FINAL PROJECT REPORT

# ER DIAGRAM

https://lh4.googleusercontent.com/oqMSdGBImR2xoa4q2qDjuYqzuvf0ircYfpXFcoEb_XTpRXpb51fTZKW20K4VIPzjffnXZcmWEDD_6AbOfzIW85IZdFtBYk_9PD0OcP81AdtLUzr-95R8I-OeZyDjDmIngsaPCw5N

Entities are represented in italics; Relationships are in bold.

|  |  |
| --- | --- |
| **Requirement** | **Solution** |
| 1.a | *User* is created with attributes display name, creation date, location and user id. |
| 1.b | *Post* is created with attributes post date, title, post type and body. One to many relation (**Authors**) between *Post* and *User* tables is created to ensure that each post is authored by exactly one user. |
| 1.c | *Question* and *Answer* inherited from *Posts* are disjoints. A one to many relation (**Belongs**) between themensures a questioncan have multiple answer and answers belongs to exactly one question post.  Also, a one to one relationship (**Accepted Answer**) between them ensuresthatat most one correct answer can be accepted for a question. |
| 1.d | *Tag* is created with attributes tag id and name. Many to many ternary relationships between *Tag*, *User* and *Question* ensures that questions can be tagged by users.  **Not captured in ER:** The ER doesn’t recognize that questions posts are to be tagged onlyby the user that posted them. In front-end application layer ensure that only author of the *Question* can add tags to the post. |
| 1.e | *Comment* is created with attributes date, body and comment id. To ensure users comment belong to exactly one post there is a ternary many to one relation (**Comments**) between *Comment*, *User* and *Post* |
| 1.f | There is a many to manyrelation (**Favorites**) between *Question* and *User* with attributes fav\_ date (required to support data in text file)  **Not captured in ER:**  User may favorite a particular question just once.  **Possible Solutions:**   1. Fire a trigger before user favorites a question to check whether the same user has already favorited the post. 2. In Favs table create a composite primary key with attributes post id and user id that are foreign keys referencing tables Post and User respectively. (**chosen** solution for this project) |
| 1.g | There is a many to many relationship (**Votes**) between *Post* and *User* with attributes date and type.  **Not captured in ER:**  User may favorite a particular question just once.  **Possible Solutions:**  a. Fire a trigger before user votes a post to check whether the same user has already up-voted or down-voted the post.  b. In the Votes table create a composite primary key with attributes post id and user id that are foreign keys referencing tables Post and User respectively. (**Chosen** solution for this project) |

# RELATIONAL MODEL

C:\Users\Vinit\Downloads\Relational Schema (1).png

Changes necessary to convert ER diagram to relational database model:

Entities are represented in italics, relationships are in bold and tables are underlined.

|  |  |
| --- | --- |
| # | Description |
| 1 | In Posts table, user\_id is a foreign key to the primary key of User table to ensure post is authored by a single user. |
| 2 | MySQL does not support inheritance. Two alternate database schema solutions are implemented in this project to ensure mapping requirements:  Project version 1: A table Posts with two fields parent\_post\_ID and accepted\_post\_ID is created to incorporate the relations **‘Belongs’** and **‘Accepted Answer’.** These keysrefer the primary key of Posts (self-referencing foreign keys).  Project version 2: Create individual tables for questions and answers. |
| 3 | Comments table: user\_id and post\_id are foreign keys referencing tables Users and Posts respectively such that users can comment on only one post. |
| 4 | **‘Favorites’** relationis transformed into a table with foreign keys post\_id and user\_id, referencing tables Posts and Users. |
| 5 | **‘Votes’** relation is transformed into a table with foreign keys post\_id and user\_id referencing to the primary keys of tables Posts and Users. |

# DATA IMPORT

|  |  |
| --- | --- |
| **Decisions while importing data** | |
| 1 | **Order** |
|  | Data from text files is inserted into Users, Tags, Posts and Comments tables consecutively. Followed by loads on Posttags, Favs and Votes. The order of insertion is necessary to maintain foreign key constraints that exist between the tables. |
| 2 | **Loading data into Posts table** |
|  | Loading data by turning off foreign key constraints.  Project Version 1: Due to posts table referencing itself, the method violates integrity constraints. Using set\_foreign\_key\_checks=0 setting reduces the time to insert data and doesn’t violate the business logic. Foreign key constraints are turned on again after insertion  Project Version 2:  Data is loaded into Questions and Answers tables similar |
| 3 | **Loading data into Favs table** |
|  | Text data for Favorites table has redundant data. A ‘Dummy’ column is created at position 3 to load this redundant data. This column is deleted after loading. |
| 4 | **Decreasing time to import data from text file to tables** |
|  | Change innodb\_buffer\_pool\_size from 8M (default) to 1G and innodb\_log\_file\_size from 48M (default) to 250M (log file size should be minimum 25% of buffer\_pool for optimal results).  **For example**  Time to load data before (Posts table):42.50 sec  Time to load data after (Posts table): 26.03 sec |
| 5 | **Observations** |
|  | 1. Some posts have anonymous user. Some user ids in Posts are not listed in Users table 2. Some posts do not have any tags 3. Some posts do not have favorites |

# QUERY OPTIMIZATIONS

Queries were run on both version 1 and 2. The first version is described in the sections above. The second versions splits Posts table into Questions (fields id, post\_date, user\_id, title, body, accepted\_post\_ID) and Answers (ID, parent\_post\_ID, post\_date, user\_id, body).

|  |  |
| --- | --- |
| **Question Number** | **Query Description** |
| 4.a.i | Step 1: Secondary index lookup on tags table using tags\_name.  Step 2: Secondary index loop on posttags using tags\_id.  Step 3: Secondary index loop on posts using accepted\_post\_id to find questions which are open.  Step 4: Primary index loop on users table  Step 5: Secondary index loop on posts for finding answers using parent\_post\_ID.  Step 6: Secondary index loop on votes table using index post\_id.  Step 7: File sort to perform aggregation  Post tags gives us the question IDs having tag name=’Java’ (Step 1 and Step 2). These question IDs are used to find answers, user\_name and total number of votes.  Query time normal: **3.5 seconds**  Optimized time (Version 1 of database): **0.227 sec**  Optimized time (Version 2 of database): **0.19 sec** |
| 4.a.ii | Step 1. Full file scan of posttags  Step 2. Primary index lookup of posts to infer questions  Step 3. Primary index lookup on users  Step 4. Secondary index lookup on votes using post\_id  Step 5. Primary index lookup on tags  Step 6. Secondary index lookup on posts using parent\_post\_ID  Step 7. File sort to perform aggregation  Step 8: File sort to order  Optimized time (Version 1 of database): **3.292sec**  Optimized time (Version 2 of database): **2.411 sec** |
| 4.a.iii | Step 1. Full file scan of posttags  Step 2. Primary index lookup of posts to infer questions  Step 3. Primary index lookup on users  Step 4. Secondary index lookup on votes using post\_id  Step 5. Primary index lookup on tags  Step 6. Secondary index lookup on posts using parent\_post\_ID  Step 7. File sort to perform aggregation  Step 8: File sort to order  Optimized time (Version 1 of database): **3.235sec**  Optimized time (Version 2 of database): **2.399 sec** |
| 4.b | Step 1. Nested SQL queries are used to calculate number of votes and favorites.  Step 2. Subqueries are used to calculate answers, their votes and comments  Optimized time (Version 1 of database): **0.0076sec**  Optimized time (Version 2 of database): **0.0064sec** |