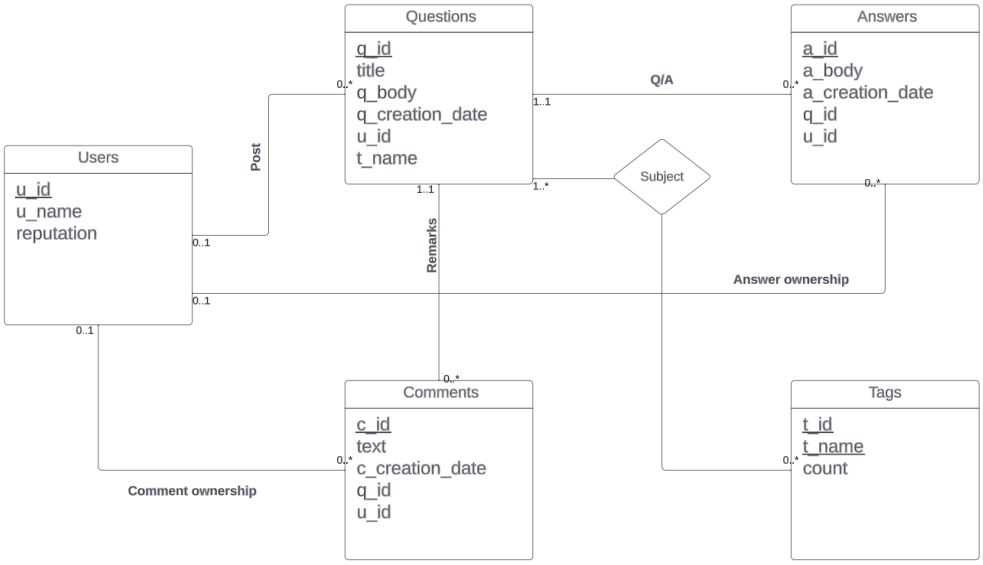
DATABASE DESIGN

UML DIAGRAM



Assumptions

Users:

- Every user has a unique user id
- Every user has a user name (May not be unique)
- Every user can have multiple or none questions/answers/comments

Questions:

- Every question has a unique question id
- Every question has a title
- Every question has a body
- Every question may or may not have a user id. (Anonymous posts/ deleted user accounts)
- Every question can have multiple or none comments
- Every question can have multiple or none answers
- Every question can have multiple or none tags

Answers:

- Every answer has a unique answer id
- Every answer has a body
- Every answer may or may not have a user id. (Anonymous posts/ deleted user accounts)
- Every answer will have exactly one question linked to it

Comments:

- Every comment has a unique comment id
- Every comment has a body
- Every comment may or may not have a user id. (Anonymous posts/ deleted user accounts)
- Every comment will have exactly one question linked to it

Tags:

- Every tag name is unique
- Each tag will have at least one question linked to it, hence making count positive

A description of each relationship and its cardinality

Q/A: Relationship between Questions and Answers and they are connected through question id. Its cardinality is one to many.

Subject: Relationship between Questions and Tags and they are connected via the tag names. This tells us what the question is about (Ex. Python/ Java/ SQL/ …. etc). Its cardinality is many to many.

Answer Ownership: Relationship between Users and Answers they posted and they are connected through user id. It tells us which answer is posted by which user and if the user is reputable or not. Its cardinality is one to many.

Remarks: Relationship between Questions and Comments and they are connected through question id. Its cardinality is one to many.

Post: Relationship between Users and the Questions they posted and they are also connected on user id. It tells us which user posted which question. Its cardinality is one to many.

Comment Ownership: Relationship between Users and Comments they posted and they are connected through user id. It tells us which comment is posted by which user. Its cardinality is one to many.

BCNF over 3NF

We chose BCNF because our data already complies with BCNF. All our tables have just one primary key, and each functional dependency has a super key which complies with BCNF. BCNF is a stricter form and an extension of 3NF which means that if our data is in BCNF, it is also in 3NF. We chose BCNF over 3NF also because it reduces the redundancy, and it is stronger.

Normalizing Database

All our relations are already in BCNF because they all have one primary key, which are also super keys.

Users(u_id, u_name, reputation) with FD
u_id → u_name, reputation
Here u_id is the primary key and a super key hence the relation is already in BCNF

Questions(q_id, title, q_body, q_creation_date, u_id, T_name) with FD
q_id → title, q_body, q_creation_date, u_id, T_name
Here q_id is the primary key as well as the super key hence the relation is already in BCNF

Answers(a_id, a_body, a_creation_date, q_id, u_id) with FD
a_id → a_body, a_creation_date, q_id, u_id
Here a_id is the primary key as well as the super key hence the relation is already in BCNF

Tags(t_name, count) with FD
t_name → count
Here t_name is the primary key as well as the super key hence the relation is already in BCNF

Comments(c_id, text, c_creation_date, q_id, u_id) with FD
c_id → text, c_creation_date, q_id, u_id
Here c_id is the primary key as well as the super key hence the relation is already in BCNF

Based on functional dependency and the fact that each relation has a super key, makes all these relations BCNF.

Relational Schema

Format:
Table-Name(Column1:Domain [PK], Column2:Domain [FK to table.column], Column3:Domain,...)

1) Users(u_id: VARCHAR(10) [PK], u_name: VARCHAR(255), reputation: INT)

2) Questions(q_id: VARCHAR(10) [PK], title: VARCHAR(255), q_body: VARCHAR(25000), q_creation_date: DATE, u_id: VARCHAR(10) [FK to Users.u_id), t_name: VARCHAR(255) [FK to Tags.t_name])

3) Answers(a_id: VARCHAR(10) [PK], a_body: VARCHAR(25000), a_creation_date: DATE, q_id: VARCHAR(10) [FK to Questions.q_id), u_id: VARCHAR(10) [FK to Users.u_id])

4) Comments(c_id: VARCHAR(10) [PK], text: VARCHAR(25000), c_creation_date: DATE, q_id: VARCHAR(10) [FK to Questions.q_id), u_id: VARCHAR(10) [FK to Users.u_id])

5) Tags(t_name: VARCHAR(255) [PK], count: INT)

6) Subject(q_id: VARCHAR(10) [PK, FK to Questions.q_id], t_id: VARCHAR(10) [PK, FK to Tags.t_id], t_name: VARCHAR(255) [PK, FK to Tags.t_name])