**DATABASE SYSTEMS**

Project Report

**a) E R Model -**

We built the ER model with the help of the requirements given in the project.

Explanation of building ER below point by point:

1. Users had a some fields in them so we chose it as an Entity.
2. Users can author posts so we decided the other entity to be Posts and connected by a relationship of authors. The posts can have only one user so a single arrow pointed to users and all the posts will have a author and many user can have many posts so double participation with no arrow at Posts.
3. We divided the posts into questions and answers. Questions post id are separate from answers and they both below to posts so its an aggregation between them and they are mutually exclusive also.

- Here we cannot capture that each answer belong to only one question.

Their is a relation between questions and answers of hasaccepted which tells if there is a accepted answer for the question and it can only be one.

1. Users entity has a relation with entity Questions and a new entity Posttags with the relationship of tags. The tag relationship has a name of it.

-Here only that user who has posted the question can tag cannot be captured.

1. Users comments on posts and has a relationship with it. The relationship is comments and comment has its data and body as attributes. Many users can comment on many posts.
2. Users have relation with entity question and the relationship is favs. Favs has attribute date on which user favorites a question. Has many to many relationship with questions.

- Cannot capture fav can be done only once.

1. Users have relation with posts of voting. The attributes of vote relation are upvote downvote and has a many to many relations.

- Here a user can vote a post once cannot be captured.

**b) Relation Model -**

Looking to the ER Model, we build 9 tables to load Stackoverflow data.

The tables are users, question, answers, comments, tags, posttags, votes, favs, accepted.

Below is the explanation about the Relation Model and the Schema :

**users:** The users entity converted to users table with uid as the primary key and name, cdate, location as other fields.

**questions:** The posts were broken into questions and answers. The questions table had qid which is same as pid which is the primary key, uid references to user table uid and pdate, title, body, hasacc as other fields.

**answers:** The answers table had aid which is basically post id as primary key, uid referring to users uid, qid referring to questions qid because all answers have a question and other fields as pdate, body.

**tags:** The tags relation converted to a table which has all the tags in it. The tags table has tid as primary key and name as another field.

**posttags:** The entity posttags converted to table which contains all the tags made my the users for different questions. The pid and tid is the primary key for this table and tags for each question can be uniquely determined.

**comments:** The comment relation converts to comments table. The primary key is cid, pid refers to both questions and answers table qid and aid respectively. The other fields are cdate, ctext.

**favs:** The relation favs convert to favs table. The primary key is both pid and uid, pid referes to questions pid and uid references to users uid. The other fiels is the fav date fdate.

**votes:** The votes can be both for questions and answers. So the primary key is pid, uid where pid refers to both qid and aid and uid refers to users uid. The votes also have vote type and vote date.

**c) Data Import -**

The data was split in some .txt files but we decided to preprocess it so that we can have the desired data ito the out tables.

We need to create 3 new text files and alter one text file because there was a redundant column in it.

New Text File: ques.txt, ans.txt acc.txt

Alter Text File: favs.txt

**The description is following:**

Posts Text File:

This text file had all the question and answers. Had many fields which tell about whether its a question or an answer. Is there any accepted answer for a question.

We decided to split the posts into ques.txt and ans.txt so it separately have the question and answers. We did the preprocessing of in python. The script will be attached with the submission.

* If the post type was 1, then its a question and that record has to go in ques.txt.
* a field hasacc added to question reord that tells it has a accepted answer or not.
* If the question has accepted answer then the qid and aid goes in acc.txt and question record hasacc is 1.
* If the question doesn’t have accepted answer then hasacc is 0.
* The post type, parent postid and answer post id was removed from the question record. This record is added too ques.txt.
* If its not a question then its an answer. Post type, title(because answer doesn’t have one), accepted post id are removed from the record and this record is added to ans.txt.

Fav Text File:

This text file had a field which was not used and not in the project description. The third column was removed from each record.

All the files were closed and we have three new files and one altered file.

When loading data, there was some problem with 1452 post id. I think there were some rows in favs or votes of 1452 and no record for it in posts.txt so at that time only we have to disable the foreign key constraint.

**d) Query Optimization -**

* Not build any index. The indexed were on the fields we wanted. Instead re-factored the data and also added additional field to have an optimized query. We did all things when we were building the database so it was very hard to optimize it.
* We re-factored the posts.txt into ques.txt and ans.txt to have an efficient query. Also we added a field hasacc(has accepted answer) for each of the question record. It will tells whether the question is a open question or it has a accepted answer.
* Also we made another txt file acc.txt which has only the accepted answer for every question.

The performance of the queries are pretty good. Its shown below:

|  |  |
| --- | --- |
| Question | Run Time(seconds) |
| a | 1.40 |
| a i | .45 |
| a ii | 1.60 |
| a iii | 1.62 |
| b | .12 |

**\*Note:**

* The queries for a) are all fine. Re-checked many times but ai), aii), aiii) halts at a particular spot when running. We had to do Ctrl + D for halting and viewing how many rows we got. It shows us 35712 rows for a) but the open questions should be 36274. The queries are correct but don’t know why its not doing it and justs halts.