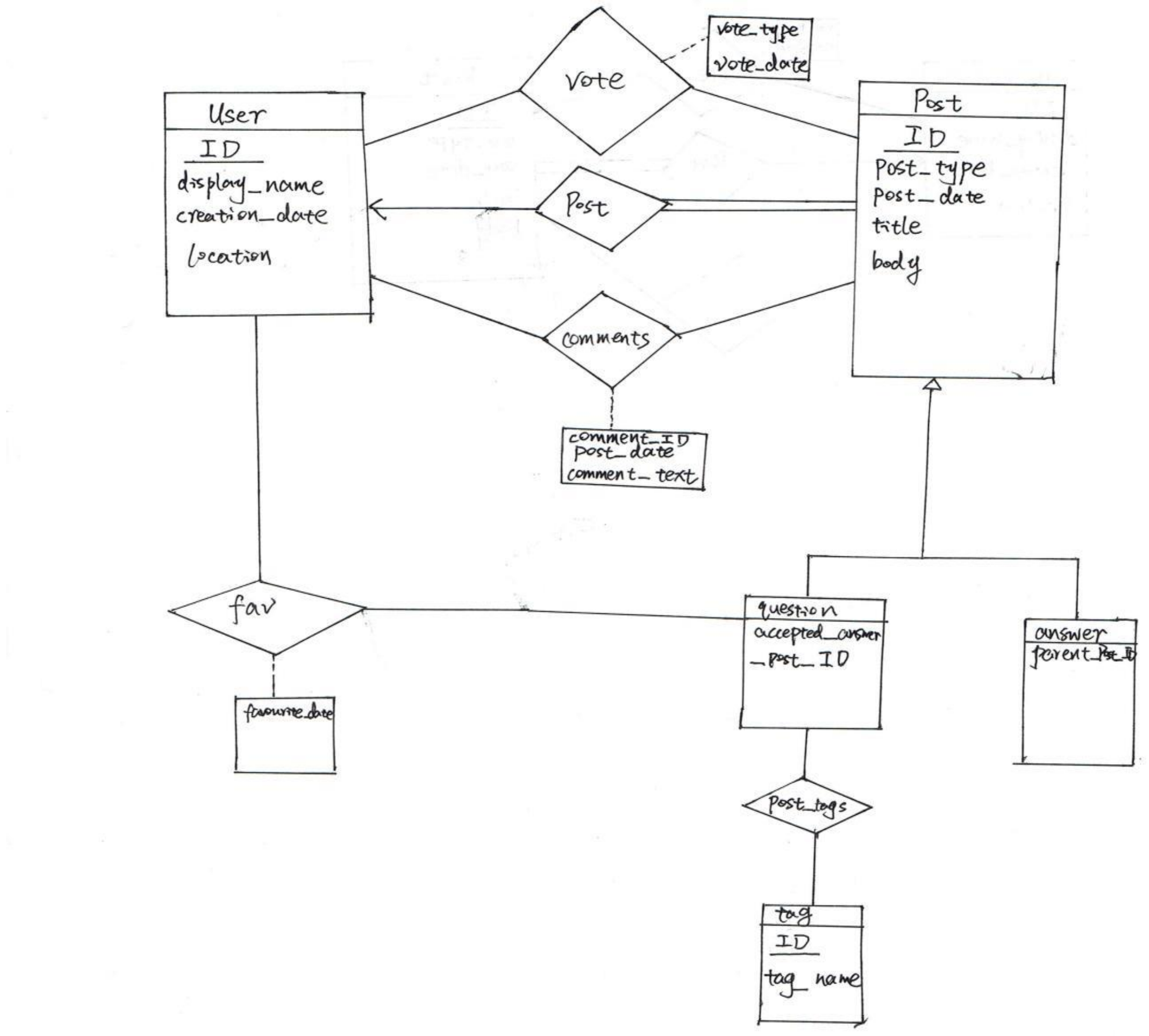
Report

This project is done by FengMi, WEIDA ZHU, Bohan Ouyang



E-R model formulation:

User table has three relationships with the Post table :

1 User can post posts on the site(one post is authored by exactly one user)
And the relationship between user table and post table is one to many (user at one side)
And also user cannot post posts but if post exists it must be created by user. So user is partial participation but post is total participation.

2. Users can vote posts. So there is a vote relationship between user table and post table. And one user could vote several posts and posts could be voted by several users. So they are many to many relationship.

Both user table and post table are partial participation since that is not compulsive to vote.

3. Users can comment on posts.

Comments are not compulsive so both user table and post table partially participate in this relationship.

The relationship of these two tables are many to many is because user can comment different posts and post can be commented by different users.

For question table and answer table:

Also since the requirement need to have two different kinds of posts .One is question and the another is answer. We make question and answer table inherits from post table .(Via this method we think it can be more clear but actually we merge question and answer with post table in relational model which could be more realistic and convenient to insert table into mysql.)

The relationship of fav:

User can click “favourite” on question so we create a fav relationship between user and question.

And since user can fav multiple questions and questions can be favourited by multiple users they are many to many relationship .

But also that is not compulsive so both users and questions are partial participation.

Since Users can tag posts using a pre-defined set of tags. We create a post_tag relationship between question and tag. Table tag is used to store the predefined tags.

Since question may have multiple tags and tags can be set on different questions .They are many to many relationship. Since question is not compulsive to have tags and predefined tags are not required to be used .Both of them are partial participation.

All the attributes in the table is according to the requirement from the project description.

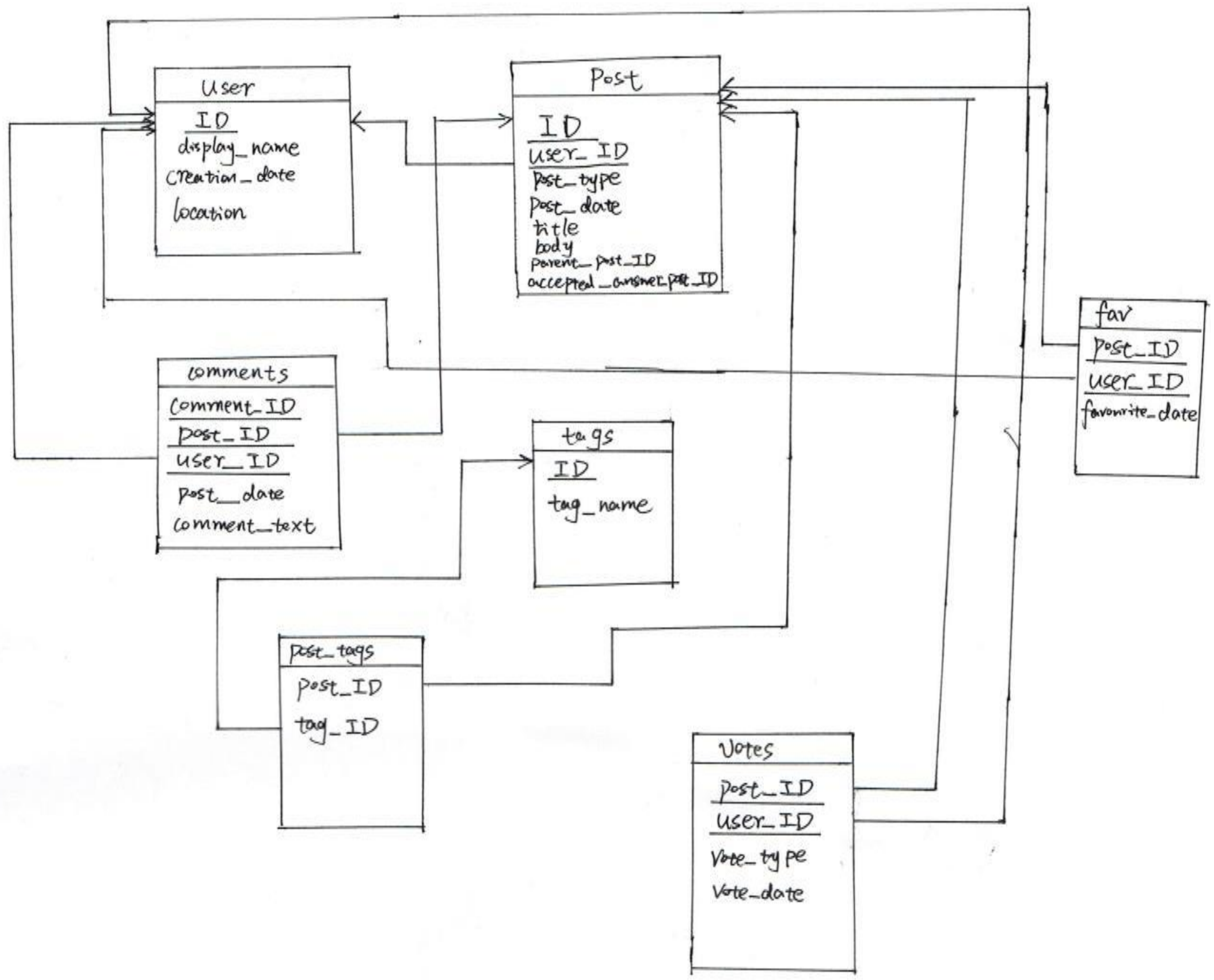
However, there are several requirements we are not able to capture in this E-R diagram.

First, users have directly relationship with predefined tag by the project description. But in order to make the diagram more concise we just create a relationship between question and tag. However this would not affect the integrity of the database.

Second, we can not capture the requirements of “a user may only favorite a particular question once.” Since this requirement is not the mapping cardinality nor the participation constraint we can not capture this feature by this E-R diagram.

Third, we can not capture the requirements of “users may only vote on any post once”. The reason is the same as the second point :this requirement is not mapping cordinality nor participation constraint.

Relational model translation:


User
ID
display_name
creation_date
location
Post
ID
user_ID
post_type
post_date
title
body
parent_post_ID
accepted_answer_post_ID
comments
comment_ID
post_ID
user_ID
post_date
comment_text
tags
ID
tag_name
post_tags
post_ID
tag_ID
Votes
post_ID
user_ID
Vote_type
Vote_date
fav
post_ID
user_ID
favourite_date

First, since question and answer inherit from table post. We merge the attributes which are originally in table question and answer into table post.

Second, since the relationship vote is many to many ,we retain the relationship vote as table vote and put both primary keys in table user and table post into table vote as its primary key .And make the primary key in table vote foreign key point to other two tables respectively.

Also insert the attribute “vote_type”,”vote_date” into vote table.

Third, since user and post is one to many relationship we delete the relationship table “post”. And the the primary key of user(one side) into table Post (many side) as foreign key points back the user table.

Third, Since the relationship comment is many to many ,we retain the relationship comment as table comment and put both primary keys in table user and table post into table comment as its primary key .And make the primary key in table comment foreign key point to other two tables respectively.

And also put the attribute “comment_ID”,”post_date”,”comment text” into table comment.

The method to address relationship table “fav”and “post_tags” is very similar with step three since both of these two relationships are many to many relation so they should be retained.

There is one feature lost from translating E-R model into relational model:the table user and table tag originally occur relation with table question.Since we merge the attributes in table question into table post.So they now have relationship with post .

Also for attribute post_type in post if “post_type”= 1 the table post represents question, if “post_type=2” then the table post represents answer.

(c) data import:

We just import original data from the project which instructor gave us into our table. These seven txt file match our seven tables very well.

(d)query optimization:

index creation:

create index parent_post_ID on post(parent_post_ID);

For 4.a After explain our query we found that there is no index on post.parent_post_id so we create a secondary index on that attribute .Then the performance of the query improved significantly. Since it could not need to full file scan this field.

For 4.a This query need 35.27s before the implementation.But after optimization using creating index it only took 11s to implement this query.

For a(I..) we tested that before optimization it took 17.24s to implement this query but after optimization it only took 3.29s to implement it.

For a.II the performance is 35s before the optimization but after optimization the performance it improved to 15s.

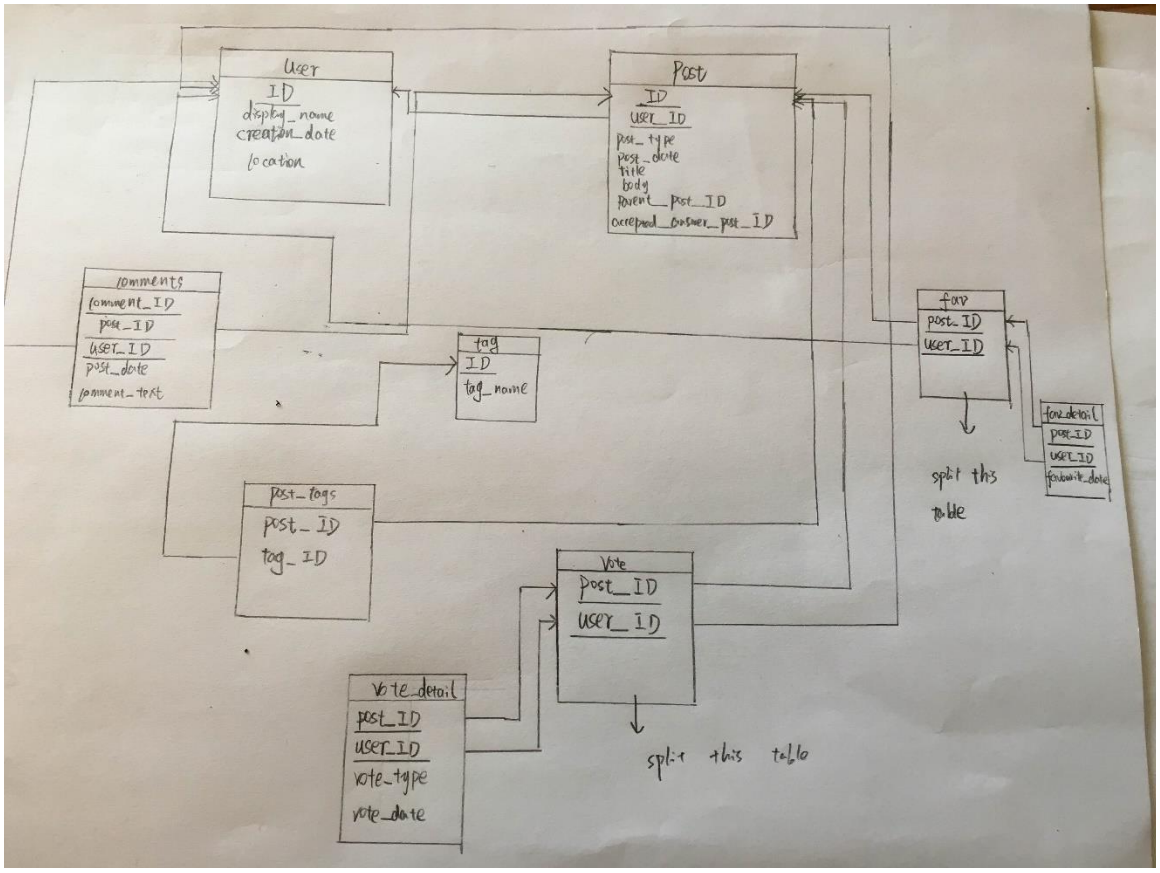
For a.III the performance is 13.77s after optimization, but before the optimization the performance is only 34.47s.

b. For query b, before optimization, the performance of this query is 1.3s.After optimization(create the index on post.parent_post_id) ,the performance of the query is 0.2s

Another trail:

We thought using splitting table to improve the performance of the query by putting the redundancy information in table vote and fav into another detail-table so that some information which is not used so frequent would not occupy blocks in our query.

The reason for this method may improve a bit performance is that table vote and table fav occupy less blocks .But in case we need the vote_type and vote_date and favourite_date in the future we put these information into table vote_detail,and table fav_detail.


User
ID
display_name
creation_date
location
Post
ID
user_ID
post_type
post_date
title
body
parent_post_ID
accepted_answer_post_ID
comments
comment_ID
post_ID
user_ID
post_date
comment_text
tag
ID
tag_name
post_tags
post_ID
tag_ID
fav
post_ID
user_ID
split this table
fav_detail
post_ID
user_ID
favorite_date
vote
post_ID
user_ID
split this table
vote_detail
post_ID
user_ID
vote_type
vote_date