa	Borunda	female	2020-06-03
9	Paolo	Maldini	male	2020-06-03
10	Camila	Giorgi	female	2020-06-03

Developer User



id	firstName	lastName	email	birthDate	gender	joinDate	active
1	Paul	Panaitescu	paul000@gmail.com	1996-10-07	male	2020-06-03	1
2	Constantin	Tarau	cons000@gmail.com	1998-09-15	male	2020-06-03	1
3	Marius	Munteanu	mari000@gmail.com	1998-07-31	male	2020-06-03	1
4	Gianluigi	Buffon	gigi000@gmail.com	1978-01-28	male	2020-06-03	1
5	Andrea	Pirlo	andr000@gmail.com	1995-01-17	male	2020-06-03	1
6	Alessandro	Nesta	ales000@gmail.com	1976-03-19	male	2020-06-03	1
7	Carlo	Ancelotti	carl000@gmail.com	1959-06-10	male	2020-06-03	1
8	Alyssa	Borunda	AlyssaCantamessa@armyspy.com	1977-01-25	female	2020-06-03	1
9	Paolo	Maldini	paol000@gmail.com	1968-06-26	male	2020-06-03	1
10	Camila	Giorgi	cami000@gmail.com	1991-12-30	female	2020-06-03	1

Admin User



User	Select_priv	Insert_priv	Update_priv	Delete_priv	Create_priv
role_administrator	Y	Y	Y	Y	Y
role_create	N	Y	N	N	N
role_delete	N	N	N	Y	N
role_read	Y	N	N	N	N
role_update	N	N	Y	N	N
user_admin	N	N	N	N	N
user_basic	N	N	N	N	N
user_developer	N	N	N	N	N

III. Appendix - SQL Queries (separate file)