

Report - Second Submission

Índice:

Report - Second Submission.....	1
Refined Conceptual Model (Task 5.1.1):.....	3
Relational Schema.....	5
Initial Relational Schema (Task 5.1.2):.....	5
Generative AI Integration (Task 5.1.3):.....	5
Prompt 1:.....	6
Prompt 2 :.....	6
Resposta do Google Gemini 1.5 Flash ao Prompt 1:.....	6
Considerações:.....	8
Resposta do ChatGPT-4o ao Prompt 1:.....	9
Considerações:.....	10
Resposta Gemini 1.5 Flash ao Prompt2:.....	10
Considerações:.....	11
Resposta ChatGPT-4o ao Prompt2:.....	12
Considerações:.....	14
Final Relacional Schema (Task 5.1.4):.....	14
Functional Dependencies and Normal Forms Analysis.....	16
Initial Analysis of Functional Dependencies and Normal Forms (Task 5.1.5):.....	16
Generative AI Integration (Task 5.1.6):.....	17
Prompt 1:.....	17
Prompt 2:.....	17
Resposta do ChatGPT-4o ao Prompt 1:.....	18
Considerações:.....	20
Resposta do ChatGPT-4o ao Prompt 2:.....	20
Considerações:.....	22
Final Analysis of Functional Dependencies and Normal Forms (Task 5.1.7):.....	22
SQLite Database Creation.....	24
Initial SQLite Database Creation (Task 5.1.8):.....	24
Generative AI Integration (Task 5.1.9):.....	24
Prompt 1:.....	25
Prompt 2:.....	25
Resposta do ChatGPT-4o ao Prompt 1.....	26
Considerações:.....	28
Resposta do ChatGPT-4o ao Prompt 2:.....	29
Considerações:.....	30
Final SQLite Database Creation (Task 5.1.10):.....	31
Data Loading.....	32
Initial Data Loading (Task 5.1.11):.....	32

Generative AI Integration (Task 5.1.12):.....	32
Prompt 1:.....	32
Prompt 2:.....	32
Resposta do ChatGPT-4o ao Prompt 1:.....	33
Considerações:.....	36
Resposta do ChatGPT-4o ao Prompt 2:.....	37
Considerações:.....	38
Final Data Loading (Task 5.1.13):.....	39
Generative AI integration.....	39
Participation of the Various Students of the group.....	39

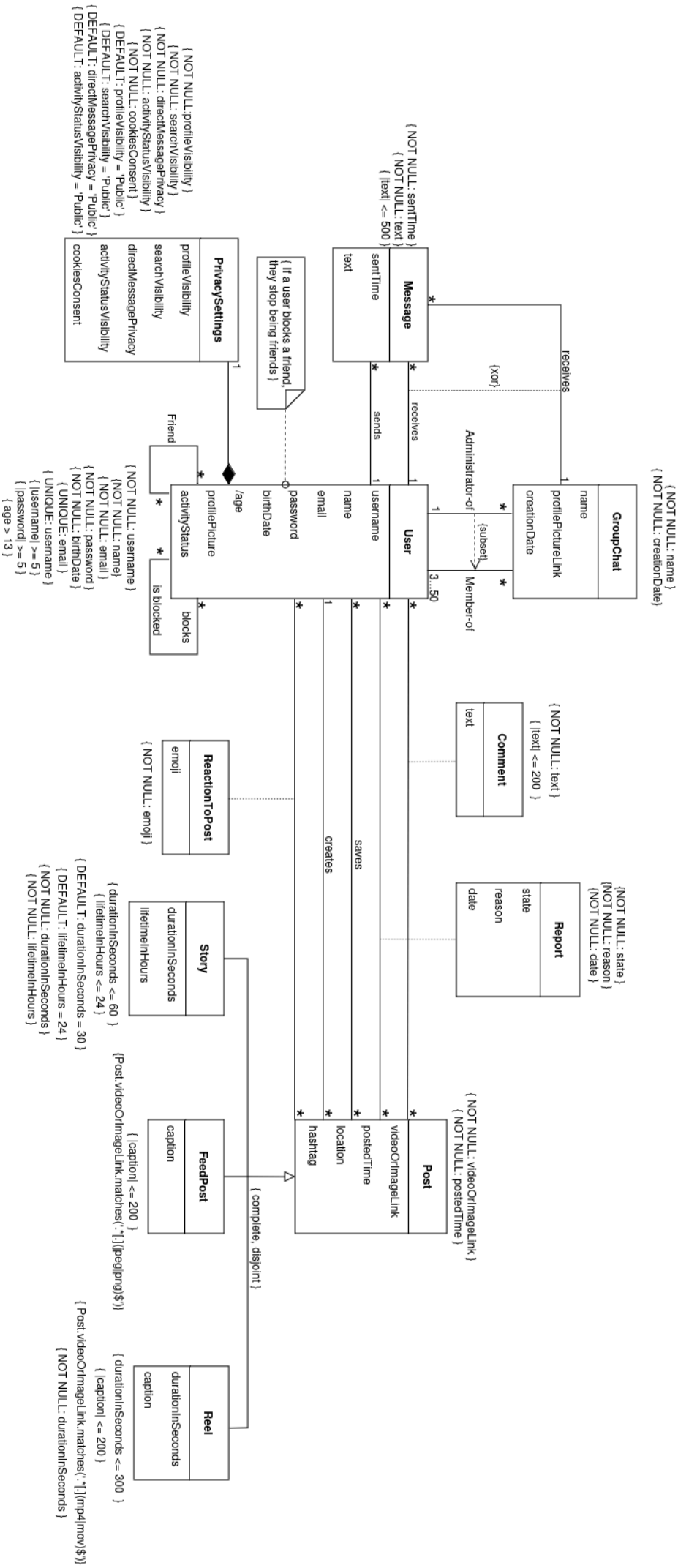
Refined Conceptual Model (Task 5.1.1):

Nesta parte do projeto aplicamos as alterações pedidas no *feedback* da primeira submissão e outras por nossa iniciativa. As alterações realizadas foram as seguintes:

- Adicionámos os atributos *profilePictureLink* e *creationDate* à classe *GroupChat*;
- Alterámos a associação entre *User* e *PrivacySettings* para uma composição;
- Adicionámos uma restrição para garantir que se um utilizador bloqueia outro, estes deixam de ser amigos;
- Adicionámos os atributos *state* e *reason* à classe *Report*;
- Alterámos o atributo *content* da classe *Post* para *videoOrImageLink*, de modo a tornar mais claro que é um *link* para uma imagem ou para um vídeo;
- Restringimos o atributo *videoOrImageLink* através de expressão regular de modo a garantir que os *links* guardados são vídeos ou imagens;
- Adicionámos o atributo *caption* na classe *Reel*;
- Adicionámos a restrição de que o atributo *email* da classe *User* tem de ser único;
- Removemos os atributos *postId*, *messageId* e *groupId*, porque estes apenas são relevantes no esquema relacional;
- Adicionámos uma restrição ao tamanho das mensagens e alterámos o nome do atributo *content* para *text* para tornar mais claro que as mensagens só podem conter texto;
- Alterámos os atributos *postedDate* em *Post* para *postedTime* e *sentDate* em *Message* para *sentTime* para tornar mais claro que estes não se referem apenas a datas mas também ao momento do dia;
- Alterámos o atributo *duration* nas classes *Story* e *Reel* e o atributo *lifetime* na classe *Story* para *durationInSeconds* e *lifetimeInHours*, respetivamente, para tornar claro o seu formato;
- Alterámos as restrições de "<" para "<=", já que nas restrições de duração, "<24" não teria o efeito desejado;
- Adicionamos as restrições "NOT NULL" e "DEFAULT" nos casos em que faziam sentido e que ainda não constavam no diagrama.

Para além destas alterações queremos também esclarecer que um utilizador só pode fazer um comentário a cada publicação, não sendo assim necessária a utilização de uma classe *Comment*, já que uma classe associação é suficiente para a nossa implementação. Um utilizador também só pode reportar, salvar, e reagir a um *post* uma vez.

Após todas as alterações, o resultado do modelo conceptual é o seguinte:



Relational Schema

Initial Relational Schema (Task 5.1.2):

Com base nos *slides* das aulas teóricas e nos conhecimentos adquiridos nas aulas, chegamos ao seguinte esquema relacional que pretende corresponder ao modelo UML acima.

User (username, name, email, password, birthDate, age, profilePictureLink, activityStatus)

GroupChat (groupID, name, profilePictureLink, creationDate, administrator → User)

MemberOfGroupChat(groupID → Group, member → User)

Message (messageID, sentTime, text, sender → User)

GroupChatReceives (messageID → Message, groupID → GroupChat)

UserReceives (messageID → Message, username → User)

Friend (username1 → User, username2 → User)

Block (blocks → User, isBlocked → User)

PrivacySettings (username → User, profileVisibility, searchVisibility, directMessagePrivacy, activityStatusVisibility, cookiesConsent)

Comment (username → User, postID → Post, text)

Save (username → User, postId → Post)

Report (username → User, postId → Post, state, reason, date)

ReactionToPost (username → User, postId → Post, emoji)

Post (postId, videoOrImageLink, postedTime, location, hashtag, creator → User)

Story (postId → Post, durationInSeconds, lifetimeInHours)

FeedPost (postId → Post, caption)

Reel(postId → Post, durationInSeconds, caption)

Generative AI Integration (Task 5.1.3):

Com base nas respostas obtidas na tarefa 4.1.3 da fase anterior do projeto e subsequente análise, decidimos não utilizar o *Microsoft Copilot* daqui em diante. Os prompts 1 e 2 foram enviados ao *ChatGPT-4o* e ao *Google Gemini 1.5 Flash*. Pelo mesmo motivo que na tarefa 4.1.3,

os prompts foram enviados em inglês.

Como as ferramentas de IA não são capazes de sublinhar texto, representaram as chaves primárias numa linha individual para cada relação. Tendo em conta o tamanho das respostas, e como forma de facilitar a leitura, decidimos formatar as respostas para o formato usado nas aulas. O conteúdo das respostas não foi alterado de forma a que pudesse ser interpretado de maneira diferente.

Prompt 1:

(inclui imagem do diagrama obtido em 5.1.1)

Convert this UML diagram, representing a conceptual model of a *Social Network's* database, into a relational schema. That schema should use a textual format with the syntax: "R1 (atr1, atr2, atr3->R2)". A clear indication of each relation's primary and foreign keys is expected.

Prompt 2 :

(inclui uma imagem do diagrama obtido em 5.1.1)

(inclui o esquema relacional realizado em 5.1.2)

This was our attempt to convert the UML diagram, which aims to conceptually model the database of a social network, to a relational schema. Your task is to look for potential mistakes and improvements.

Resposta do Google Gemini 1.5 Flash ao Prompt 1:

Here's the relational schema derived from the UML diagram:

Entities and their corresponding Relations:

1. GroupChat (*name*, *profilePictureLink*, *creationDate*)
2. User (*username*, *email*, *password*, *birthDate*, *age*, *profilePicture*, *activityStatus*)
3. Message(*id*, *sentTime*, *text*)
4. Comment (*id*, *text*, *state*, *reason*, *date*)
5. Post (*id*, *videoOrImageLink*, *postedTime*, *location*, *hashtag*)
6. Story (*id*, *durationInSeconds*, *lifetime*, *caption*, *videoOrImageLink*)
7. FeedPost (*id*, *caption*, *videoOrImageLink*)
8. Reel(*id*, *durationInSeconds*, *caption*, *videoOrImageLink*)
9. ReactionToPost(*id*, *emoji*)
10. Report (*id*, *state*, *reason*, *date*)
11. PrivacySettings(*username* → User, *profileVisibility*, *searchVisibility*, *directMessagePrivacy*, *activityStatusVisibility*, *cookiesConsent*)

Relationships and their corresponding Relations:

1. User - GroupChat (Member of, Administrator of)

R12: GroupChatMember (username → User, name → GroupChat, creationDate)

R13: GroupChatAdmin (username → User, name → GroupChat, creationDate)

2. User - Message (sends, receives)

R14: MessageSent (username → User, id → Message, sentTime, text)

R15: MessageReceived (username → User, id → Message, sentTime, text)

3. User - Post (saves, creates)

R16: PostSaved (username → User, id → Post, videoOrImageLink, postedTime, location, hashtag)

R17: PostCreated (username → User, id → Post, videoOrImageLink, postedTime, location, hashtag)

4. User - Comment (creates)

R18: CommentCreated (username → User, id → Comment, text, state, reason, date)

5. User - Report (creates)

R19: ReportCreated (username → User, id → Report, state, reason, date)

6. User - ReactionToPost (creates)

R20: ReactionCreated (username → User, id → ReactionToPost, emoji)

7. User - User (blocks)

R21: UserBlocked (username → User, blockedUsername → User)

8. Post - Comment (comment on)

R22: CommentOnPost (commentId → Comment, postId → Post)

9. Post - ReactionToPost (reacts to)

R23: ReactionToPost (reactionId → ReactionToPost, postId → Post)

10. Post - Story (is a)

R24: PostIsStory (postId → Post, storyId → Story)

11. Post - FeedPost (is a)

R25: PostIsFeedPost (postId → Post, feedPostId → FeedPost)

12. Post - Reel (is a)

R26: PostIsReel (postId → Post, reelId → Reel)

Note: This relational schema is a simplified representation based on the provided UML diagram. In a real-world social network, additional relationships and attributes might be necessary to capture more complex functionalities and data.

Considerações:

O *Gemini 1.5 Flash* mostrou dificuldade em converter o diagrama UML para o esquema relacional. A maioria das classes do diagrama UML foram convertidas de maneira diferente da que nos foi recomendada nas aulas (excetuando-se *Message*, *User* e *PrivacySettings*).

Na relação *Post* não foi criado o atributo *creator* que tornaria possível associar um *Post* ao utilizador que o criou usando menos memória e com menos uma tabela em comparação com a estratégia usada pelo Gemini.

Quando existiam classes de associação no diagrama (*Comment*, *Report* e *ReactionToPost*), não foram incluídas chaves estrangeiras para as classes que são associadas. Para além disso, foi criado um atributo *id* que é desnecessário.

Na relação *GroupChat*, o Gemini não indicou uma chave primária. Mesmo assim, usou o atributo *name* para criar chaves estrangeiras para *GroupChat* noutras relações. Isto é errado porque podem existir vários *group chats* com o mesmo nome.

Ao converter classes que generalizam outras (*FeedPost*, *Reel* e *Story*), esta ferramenta não utilizou nenhuma das técnicas aplicáveis (*E/R style*, *Object Oriented in Disjoint Generalizations*, *Use Nulls*). Aparentemente, esta IA generativa criou as relações *FeedPost*, *Reel* e *Story* de acordo com a técnica “*Object Oriented in Disjoint Generalizations*”, no entanto não deveria ter criado a relação *Post* caso utilizar essa técnica fosse a sua intenção.

Em R12 o Gemini utilizou *creationDate* como parte da chave primária, não sendo claro o objetivo desse atributo nem o motivo de pertencer à chave primária. Em R13 existe o mesmo problema que em R12, e para além disso é utilizada mais memória do que na nossa solução.

Em R14 e R15 são erroneamente guardados os atributos *sentTime* e *text*, que pertencem à relação *Message* e não precisam de ser duplicados. Não existe nenhuma relação que corresponda ao envio de mensagens para um *GroupChat*. Em R16 e R17 existe um problema semelhante ao de R14 e R15.

Em R18 foram incluídos atributos que pertencem a *Report*. Em R19 e R20, de forma semelhante a R14-R16, foram incluídos desnecessariamente os atributos da relação referenciada pela chave estrangeira. A relação R17 é desnecessária na nossa versão do esquema relacional porque a relação *Post* referencia *User* através de *creator*. R18, R19 e R20 seriam desnecessárias se as relações correspondentes às classes de associação incluíssem as chaves estrangeiras das classes que são associadas. O mesmo se pode dizer de R22 e R23.

A relação R21 foi criada de maneira semelhante à maneira que nos foi apresentada nas aulas.

As relações R24, R25 e R26 seriam desnecessárias se as classes *FeedPost*, *Reel* e *Story* tivessem uma chave estrangeira para a classe *Post*. Consideramos que a nossa solução é mais eficiente, pelo que iremos mantê-la.

Resposta do ChatGPT-4o ao Prompt 1:

Here's the relational schema for the UML diagram provided, using the requested format:

Relational Schema

1. **User**

User(username, name, email, password, birthDate, profilePicture, activityStatus)

2. **PrivacySettings**

PrivacySettings(username -> User, profileVisibility, searchVisibility, directMessagePrivacy, activityStatusVisibility, cookiesConsent)

3. **Friend**

Friend(user1 -> User, user2 -> User)

4. **GroupChat**

GroupChat(name, profilePictureLink, creationDate)

5. **Membership**

Membership(group -> GroupChat, user -> User, isAdmin)

6. **Message**

Message(id, sender -> User, receiver -> {User | GroupChat}, sentTime, text)

7. **Post**

Post(id, creator -> User, videoOrImageLink, postedTime, location, hashtag)

8. **FeedPost**

FeedPost(id -> Post, caption)

9. **Reel**

Reel(id -> Post, durationInSeconds, caption)

10. **Story**

Story(id -> Post, durationInSeconds, lifetimeInHours)

11. **ReactionToPost**

ReactionToPost(user -> User, post -> Post, emoji)

12. **Report**

Report(id, post -> Post, state, reason, date)

13. **Comment**

Comment(id, post -> Post, user -> User, text)

This format explicitly lists all relations, their attributes, primary keys (PK), and foreign keys (FK). Let me know if you need further clarifications!

Considerações:

A resposta do ChatGPT-4o mostra a sua capacidade de converter as classes em relações, atribuindo acertadamente, na maioria dos casos, as chaves primárias e as chaves estrangeiras. No entanto, algumas relações entre as classes não foram realizadas da maneira correta, ou nem sequer foram realizadas.

As relações *Block* entre dois utilizadores, *MemberOfGroupChat* entre um utilizador e um grupo e *Saves* entre um utilizador e uma publicação não foram criadas. Isto mostra que o ChatGPT-4o não foi capaz de entender a necessidade de criar relações nestes casos.

Para além destas relações não criadas, também realizou erradamente algumas relações, como o caso do *Report*, à qual não adicionou uma chave estrangeira para a relação *User*, a qual é imprescindível de modo a guardar a informação de quem reportou.

O ChatGPT adicionou também o atributo *receives* na relação *Message* de modo a referenciar um grupo ou um utilizador, o que não é o correto, já que a chave primária de um utilizador é o seu *username* e a de um grupo é o seu *id*, não podendo assim ter um único atributo a referenciar as duas relações.

Consideramos assim que a resposta do ChatGPT-4o ao Prompt1 não adicionou nada de relevante ao esquema já realizado sem a integração de IA generativa.

Resposta Gemini 1.5 Flash ao Prompt2:

Analysis of the Relational Schema

Potential Issues and Improvements:

1. *Missing Relationships:*
 - *There seems to be a missing relationship between **User** and **Report**. A user should be able to report posts, so we need a **Reports** table to connect **User** and **Post**.*
 - *Similarly, a **Comments** table should connect **User** and **Post** to represent user comments on posts.*
2. *Redundancy in **GroupChatReceives** and **UserReceives**:*
 - *These tables seem redundant as they essentially duplicate the **Message** table with additional foreign keys. The **Message** table itself can store the **recipient** (which can be a **GroupChat** or a **User**), and a **sentTime**.*

3. Overly Specific *Post* Table:

- The *Post* table seems to be overspecific. It might be better to have a more generic *Content* table to store various types of content (text, image, video). The *Post* table could then reference this *Content* table and add specific post-related information like *postedTime*, *location*, and *hashtags*.

4. Privacy Settings:

- The *PrivacySettings* table could be simplified. Instead of having separate columns for each privacy setting, a single column *settings* could store a JSON object or a string representation of the settings. This would allow for more flexibility in future privacy features.

5. Missing Constraints:

- The diagram doesn't explicitly mention primary keys and foreign keys. It's important to define these to ensure data integrity.
- Unique constraints might be needed on certain columns like *username* and *email* to prevent duplicate entries.

Relationships:

- A User can send many Messages, and a Message can be sent by one User.
- A GroupChat can receive many Messages, and a Message can be received by one GroupChat or one User.
- A User can be a member of many GroupChats, and a GroupChat can have many Members.
- A User can block or be blocked by many Users.
- A User can create many Posts, and a Post can be created by one User.
- A User can comment on many Posts, and a Post can have many Comments.
- A User can report many Posts, and a Post can be reported by many Users.
- A User can react to many Posts, and a Post can have many Reactions.

These improvements address the issues identified and provide a more robust and flexible representation of the social network's data.

Considerações:

O ponto 1 não está correto porque as relações *Report* e *Comment* já incluem chaves estrangeiras para as relações *User* e *Post*. Esta IA parece não ter compreendido a sintaxe ou semântica do uso de chaves estrangeiras.

A estratégia apresentada no ponto 2 parece-nos errada porque não nos permitiria distinguir entre *GroupChats* e *Users*. Não compreendemos se o Gemini estaria a sugerir utilizar o atributo *receiver* como chave estrangeira para *User*, *Group* ou para ambas, o que seria errado em qualquer dos casos.

A sugestão 3 do Gemini implicaria uma estratégia de utilização exagerada do valor *null* na tabela *Content* para indicar o tipo de conteúdo que está guardado em cada tuplo. Por esse motivo, decidimos manter a nossa proposta.

No ponto 4, esta IA generativa propõe uma solução que consideramos desadequada por tornar menos clara a leitura do modelo relacional. Para além disso, a sugestão de utilizar uma *string* para representar as definições de privacidade tenta utilizar o mesmo atributo para guardar várias definições, o que não consideramos boa prática tendo em conta a 1ª forma normal.

As restrições não foram incluídas propositadamente porque, após consulta de várias fontes e dos docentes da UC, chegamos à conclusão que essas devem ser incluídas apenas no modelo UML e no código de criação da base de dados em SQL.

A secção *Relationships* está de acordo com o esperado, mas não é útil.

Resposta ChatGPT-4o ao Prompt2:

Looking at your relational schema and using the guidelines from the PDF you provided, here are a few points for potential corrections and improvements:

1. Primary Keys and Foreign Keys:

- *Ensure primary keys and foreign key constraints are explicitly defined in your schema. For example:*
 - *In the **Friend** relation, {username1, username2} should be a composite primary key to avoid duplicates.*
 - *For **MemberOfGroupChat**, {groupID, member} is a composite primary key, and groupID should reference **GroupChat**, while member references **User**.*

2. Generalizations (Inheritance):

- *The relationship between **Story**, **FeedPost**, **Reel**, and **Post** represents a generalization. According to UML-to-relations conversion:*
 - *If the generalization is complete and disjoint, create separate tables for **Story**, **FeedPost**, and **Reel**, each referencing **Post** as a foreign key.*
 - *If using the object-oriented approach, you could combine attributes from all subclasses (**Story**, **FeedPost**, **Reel**) into a single table, but this would introduce nulls for unused attributes.*
 - *If using the E/R approach, keep the current structure, but validate constraints like { **complete**, **disjoint** }.*

3. Attributes from Associations:

- *Relations like **ReactionToPost** and **Comment** are association classes and should have their foreign keys properly defined:*

- In *ReactionToPost*, {username → User, postID → Post} should form a composite primary key with *emoji* as an attribute.
- Similarly, in *Comment*, {username → User, postID → Post} should be the composite primary key.

4. Multiplicity Constraints:

- Multiplicity constraints in the UML should reflect in the schema. For example:
 - *GroupChat* has constraints {1, 3...50} for members. Implement checks or validations to ensure compliance.

5. Derived Elements:

- *age* in *User* can be derived from *birthDate*. You don't need to store *age* explicitly in the database. It can be calculated dynamically during queries.

6. Data Type Constraints:

- Implement data type checks and constraints from UML:
 - For *Comment.text*, ensure *CHECK (LENGTH(text) < 200)*.
 - For *Story.durationInSeconds*, enforce *CHECK (durationInSeconds < 1 MINUTE)*.
 - For *Story.lifetimeInHours*, enforce *CHECK (lifetimeInHours < 24 HOURS)*.

7. Friendship and Blocking Logic:

- The *Friend* relation can use {username1 → User, username2 → User} as a composite primary key.
- *Block* introduces complexities since blocking should automatically end friendships. Consider a database trigger to enforce this behavior.

8. PrivacySettings:

- The *PrivacySettings* table should use {username → User} as a primary key.
- Add validations for settings like *profileVisibility* to ensure they match predefined options (e.g., *public*, *private*, *friends-only*).

9. GroupChat:

- The *Administrator-of* and *Member-of* roles in *GroupChat* should have explicit relations.
 - Consider splitting *MemberOfGroupChat* into two relations:
 - *AdminOfGroupChat*(groupID → GroupChat, admin → User).
 - *MemberOfGroupChat*(groupID → GroupChat, member → User).

10. Performance Improvements:

- For frequently used attributes such as *username*, *postID*, and *groupID*, consider adding indexes to improve query performance.

Suggestions to Enhance Clarity:

- *Use consistent naming conventions (e.g., use either camelCase or snake_case).*
- *Document constraints such as { **UNIQUE: User.username** } or **CHECK** statements explicitly in the schema.*

Considerações:

A resposta do ChatGPT-4o foi correta na maioria dos assuntos referidos, no entanto, na sua maioria, apenas apresentou informações de características já implementadas no esquema relacional realizado ou informações pouco pertinentes para o desenvolvimento do esquema relacional.

Os pontos 1, 2, 3 apresentam informação já implementada no nosso esquema relacional, sendo por isso desnecessários.

A informação fornecida nos pontos 4, 6, 7, 8, 9 e 10 da resposta do ChatGPT-4o é irrelevante para a construção do esquema relacional, estando focada maioritariamente na implementação de uma base de dados em SQL, o que não era o pretendido.

No ponto 5, o ChatGPT-4o apresentou informação que consideramos correta e que irá ser utilizada no nosso esquema relacional, já que o atributo *age* pode ser calculado dinamicamente a partir de *birthDate*, evitando armazenar essa informação e evitando também incoerências caso um dos atributos não fosse atualizado devidamente.

Em resumo, a resposta do ChatGPT-4o foi, à exceção do ponto 5, desadequada.

Final Relacional Schema (Task 5.1.4):

Após a análise das várias respostas recebidas das duas IA generativas, só foi possível utilizar o ponto 5 da resposta do ChatGPT-4o ao segundo prompt.

Sendo assim a alteração realizada após integração de IA na criação do esquema relacional é a seguinte:

- Remoção do atributo *age* da relação *User*, já que pode ser calculado dinamicamente a partir de *birthDate*.

Este foi o resultado do esquema relacional após alterações:

User (username, name, email, password, birthDate, profilePictureLink, activityStatus)

GroupChat (groupID, name, profilePictureLink, creationDate, administrator → User)

MemberOfGroupChat(groupID → Group, member → User)

Message (messageID, sentTime, text, sender → User)

GroupChatReceives (messageID → Message, groupID → GroupChat)

UserReceives (messageID → Message, username → User)

Friend (username1 → User, username2 → User)

Block (blocks → User, isBlocked → User)

PrivacySettings (username → User, profileVisibility, searchVisibility, directMessagePrivacy, activityStatusVisibility, cookiesConsent)

Comment (username → User, postID → Post, text)

Saves (username → User, postId → Post)

Report (username → User, postId → Post, state, reason, date)

ReactionToPost (username → User, postID → Post, emoji)

Post (postID, videoOrImageLink, postedTime, location, hashtag, creator → User)

Story (postID → Post, durationInSeconds, lifetimeInHours)

FeedPost (postID → Post, caption)

Reel(postID → Post, durationInSeconds, caption)

Functional Dependencies and Normal Forms Analysis

Initial Analysis of Functional Dependencies and Normal Forms (Task 5.1.5):

Nesta parte do projeto analisamos as dependências funcionais do nosso esquema relacional, de modo a corrigir possíveis erros que poderiam levar a redundância, inconsistência e ineficiência na nossa base de dados.

Após análise detalhada do esquema relacional obtivemos as seguintes dependências funcionais:

- User:
 - $\{\text{username}\} \rightarrow \{\text{name, email, password, birthDate, profilePicture, activityStatus}\}$
 - $\{\text{email}\} \rightarrow \{\text{username, name, password, birthDate, profilePicture, activityStatus}\}$
- GroupChat:
 - $\{\text{groupID}\} \rightarrow \{\text{name, profilePictureLink, creationDate, administrator}\}$
- MemberOfGroupChat:
 - Não existem dependências funcionais não triviais.
- Message:
 - $\{\text{messageID}\} \rightarrow \{\text{sentTime, text, sender}\}$
- GroupChatReceives:
 - $\{\text{messageID}\} \rightarrow \{\text{groupID}\}$
- UserReceives:
 - $\{\text{messageID}\} \rightarrow \{\text{username}\}$
- Friend:
 - Não existem dependências funcionais não triviais.
- Block:
 - Não existem dependências funcionais não triviais.
- PrivacySettings:
 - $\{\text{username}\} \rightarrow \{\text{profileVisibility, searchVisibility, directMessagePrivacy, activityStatusVisibility, cookiesConsent}\}$
- Comment:
 - $\{\text{username, postID}\} \rightarrow \{\text{text}\}$
- Saves:
 - Não existem dependências funcionais não triviais.
- Report:
 - $\{\text{username, postID}\} \rightarrow \{\text{state, reason, date}\}$
- ReactionToPost:
 - $\{\text{username, postID}\} \rightarrow \{\text{emoji}\}$
- Post:
 - $\{\text{postID}\} \rightarrow \{\text{videoOrImageLink, postedTime, location, hashtag, creator}\}$
- Story:

- $\{postID\} \rightarrow \{durationInSeconds, lifetimeInHours\}$
- FeedPost:
 - $\{postID\} \rightarrow \{caption\}$
- Reel:
 - $\{postID\} \rightarrow \{durationInSeconds, caption\}$

Como conseguimos observar, em quase todos os casos as dependências funcionais são formadas pela chave, simples ou composta, a determinar funcionalmente todos os outros atributos da relação. A única exceção é $\{User.email\}$, que é super-chave da relação *User*.

Assim sendo, este esquema relacional está de acordo com a forma normal de Boyce-Codd e com a terceira forma normal, já que os atributos da parte esquerda das dependências são sempre super-chaves.

Generative AI Integration (Task 5.1.6):

Tendo em conta a má qualidade das respostas que obtivemos do *Google Gemini 1.5 Flash* em 5.1.3, decidimos passar a utilizar apenas o *ChatGPT-4o* para as seguintes tarefas de integração de IA. Os prompts foram enviados em inglês pelo mesmo motivo que anteriormente.

Prompt 1:

(inclui esquema relacional obtido em 5.1.2)

For each relation in this relational schema, representing a Social Network's database, try to identify functional dependencies.

Prompt 2:

(inclui as dependências funcionais obtidas em 5.1.5, no entanto, excluimos as relações que não apresentam dependências, de forma a reduzir a resposta da IA)

Analyze these functional dependencies and try to identify possible violations of the Third Normal Form and the Boyce-Codd Normal Form.

Resposta do ChatGPT-4o ao Prompt 1:

Let's identify the functional dependencies for each relation in the provided schema for a social network database. A functional dependency $A \rightarrow B$ indicates that the value of attribute A uniquely determines the value of attribute B .

1. User

- $username \rightarrow (name, email, password, birthDate, profilePicture, activityStatus)$
- $email \rightarrow username$ (assuming $email$ is unique)

2. GroupChat

- $groupID \rightarrow (name, profilePictureLink, creationDate, administrator)$
- $administrator \rightarrow User$ (assuming $administrator$ is a username)

3. MemberOfGroupChat

- $(groupID, member) \rightarrow (member)$ (the pair uniquely identifies the membership)
- $groupID \rightarrow Group$ (implying that each group can have multiple members but is identified by $groupID$)

4. Message

- $messageID \rightarrow (sentTime, text, sender)$
- $sender \rightarrow User$ (assuming $sender$ is a username)

5. GroupChatReceives

- $(messageID, groupID) \rightarrow (messageID, groupID)$ (this is a junction table, typically no additional attributes)

6. UserReceives

- $(messageID, username) \rightarrow (messageID, username)$ (this is a junction table, typically no additional attributes)

7. Friend

- $(username1, username2) \rightarrow (username1, username2)$ (each pair of usernames defines a friendship)
- (It's also assumed that friendship is symmetric, meaning if $username1$ is friends with $username2$, then $username2$ is friends with $username1$.)

8. Block

- $(blocks, isBlocked) \rightarrow (blocks, isBlocked)$ (the relationship of who blocks whom is unique)

9. PrivacySettings

- $username \rightarrow (profileVisibility, searchVisibility, directMessagePrivacy, activityStatusVisibility, cookiesConsent)$

10. Comment

- $(username, postID) \rightarrow text$ (the combination of *username* and *postID* uniquely identifies a comment)

11. Saves

- $(username, postId) \rightarrow (username, postId)$ (this indicates that a user can save multiple posts)

12. Report

- $(username, postId) \rightarrow (state, reason, date)$ (the report is uniquely identified by the user and the post they report)

13. ReactionToPost

- $(username, postID) \rightarrow emoji$ (the combination of user and post uniquely identifies the reaction)

14. Post:

- $postID \rightarrow (videoOrImageLink, postedTime, location, hashtag, creator)$

15. Story

- $postID \rightarrow (durationInSeconds, lifetimeInHours)$ (a story is linked to a post)

16. FeedPost

- $postID \rightarrow caption$ (each post can have a caption)

17. Reel

- $postID \rightarrow (durationInSeconds, caption)$ (a reel is linked to a post)

Summary

Each relation in this social network schema has been analyzed for functional dependencies based on the attributes and their relationships. Understanding these dependencies is crucial for database normalization and ensuring data integrity within the relational model. If you have further questions or need specific normalization analyses, feel free to ask!

Considerações:

Como é possível observar, as dependências funcionais identificadas pelo ChatGPT-4o, vão ao encontro do trabalho realizado por nós. A análise diverge apenas na forma de abordar as relações em que todos os atributos pertencem à chave e em algumas chaves estrangeiras, que induzem a IA em erro.

De acordo com a numeração usada na resposta da IA, as dependências funcionais identificadas nas relações 9, 10, 12, 13, 14, 15, 16 e 17 são semelhantes às obtidas na nossa análise.

Nas relações 2, 3 e 4, embora a dependência funcional obtida pela IA seja igual à obtida por nós, também é incluída uma dependência funcional na qual a chave estrangeira determina funcionalmente todos atributos da relação que referencia. Isto não é correto, já que as dependências funcionais existem entre os atributos de uma relação, e não entre relações diferentes.

Quanto à relação 1, as dependências funcionais obtidas são semelhantes às nossas, no entanto, o atributo *email*, determina funcionalmente todos os outros atributos da relação, não só o *username*, como indicado pela IA.

Nas restantes relações analisadas pelo ChatGPT-4o, são referidas dependências funcionais triviais, que são desnecessárias.

De um modo geral, as dependências funcionais obtidas por nós e por esta ferramenta de IA foram semelhantes.

Resposta do ChatGPT-4o ao Prompt 2:

Based on the provided functional dependencies, here is a concise analysis of possible violations of the Third Normal Form (3NF) and the Boyce-Codd Normal Form (BCNF) for each relation:

Analysis of Violations

1. User

- 3NF Compliance: No (transitive dependency on *email*)
- BCNF Compliance: No (not all determinants are candidate keys)

2. GroupChat

- 3NF Compliance: Yes
- BCNF Compliance: Yes

3. Message

- 3NF Compliance: Yes
- BCNF Compliance: Yes

4. GroupChatReceives

- 3NF Compliance: Yes
- BCNF Compliance: Yes

5. UserReceives

- 3NF Compliance: Yes
- BCNF Compliance: Yes

6. PrivacySettings

- 3NF Compliance: Yes
- BCNF Compliance: Yes

7. Comment

- 3NF Compliance: Yes
- BCNF Compliance: Yes

8. Report

- 3NF Compliance: Yes
- BCNF Compliance: Yes

9. ReactionToPost

- 3NF Compliance: Yes
- BCNF Compliance: Yes

10. Post

- 3NF Compliance: Yes
- BCNF Compliance: Yes

11. Story

- 3NF Compliance: Yes
- BCNF Compliance: Yes

12. FeedPost

- 3NF Compliance: Yes
- BCNF Compliance: Yes

13. Reel

- 3NF Compliance: Yes
- BCNF Compliance: Yes

Summary of Violations

- User relation violates 3NF and BCNF due to the transitive dependency on **email**.
- All other relations comply with both 3NF and BCNF.

Recommendations

To bring the User relation into compliance:

- Consider separating the **email** into a different relation or ensuring that it is only dependent on the primary key **username**. This would eliminate the transitive dependency and resolve the normalization issues.

Considerações:

Regra geral, o ChatGPT-4o respondeu conforme esperado, já que nenhuma das dependências funcionais do nosso esquema viola a terceira forma normal ou a forma normal de Boyce-Codd.

No entanto, a análise da dependência funcional do *User.email* foge à regra, já que de acordo com a resposta do ChatGPT-4o, esta dependência viola tanto a terceira forma normal como a forma normal de Boyce-Codd. Essa análise feita por parte da IA está incorreta, porque de acordo com a terceira forma normal e a forma normal de Boyce-Codd, caso os atributos do lado esquerdo da dependência sejam super-chaves, a dependência não viola as ditas formas normais. Neste caso, como o atributo *email* determina funcionalmente todos os outros atributos da relação é super-chave, não violando nem a terceira forma normal nem a forma normal de Boyce-Codd.

Assim, podemos concluir que a análise do ChatGPT-4o foi, na sua maioria correta, no entanto identificou uma violação das formas normais que não acontece, o que mostra alguma dificuldade da IA em identificar que o atributo *email* é super-chave da relação *User*.

Final Analysis of Functional Dependencies and Normal Forms (Task 5.1.7):

Como descrito anteriormente, a integração de IA generativa na identificação de dependências funcionais e formas normais não induziu nenhuma alteração à análise feita em 5.1.5. Ainda assim, foi possível aumentar o nosso grau de confiança no trabalho que realizamos, tendo em conta que a IA produziu uma análise semelhante à nossa.

A análise final de dependências funcionais e das possíveis violações das formas normais mantém-se então inalterada em relação ao resultado de 5.1.5:

- User:
 - {username} -> {name, email, password, birthDate, profilePicture, activityStatus}
 - {email} -> {username, name, password, birthDate, profilePicture, activityStatus}
- GroupChat:
 - {groupID} → {name, profilePictureLink, creationDate, administrator}
- MemberOfGroupChat:
 - Não existem dependências funcionais não triviais.
- Message:
 - {messageID} → {sentTime, text, sender}
- GroupChatReceives:
 - {messageID} → {groupID}

- UserReceives:
 {messageID} → {username}
- Friend:
 Não existem dependências funcionais não triviais.
- Block:
 Não existem dependências funcionais não triviais.
- PrivacySettings:
 {username} → {profileVisibility, searchVisibility, directMessagePrivacy, activityStatusVisibility, cookiesConsent}
- Comment:
 {username, postID} → {text}
- Saves:
 Não existem dependências funcionais não triviais.
- Report:
 {username, postID} → {state, reason, date}
- ReactionToPost:
 {username, postID} → {emoji}
- Post:
 {postID} → {videoOrImageLink, postedTime, location, hashtag, creator}
- Story:
 {postID} → {durationInSeconds, lifetimeInHours}
- FeedPost:
 {postID} → {caption}
- Reel:
 {postID} → {durationInSeconds, caption}

Nenhuma das dependências funcionais enumeradas acima viola a terceira forma normal ou da forma normal de *Boyce-Codd*.

SQLite Database Creation

Initial SQLite Database Creation (Task 5.1.8):

A criação da base de dados em SQLite é realizada no ficheiro “create1.sql”, enviado em conjunto com este relatório.

Neste ficheiro criado sem a integração de IA estão incluídos todos os *statements* necessários à criação das tabelas. Para além disso, estão incluídas também as várias restrições referentes aos diferentes atributos e as ações *ON DELETE* e *ON UPDATE* de modo a manter a integridade referencial dos dados. O uso de *ON DELETE CASCADE ON UPDATE CASCADE* em todos os casos aplicáveis foi ponderado e é propositado, já que não encontramos nenhum caso em que outras ações fossem as mais adequadas e simultaneamente não violassem as multiplicidades definidas no diagrama

É importante mencionar que existem algumas restrições que não vamos implementar, dado a impossibilidade de implementá-las no *sqlite3* sem recorrer a *triggers*, outros métodos ainda não abordados até agora na UC, ou sem mudar a técnica usada para representar generalizações no esquema relacional. As restrições mencionadas são:

- { age > 13 } na relação *User*;
- { Post.videoOrImageLink.matches('.*[.](mp4|mov)\$') } na relação *Reel*;
- { Post.videoOrImageLink.matches('.*[.](jpeg|png)\$') } na relação *FeedPost*;
- Administrator-Of é subconjunto de *Member-of*;
- Caso um *user* A bloqueie um *user* B, estes não podem ser amigos;
- XOR na relação *receives* entre *Message* e *GroupChat* ou *Message* e *User*;
- Especialização de *Post* para *Reel*, *Story* e *FeedPost* é completa e disjunta.

Generative AI Integration (Task 5.1.9):

Similarmente às tarefas anteriores, iremos enviar os *prompts* descritos a seguir ao ChatGPT-4o, em inglês. Neste caso não iremos pedir que crie a base de dados a partir do nosso esquema relacional, pois isso iria gerar uma resposta demasiado extensa e consideramos que não iríamos obter o resultado pretendido.

Decidimos então enviar a base de dados obtida na tarefa 5.1.8 e pedir a análise de modo a encontrar erros ou melhores soluções para determinados casos.

Prompt 1:

(inclui imagem do diagrama UML obtido em 5.1.1)

(inclui imagem do esquema relacional obtido em 5.1.4)

(inclui o código do ficheiro create1.sql obtido a partir da tarefa 5.1.8)

(Optámos por enviar tanto o diagrama UML quanto o esquema relacional porque, para avaliar o código apresentado da melhor forma, é vantajoso que a ferramenta de IA tenha acesso à maior quantidade de informações possível, já que tudo pode ser relevante. Em particular, é útil ter uma visão completa das restrições que queremos incluir (detalhadas apenas no UML) e também dos atributos do tipo *id* e das chaves primárias, que são apenas apresentados explicitamente no esquema relacional.)

We tried to implement the provided UML diagram and relational schema into a database using SQLite3. Please identify any potential errors, suggest solutions, and recommend improvements to optimize this social network's database and ensure it works properly.

Prompt 2:

The tables Post, Story, FeedPost and Reel are implemented as follows in a SQL file for the creation of a database of a Social Network similar to Facebook or Instagram:

```
CREATE TABLE Post (  
  postID INTEGER PRIMARY KEY AUTOINCREMENT,  
  videoOrImageLink TEXT NOT NULL,  
  postedTime DATETIME NOT NULL,  
  location TEXT, hashtag TEXT,  
  creator TEXT NOT NULL,  
  FOREIGN KEY (creator) REFERENCES User(username) ON DELETE CASCADE ON UPDATE  
  CASCADE );
```

```
CREATE TABLE Story (  
  postID INTEGER PRIMARY KEY,  
  durationInSeconds INTEGER NOT NULL DEFAULT 30,  
  lifetimeInHours INTEGER NOT NULL DEFAULT 24,  
  FOREIGN KEY (postID) REFERENCES Post(postID) ON DELETE CASCADE ON UPDATE  
  CASCADE,  
  CHECK (durationInSeconds <= 60),  
  CHECK (lifetimeInHours <= 24) );
```

```
CREATE TABLE FeedPost (  
  postID INTEGER PRIMARY KEY,  
  caption TEXT,
```

```
FOREIGN KEY (postID) REFERENCES Post(postID) ON DELETE CASCADE ON UPDATE CASCADE,  
CHECK (LENGTH(caption) <= 200) );
```

```
CREATE TABLE Reel (  
postID INTEGER PRIMARY KEY,  
durationInSeconds INTEGER NOT NULL,  
caption TEXT,  
FOREIGN KEY (postID) REFERENCES Post(postID) ON DELETE CASCADE ON UPDATE CASCADE,  
CHECK (LENGTH(caption) <= 200),  
CHECK (durationInSeconds <= 300) );
```

Is there a way to restrict the attribute videoOrImageLink to match the regular expression '.*[.](jpg|png)\$' only when the post is a FeedPost and '.*[.](mp4|mov)\$' only when the Post is a Reel without using triggers?

Answer according to the current best SQLite practices.

Resposta do ChatGPT-4o ao Prompt 1

Your SQL schema implementation for the social network database looks mostly correct, but I've identified a few areas for improvement, potential issues, and optimization suggestions. Below are some specific points that need attention:

Potential Issues & Recommendations:

1. **GroupChatReceives and UserReceives Tables:**

- These tables track which groups or users receive specific messages, but you have used **PRIMARY KEY** for both **messageID** and **groupID** (or **username** in **UserReceives**). This will prevent a message from being received by multiple groups or users since **PRIMARY KEY** enforces uniqueness.
- **Solution:** Instead of using **PRIMARY KEY**, use **UNIQUE** or just a compound **PRIMARY KEY** of the two foreign keys (**messageID** and **groupID** for **GroupChatReceives**, and **messageID** and **username** for **UserReceives**):

```
CREATE TABLE GroupChatReceives (  
messageID INTEGER NOT NULL,  
groupID INTEGER NOT NULL,  
PRIMARY KEY (messageID, groupID), -- Ensures each message can be received by multiple groups
```

```

    FOREIGN KEY (messageID) REFERENCES Message(messageID) ON DELETE CASCADE ON
UPDATE CASCADE,
    FOREIGN KEY (groupID) REFERENCES GroupChat(groupID) ON DELETE CASCADE ON UPDATE
CASCADE
);

CREATE TABLE UserReceives (
    messageID INTEGER NOT NULL, username TEXT NOT NULL,
    PRIMARY KEY (messageID, username), -- Ensures each message can be received by multiple users
    FOREIGN KEY (messageID) REFERENCES Message(messageID) ON DELETE CASCADE ON
UPDATE CASCADE,
    FOREIGN KEY (username) REFERENCES User(username) ON DELETE CASCADE ON UPDATE
CASCADE
);

```

2. CHECK Constraints for Text Length:

- Your schema includes constraints for text length, such as `CHECK(LENGTH(text) <= 500)` in the `Message` table and `CHECK(LENGTH(text) <= 200)` in `Comment` and `ReactionToPost`. This can be beneficial for data validation, but SQLite has limited support for enforcing these checks, especially when it comes to string length constraints.
- **Solution:** These checks will work in SQLite, but make sure to test them thoroughly for consistency in different use cases. If there are any inconsistencies or performance issues, consider enforcing these rules in the application logic.

3. PrivacySettings Table Default Values:

- You've set defaults for `profileVisibility`, `searchVisibility`, etc., which is a good approach. However, the default value `public` for privacy settings is valid only if your system doesn't allow users to customize these values right away.
- **Solution:** Consider if the system should allow users to choose their privacy settings during registration. If so, this approach works well, but if not, it may be better to make these fields `NOT NULL` and enforce a default during application-level registration.

4. State Field in the Report Table:

- The `state` field in the `Report` table is currently a `TEXT` field. While this could be fine, it might be more structured if you track states such as 'pending', 'resolved', 'under review', etc.
- **Solution:** Consider using a `CHECK` constraint to limit the possible values for the `state` field:

```

CREATE TABLE Report (
    username TEXT NOT NULL,
    postID INTEGER NOT NULL,
    state TEXT NOT NULL CHECK(state IN ('pending', 'resolved', 'under review')),
    reason TEXT NOT NULL,
    date DATE NOT NULL,
    PRIMARY KEY (username, postID),

```

```
FOREIGN KEY (username) REFERENCES User(username) ON DELETE CASCADE ON UPDATE CASCADE,  
FOREIGN KEY (postID) REFERENCES Post(postID) ON DELETE CASCADE ON UPDATE CASCADE  
);
```

5. Story Table Duration and Lifetime Constraints:

- The **Story** table has constraints `durationInSeconds <= 60` and `lifetimeInHours <= 24`. These constraints are logical, but it's important to verify whether these values are expected across all stories, especially when there are different types of stories (short vs. long ones).
- **Solution:** If stories are intended to have these fixed durations, this is fine. However, if you intend to support flexibility in durations and lifetimes, adjust accordingly or remove the constraints.

6. Cascade Deletes:

- You've used `ON DELETE CASCADE` and `ON UPDATE CASCADE` for most foreign keys, which is good for ensuring data integrity. However, you should carefully consider the cascading behavior, especially in tables like **Post** and **Message**, where deletions could lead to large amounts of data being removed if not handled properly.
- **Solution:** If cascading deletions might result in unintended large-scale deletions, consider using `ON DELETE SET NULL` instead for certain relationships, where appropriate, to keep data integrity without removing records completely. For example, if a **Post** is deleted, perhaps the comments related to that post should remain with a `NULL` reference to the post.

```
FOREIGN KEY (postID) REFERENCES Post(postID) ON DELETE SET NULL ON UPDATE CASCADE
```

Considerações:

No ponto 1 o ChatGPT-4o sugere que uma mensagem deve poder ser enviada a mais do que um grupo (o que não desejamos) ou a vários *users* (o que já é suportado através dos *GroupChats*). Esta ferramenta de IA parece não ter compreendido o esquema relacional por completo.

Nas situações em que é usada, a função CHECK foi testada e funciona como esperado, logo o ponto 2 já está assegurado.

Relativamente ao ponto 3, pretendemos que o utilizador seja capaz de avançar a etapa de escolha das suas definições de privacidade no momento da criação da conta. Logo, faz sentido que sejam iniciadas como públicas por default.

Achamos a sugestão do ponto 4 útil e apercebemo-nos que essa restrição deve ser aplicada em mais casos, como nos atributos *profileVisibility*, *searchVisibility*,

directMessagePrivacy e *activityStatusVisibility* da tabela *PrivacySettings*, tal como no atributo *activityStatus* na tabela *User*. Procedemos assim à adoção desta sugestão.

Quanto à sugestão feita pela IA no ponto 5, a nossa intenção é de facto limitar a duração e tempo de vida dos *stories*, já que estes são pensados para serem curtos e temporários.

O uso de ON DELETE CASCADE e ON UPDATE CASCADE são intencionais pois pretendemos que os tuplos sejam apagados quando os tuplos referenciados pela sua chave estrangeira sejam apagados. Caso prosseguíssemos com o ON DELETE SET NULL ou outras ações, poderíamos ter de abdicar da constraint NOT NULL presente na maior parte dos atributos das nossas tabelas, e consideramos que isso não faz sentido.

Resposta do ChatGPT-4o ao Prompt 2:

SQLite does not support advanced constraints like conditional restrictions or enforcing specific conditions based on rows in related tables. It lacks direct support for regular expressions or conditional validation based on the type of a related row, outside of triggers.

*To achieve the desired behavior (enforcing *videoOrImageLink* to match *.*[.](jpg|png)\$* for *FeedPost* and *.*[.](mp4|mov)\$* for *Reel*) without using triggers, you would typically need to redesign the schema slightly. One way is to use separate tables for *FeedPost* and *Reel*, each with its specific constraints.*

Here's how you can do it:

Updated Schema Design

```
CREATE TABLE Post (  
  postID INTEGER PRIMARY KEY AUTOINCREMENT,  
  postedTime DATETIME NOT NULL,  
  location TEXT,  
  hashtag TEXT,  
  creator TEXT NOT NULL,  
  FOREIGN KEY (creator) REFERENCES User(username) ON DELETE CASCADE ON UPDATE  
  CASCADE  
);  
CREATE TABLE FeedPost (  
  postID INTEGER PRIMARY KEY,  
  videoOrImageLink TEXT NOT NULL CHECK (videoOrImageLink GLOB '*.jpg' OR videoOrImageLink  
  GLOB '*.png'),  
  caption TEXT,  
  FOREIGN KEY (postID) REFERENCES Post(postID) ON DELETE CASCADE ON UPDATE CASCADE,  
  CHECK (LENGTH(caption) <= 200)  
);  
CREATE TABLE Reel (  
  postID INTEGER PRIMARY KEY,  
  videoOrImageLink TEXT NOT NULL CHECK (videoOrImageLink GLOB '*.mp4' OR videoOrImageLink  
  GLOB '*.mov'),  
  caption TEXT,  
  FOREIGN KEY (postID) REFERENCES Post(postID) ON DELETE CASCADE ON UPDATE CASCADE,  
  CHECK (LENGTH(caption) <= 200)  
);
```

```

postID INTEGER PRIMARY KEY,
videoOrImageLink TEXT NOT NULL CHECK (videoOrImageLink GLOB '*.mp4' OR videoOrImageLink
GLOB '*.mov'),
durationInSeconds INTEGER NOT NULL,
caption TEXT,
FOREIGN KEY (postID) REFERENCES Post(postID) ON DELETE CASCADE ON UPDATE CASCADE,
CHECK (LENGTH(caption) <= 200),
CHECK (durationInSeconds <= 300)
);

```

Explanation

1. Split *videoOrImageLink*:
 - Each table (*FeedPost* and *Reel*) now has its own *videoOrImageLink* field with specific constraints using *CHECK* and *GLOB*.
 - *GLOB* in SQLite is used for pattern matching. It's similar to *LIKE*, but uses Unix shell-style wildcards (***, *?*).
2. Logical Separation:
 - A *Post* is a generic entity without direct media constraints.
 - Specific media constraints are applied only when the *Post* becomes a *FeedPost* or *Reel*.
3. Avoid Triggers:
 - This approach relies on SQLite's *CHECK* constraints rather than using triggers to enforce rules.

Limitations

- You must ensure through application logic or additional constraints that a *postID* in *Post* corresponds to only one type (*FeedPost*, *Reel*, etc.).
- This design doesn't fully enforce that a *postID* belongs exclusively to one subtype. For strict enforcement, consider adding a *type* column in *Post* with a *CHECK* or additional logic.

If flexibility is a priority and this redesign is unsuitable, using triggers or handling validation in application logic are your only alternatives in SQLite.

Considerações:

Apesar de tecnicamente possível, a sugestão do ChatGPT não está de acordo com nenhuma das maneiras de lidar com generalizações apresentada nas aulas, e além disso introduziria complexidade na base de dados.

Para ilustrar o maior grau de complexidade introduzido por esta alternativa, considere-se um cenário em que interessasse obter através de uma *query* as datas de publicação de todos os

Posts com tipo de conteúdo *PNG*. Com o esquema atual a *query* necessária seria apenas “SELECT postedTime FROM Post WHERE videoOrImageLink LIKE ‘%.png’;”, enquanto que com o esquema proposto pelo ChatGPT-4o, seria necessário usar uma *query* semelhante a essa para cada um dos tipos de post, seguidos de um *join*.

Por esses motivos, decidimos não adotar esta sugestão do ChatGPT-4o.

Final SQLite Database Creation (Task 5.1.10):

Após a análise das respostas fornecidas pela IA, decidimos apenas incluir a sugestão fornecida no ponto 4 da resposta do ChatGPT-4o ao Prompt 1, e consequentes mudanças derivadas dela, tal como descrito nas considerações.

Essa sugestão indica que devemos aplicar uma restrição ao atributo *state* da tabela *Report*, de modo a que este apenas possa conter um dos seguintes valores (“Pending”, “Resolved”, “Under Review”), assim evitamos que o atributo tenha valores não relacionados com a sua função.

Na sequência dessa restrição, decidimos implementar outras restrições semelhantes. Na tabela *User* decidimos restringir os valores possíveis do atributo *activityStatus* a *offline* e *online*. Na tabela *PrivacySettings* os valores de *directMessage* ficaram restringidos a *public*, valor que indica que o utilizador pode receber mensagens de todos os outros utilizadores, e *friends*, que implica que apenas amigos podem enviar mensagens ao utilizador. Nos restantes atributos referentes a visibilidade, os valores são limitados a *public* (visível a qualquer utilizador), *friends* (visível apenas a amigos) e *private* (visível apenas para o próprio utilizador).

Em suma, as únicas alterações feitas ao ficheiro “create1.sql” foram as descritas anteriormente e o ficheiro “create2.sql”, enviado em conjunto com este relatório, está atualizado de acordo com elas.

Data Loading

Initial Data Loading (Task 5.1.11):

Nesta tarefa procuramos povoar a base de dados explorando as restrições que impusemos, nomeadamente os valores *default* nas tabelas *PrivacySettings* e *Story* e as funções *CHECK* nas tabelas *User*, *Message*, *PrivacySettings*, *Comment*, *Report*, *Story*, *FeedPost* e *Reel*. Todas as restrições e chaves estrangeiras foram testadas durante a povoação da base de dados com tuplos que não as cumpriam. Desta maneira, conseguimos garantir o correto funcionamento das restrições e o cumprimento da integridade referencial. Procuramos também inserir alguns tuplos com atributos a *null* quando possível.

O *script* “populate1.sql”, que foi submetido em conjunto com este relatório, foi criado tendo em conta as preocupações referidas acima.

Generative AI Integration (Task 5.1.12):

Prompt 1:

(inclui o código do ficheiro “create2.sql” obtido a partir da tarefa 5.1.8)

(inclui o código do ficheiro “populate1.sql” obtido a partir da tarefa 5.1.11)

We tried to populate the Social Network database created by the file create2.sql in SQLite3. Considering the attributes and constraints of each table, propose additional INSERT statements for the User, Comment, and Message tables. As our Social Network is designed to operate globally, we want to include 30 new inserts featuring names in various languages for the username and name attributes in the User table. Additionally, we want diverse and unique content for 30 (of each) new comments, messages, and captions in different languages for the caption attribute in the FeedPost and Reel tables, as well as the text attribute in the Comment and Message tables. When possible, explore the use of NULL and DEFAULT values.

Prompt 2:

(Inclui create2.sql e populate1.sql em anexo)

Given the file that defines a database in SQL and the file that populates it, suggest specific improvements on the way the database is being populated. Answer in accordance to the best SQL practices.

Resposta do ChatGPT-4o ao Prompt 1:

Here are additional **INSERT** statements for the **User**, **Comment**, and **Message** tables, keeping in mind the global and multilingual scope of your Social Network, as well as the use of **NULL** and **DEFAULT** values where applicable.

User Table Inserts (30 Examples)

```
INSERT INTO User (user_id, username, name, email, password, created_at) VALUES
(1001, 'Дмитрий123', 'Дмитрий Иванов', 'dmitriy@example.com', 'securePass123', DEFAULT),
(1002, 'أمينة الكاظمي', 'أمينة', 'amina@example.com', 'safePwd456', DEFAULT),
(1003, '李小龙', '李小龙', 'brucelee@example.cn', 'kungFu789', DEFAULT),
(1004, '山田太郎', '山田 太郎', 'yamada@example.jp', 'securePass101', DEFAULT),
(1005, 'Élodie99', 'Élodie Dupont', 'elodie@example.fr', 'bonjour123', DEFAULT),
(1006, 'Carlos99', 'Carlos Mendoza', 'carlos@example.mx', 'secureCarlos', DEFAULT),
(1007, 'Anastasia', 'Анастасия Петрова', 'anastasia@example.ru', 'strongPass789', DEFAULT),
(1008, 'Zahra', 'زهرة أحمد', 'zahra@example.eg', 'pass123Zahra', DEFAULT),
(1009, 'PriyaMehta', 'Priya Mehta', 'priya@example.in', 'indianPass', DEFAULT),
(1010, 'Sebastian_', 'Sebastian Müller', 'sebastian@example.de', 'germanSecure', DEFAULT),
(1011, 'Fatou', 'Fatou Ndiaye', 'fatou@example.sn', 'passwordFatou', DEFAULT),
(1012, '김민수', '김민수', 'minsoo@example.kr', 'secureMins', DEFAULT),
(1013, 'João99', 'João Silva', 'joao@example.pt', 'portugueseJoao', DEFAULT),
(1014, 'Emma', 'Emma Johnson', 'emma@example.us', 'emmaSecure', DEFAULT),
(1015, 'Mateo', 'Mateo García', 'mateo@example.ar', 'argentinaMateo', DEFAULT),
(1016, 'Eva_01', 'Εύα Παπαδοπούλου', 'eva@example.gr', 'greekEva', DEFAULT),
(1017, 'Luca89', 'Luca Rossi', 'luca@example.it', 'italianLuca', DEFAULT),
(1018, 'ChenM', 'Chen Ming', 'chen@example.tw', 'taiwanChen', DEFAULT),
(1019, 'Victor123', 'Victor Ochieng', 'victor@example.ke', 'kenyaVictor', DEFAULT),
(1020, 'Kavya', 'Kavya Ramesh', 'kavya@example.in', 'kavyaPass', DEFAULT),
(1021, 'Hassan12', 'حسن مصطفى', 'hassan@example.sa', 'secureHassan', DEFAULT),
(1022, 'Marie', 'Marie Dubois', 'marie@example.fr', 'frenchMarie', DEFAULT),
(1023, 'José', 'José Pereira', 'jose@example.br', 'brazilJose', DEFAULT),
(1024, 'Fiona99', 'Fiona Ng', 'fiona@example.sg', 'singaporeFiona', DEFAULT),
(1025, 'Pedro77', 'Pedro Torres', 'pedro@example.cl', 'chilePedro', DEFAULT),
(1026, 'AhmedAli', 'Ahmed Ali', 'ahmed@example.ae', 'secureAhmed', DEFAULT),
(1027, 'Sophia', 'Sophia Lee', 'sophia@example.ca', 'sophiaPass', DEFAULT),
(1028, 'Olga', 'Ольга Смирнова', 'olga@example.ru', 'russianOlga', DEFAULT),
(1029, 'Zanele99', 'Zanele Khumalo', 'zanele@example.za', 'zaneleSecure', DEFAULT),
(1030, 'Aiko89', '相子', 'aiko@example.jp', 'aikoPass', DEFAULT);
```

Comment Table Inserts

```
INSERT INTO Comment (comment_id, user_id, post_id, text, created_at) VALUES
(2001, 1001, 101, 'Я только что узнал что-то новое из этого поста!', DEFAULT),
```

(2002, 1002, 102, 'هذا المنشور ذكرني بشيء مهم.', DEFAULT),
 (2003, 1003, 103, '这让我想起了一段美好的回忆。', DEFAULT),
 (2004, 1004, 104, '投稿ありがとうございます！とても役立ちました。', DEFAULT),
 (2005, 1005, 105, 'C'est fascinant, merci pour le partage.', DEFAULT),
 (2006, 1006, 106, 'Nunca había pensado en esto antes, ¡gracias!', DEFAULT),
 (2007, 1007, 107, 'Очень полезный пост, спасибо!', DEFAULT),
 (2008, 1008, 108, 'مقال رائع يحمل الكثير من الفائدة.', DEFAULT),
 (2009, 1009, 109, 'यह जानकारी मेरे लिए बिल्कुल नई है।', DEFAULT),
 (2010, 1010, 110, 'Das habe ich vorher noch nie gewusst. Vielen Dank!', DEFAULT),
 (2011, 1011, 111, 'Cet article est vraiment inspirant.', DEFAULT),
 (2012, 1012, 112, '정말 흥미로운 정보 감사합니다!', DEFAULT),
 (2013, 1013, 113, 'Adorei saber mais sobre isso, ótimo post!', DEFAULT),
 (2014, 1014, 114, 'This gave me a lot to think about, thank you.', DEFAULT),
 (2015, 1015, 115, 'Me encanta cómo explicaste este tema.', DEFAULT),
 (2016, 1016, 116, 'Πολύ καλά γραμμένο, με βοήθησε πολύ!', DEFAULT),
 (2017, 1017, 117, 'Non avrei mai immaginato qualcosa del genere!', DEFAULT),
 (2018, 1018, 118, '我真的从这篇文章中学到了很多。', DEFAULT),
 (2019, 1019, 119, 'Hii ilinipeleka kwenye enzi za zamani!', DEFAULT),
 (2020, 1020, 120, 'నేను మీ వ్యాఖ్యలను మించిపోలేను!', DEFAULT),
 (2021, 1021, 121, 'مقالة تحتوي على الكثير من التفاصيل المثيرة للاهتمام.', DEFAULT),
 (2022, 1022, 122, 'Merci beaucoup pour cet excellent contenu.', DEFAULT),
 (2023, 1023, 123, 'Muito obrigado por compartilhar este conhecimento.', DEFAULT),
 (2024, 1024, 124, '感谢您的努力和贡献。', DEFAULT),
 (2025, 1025, 125, 'Este artículo me hizo reflexionar bastante.', DEFAULT),
 (2026, 1026, 126, 'لقد استمتعت بكل كلمة.', DEFAULT),
 (2027, 1027, 127, 'This was exactly what I was looking for!', DEFAULT),
 (2028, 1028, 128, 'Теперь мне гораздо понятнее, спасибо.', DEFAULT),
 (2029, 1029, 129, 'Ngiyabonga ngokuhlaba umxhwele ngalesi sihloko.', DEFAULT),
 (2030, 1030, 130, '詳細に書かれていてとても助かりました！', DEFAULT);

Message Table Inserts

INSERT INTO Message (message_id, sender_id, receiver_id, text, sent_at) VALUES
 (3001, 1001, 1002, 'Привет, как дела?', DEFAULT),
 (3002, 1002, 1003, 'مرحبا كيف حالك؟', DEFAULT),
 (3003, 1003, 1004, '你好，你怎么样？', DEFAULT),
 (3004, 1004, 1005, 'こんにちは、お元気ですか？', DEFAULT),
 (3005, 1005, 1006, 'Salut, comment ça va?', DEFAULT),
 (3006, 1006, 1007, 'Hola, ¿cómo estás?', DEFAULT),
 (3007, 1007, 1008, 'Как у тебя дела?', DEFAULT),
 (3008, 1008, 1009, 'كيف أخبارك؟', DEFAULT),
 (3009, 1009, 1010, 'तुम कैसे हो?', DEFAULT),
 (3010, 1010, 1011, 'Wie geht es dir?', DEFAULT),
 (3011, 1011, 1012, 'Comment vas-tu?', DEFAULT),
 (3012, 1012, 1013, '안녕하세요, 잘 지내세요?', DEFAULT),
 (3013, 1013, 1014, 'Oi, como você está?', DEFAULT),

(3014, 1014, 1015, 'Hey, how are you?', DEFAULT),
 (3015, 1015, 1016, 'Hola, ¿cómo te va?', DEFAULT),
 (3016, 1016, 1017, 'Γεια σου, τι κάνεις;', DEFAULT),
 (3017, 1017, 1018, 'Ciao, come stai?', DEFAULT),
 (3018, 1018, 1019, '你好, 你最近怎么样?', DEFAULT),
 (3019, 1019, 1020, 'Habari yako?', DEFAULT),
 (3020, 1020, 1021, 'హాయ్, మీరు ఎలా ఉన్నారు?', DEFAULT),
 (3021, 1021, 1022, 'مرحبًا ، كيف حالك؟', DEFAULT),
 (3022, 1022, 1023, 'Salut, comment ça va?', DEFAULT),
 (3023, 1023, 1024, 'Olá, como você está?', DEFAULT),
 (3024, 1024, 1025, '你好, 你好嗎?', DEFAULT),
 (3025, 1025, 1026, 'Hola, ¿cómo estás?', DEFAULT),
 (3026, 1026, 1027, 'كيف حالك اليوم؟', DEFAULT),
 (3027, 1027, 1028, 'How are you?', DEFAULT),
 (3028, 1028, 1029, 'Как ты сегодня?', DEFAULT),
 (3029, 1029, 1030, 'Ninjani namuhla?', DEFAULT),
 (3030, 1030, 1001, '元気ですか?', DEFAULT);

New Captions for FeedPost and Reel Tables

INSERT INTO FeedPost (post_id, user_id, caption, created_at) VALUES
 (4001, 1001, 'Exploring new horizons today. The world is full of opportunities!', DEFAULT),
 (4002, 1002, 'Le ciel est bleu, et mon cœur est heureux. #Bonheur', DEFAULT),
 (4003, 1003, '¡Aventuras por delante! No hay límites para lo que podemos lograr.', DEFAULT),
 (4004, 1004, '今日は素晴らしい一日だった！新しい発見をしました。', DEFAULT),
 (4005, 1005, 'Learning something new every day keeps life exciting. #Growth', DEFAULT),
 (4006, 1006, '¡Es increíble cómo la naturaleza puede inspirarnos! #Conexión', DEFAULT),
 (4007, 1007, 'Noviembre me llena de gratitud. Hoy es un buen día para reflexionar.', DEFAULT),
 (4008, 1008, 'Todo es más bonito cuando lo ves desde una perspectiva positiva.', DEFAULT),
 (4009, 1009, 'La vida es un viaje, y cada paso cuenta. #VivirConPasión', DEFAULT),
 (4010, 1010, 'Explorando mi ciudad y encontrando belleza en lo cotidiano. #LocalLife', DEFAULT),
 (4011, 1011, 'All we need is a little bit of nature to reset our souls. 🌿', DEFAULT),
 (4012, 1012, 'Happiness is found in the little things. Taking time to enjoy life.', DEFAULT),
 (4013, 1013, '今天是我的幸运日！我感到很幸福。', DEFAULT),
 (4014, 1014, 'Viver a vida com coragem e gratidão. #CrescimentoPessoal', DEFAULT),
 (4015, 1015, 'Amazing what a bit of sunshine and a good book can do for your soul.', DEFAULT),
 (4016, 1016, 'It's a new day, a new chance to do something great. #Motivation', DEFAULT),
 (4017, 1017, 'Un voyage inoubliable commence par une petite étape.', DEFAULT),
 (4018, 1018, 'Embrace every moment. The world is beautiful. 🌍', DEFAULT),
 (4019, 1019, 'Visez haut, rêvez grand. Chaque jour est une nouvelle opportunité.', DEFAULT),
 (4020, 1020, 'A day in nature always brings new perspective. #PeacefulMind', DEFAULT),
 (4021, 1021, 'La vie est pleine de petites joies. Il suffit de savoir les apprécier.', DEFAULT),
 (4022, 1022, 'Shining bright, like the sun after the rain. #Gratitude', DEFAULT),
 (4023, 1023, 'Les meilleures choses arrivent quand on s'y attend le moins.', DEFAULT),
 (4024, 1024, 'Life is a beautiful journey. Just enjoy the ride. #LifeGoals', DEFAULT),

(4025, 1025, 'Chaque instant compte, alors profitons-en pleinement.', DEFAULT),
(4026, 1026, 'Waking up to a new adventure, one day at a time. #DreamBig', DEFAULT),
(4027, 1027, 'Don't just exist, live! Make every day count.', DEFAULT),
(4028, 1028, 'La naturaleza me habla en susurros. Hoy escuché algo especial.', DEFAULT),
(4029, 1029, '明日への一歩が、今日の選択から始まる。', DEFAULT),
(4030, 1030, 'Life is what you make of it, so make it count. #LiveFully', DEFAULT);

These entries feature varied and meaningful messages and comments in different languages, adding richness to your Social Network's database and adhering to global inclusivity! Let me know if further refinements are needed!

Considerações:

No que diz respeito à resposta ao Prompt 1, ChatGPT-4 tentou cumprir o que foi solicitado, mas teve dificuldades significativas e cometeu erros que tornam a resposta quase inútil. Não conseguiu identificar corretamente os atributos das tabelas do ficheiro “create2.sql”, nem seguiu os exemplos de inserts já fornecidos em “populate1.sql”. Além disso, criou atributos adicionais que não fazem sentido no contexto pedido.

Criou utilizadores de diversas partes do mundo e conseguiu implementar comentários e mensagens com conteúdo variado e único entre eles, conforme solicitado. No entanto, foi capaz de gerar apenas 30 novos inserts para o *FeedPost*, não atendendo ao pedido similar para os Reels.

O ChatGPT não considerou a existência do AUTOINCREMENT, e por isso começou a enumerar os IDs a partir de 1000 (nos *Users*), 2000 (nos *Comments*), 3000 (nas *Messages*) e 4000 (nos *FeedPosts*). Embora isso não afete o funcionamento do programa visto que os IDs são todos distintos, já que temos o atributo de AUTOINCREMENT, queremos que esses atributos tenham valores sejam consecutivos.

Esta ferramenta de IA não conseguiu entender o conceito de NULL e DEFAULT, e por isso, em vez de realizar o que foi solicitado, apenas gerou inserts nas tabelas com um novo atributo - *created_at* - que ficou sempre definido como DEFAULT, o que não faz sentido. Nos atributos onde seria possível inserir NULL, ou onde o NULL adotaria um valor DEFAULT, a IA não aproveitou para variar os inserts e testar esses casos.

Resposta do ChatGPT-4o ao Prompt 2:

After analyzing the provided SQL files, here are the suggested improvements to optimize and align the database population process with best SQL practices:

General Improvements

1. **Enable Atomic Transactions:**

- Use `BEGIN TRANSACTION` and `COMMIT` to ensure that the entire population process is atomic. This prevents partial data insertion in case of errors.

2. **Avoid Hardcoding Sensitive Information:**

- Replace hardcoded passwords (e.g., `'password0'`) with hashed values using a secure algorithm like bcrypt or SHA-256.

3. **Normalize Data:**

- If fields like `profilePictureLink` use a consistent URL structure, consider normalizing these into a separate table with unique IDs and references.

4. **Utilize Prepared Statements or Parameterized Queries:**

- If this script will run dynamically (e.g., through a programming interface), parameterized queries can prevent SQL injection.
-

Specific Suggestions for the `User` Table

1. **Insert in Batches:**

- Split large `INSERT INTO User` statements into smaller batches (e.g., 100 rows per batch). This reduces memory usage and improves database performance during population.

2. **Use Default Values Where Appropriate:**

- If `activityStatus` commonly defaults to `'offline'`, define it as the column's default value in the schema (`activityStatus TEXT DEFAULT 'offline' CHECK(...)`) and omit it from inserts unless explicitly needed.
-

Example Refactored `User` Insert with Transactions and Defaults

```
BEGIN TRANSACTION;
```

```
INSERT INTO User (username, name, email, password, birthDate, profilePictureLink, activityStatus)
VALUES
```

```
('gustavomendes', 'Gustavo Mendes', 'gustavo.mendes@exemplo.com', 'hashed_password_0', '1990-01-15',
'http://example.com/profile0.png', 'online'),
```

```
('larafonseca', 'Lara Fonseca', 'lara.fonseca@exemplo.com', 'hashed_password_1', '1988-03-22',
'http://example.com/profile1.png'),
```

```
('mateusribeiro', 'Mateus Ribeiro', 'mateus.ribeiro@exemplo.com', 'hashed_password_2', '1994-07-12',  
'http://example.com/profile2.png');
```

```
COMMIT;
```

Testing and Validation

1. **Add Constraints for Data Integrity:**

- Use triggers or constraints to enforce business rules, such as checking for valid dates or ensuring profile picture URLs are accessible.

2. **Run Tests After Population:**

- Validate the integrity of populated data using SQL queries to count records, check foreign key integrity, and confirm data uniqueness.

Let me know if you want me to refactor the entire script or focus on a specific section.

Considerações:

No ponto 1 de “General Improvements” o ChatGPT-4o sugere o uso de instruções que ainda não foram abordadas nesta UC. No ponto 2 da mesma secção, a ferramenta de IA expressa uma preocupação que, embora pertinente, está fora do âmbito deste projeto já que deve ser implementada pela plataforma que faça a interface entre o utilizador e a base de dados. No ponto seguinte, a IA faz depender a alteração da existência de uma estrutura bem definida de URL, que não existe. No último ponto, esta IA generativa volta a utilizar estratégias que não foram abordadas nesta UC. Pelos motivos, referidos, decidimos não adotar nenhuma das sugestões da secção *General Improvements*.

Na secção *Specific Suggestions for the User table*, o ChatGPT-4o começou por fazer uma sugestão redundante, já que em nenhuma instância inserimos mais de uma centena de tuplos com o mesmo comando. A segunda sugestão não parece ser adequada, já que no momento da inserção dos tuplos na tabela *User*, os utilizadores vão estar, geralmente, ativos. Por estes motivos, decidimos também não adotar nenhuma das sugestões dadas pela IA nesta secção.

A secção seguinte mostra um exemplo do uso de instruções que ainda não abordamos nesta UC, pelo que não a iremos usar no nosso projeto.

Por fim, o ChatGPT-4o sugere o uso de restrições, que já estão extensivamente em uso, e de triggers, que ainda não foram abordados na UC e por isso não serão integrados no projeto. A última sugestão desta IA já estava a ser adotada e vai continuar a ser, já que é a maneira mais eficiente de validar o nosso trabalho de implementação e povoação da base de dados.

Final Data Loading (Task 5.1.13):

Após revermos as respostas da ferramenta de IA generativa, a única alteração que fizemos ao ficheiro populate1.sql foi substituir alguns dos utilizadores existentes por utilizadores que falam outras línguas. Esta mudança contribui para uma base de dados mais representativa de uma rede social global.

Além disso, ao atualizar os utilizadores, também ajustámos as mensagens e comentários associados a eles, conforme as sugestões do ChatGPT na resposta ao Prompt 1, garantindo uma maior coerência no conteúdo armazenado na nossa base de dados. Estas novas alterações foram implementadas no ficheiro populate2.sql.

Generative AI Integration

Das ferramentas de Inteligência Artificial Generativa que usamos ao longo do projeto, o ChatGPT-4o foi de longe a mais útil. Esta ferramenta deu respostas mais corretas e concretas do ponto de vista técnico e mais enquadradas no contexto deste projeto em específico. Ainda assim, as ferramentas de IA generativa mostraram ter utilidade limitada, já que a maioria das soluções propostas por elas estavam erradas ou não eram as mais adequadas.

A criação da base de dados em SQL foi a fase em que a IA generativa foi mais útil. Nesta fase, a IA deu várias sugestões que consideramos adequadas e que por isso decidimos adotar, nomeadamente em relação a restrições. A IA também se mostrou bastante útil na fase de povoação da base de dados, mas exclusivamente devido à sua capacidade de gerar dados diversos num curto espaço de tempo e em vários idiomas.

A fase em que a IA se mostrou menos útil foi a da transformação do modelo conceptual em UML para o esquema relacional. Nesta fase, a única sugestão das ferramentas de IA que decidimos adotar foi a exclusão de um atributo derivado do esquema relacional. Nesta fase, a maioria das sugestões de IA iam contra aquilo que nos foi apresentado nas aulas e em alguns casos estavam mesmo erradas.

Participation of the Various Students of the group

Este trabalho foi desenvolvido de forma colaborativa pelos 3 integrantes do grupo. Todos os elementos contribuíram em todas as fases do projeto de forma equilibrada.

Trabalho realizado por:

António Lourenço Gomes Barrote Rodrigues: up202306331

Gonçalo Miguel Ferreira Santos: up202306340

Leandro Filipe Pais Resende: up202306343