QUESTION AND ANSWER PORTAL

Sai Prashanth Selvaganapathy
50419614

Aravind Balakrishnan
50412479

ABSTRACT

The aim of our project is to create a database that would contain information of all Question and Answers where users can ask or answer different queries grouped into various communities (Programming, photography, OS etc.). It would contain all the required information such as the community members, users, Post, likes, review and moderation process.

I. INTRODUCTION

Many question and answer portals are so widely used by modern day users. They have made a great impact on present day customers. It provides a sheer volume of usable content which has really helped many users. Our portal is jam packed with tons of information. Users come here to find answers to all sorts of questions they are struggling with. You will learn a lot of things everyday by new questions and answers posted here in our portal. There are questions that you research before answering, so you improve your knowledge.

We needed to provide a portal where anyone can ask any questions, add answers comments…

- Ability to group like-minded users/posts into communities
- Moderation by community by members itself
- Simple and yet able to capture all use cases
- Review process to prevent unwanted questions
- Able to encourage user participation

II. PROBLEM STATEMENT

Having many users base will generate hundreds of thousands of posts. These posts (Question, Answers, comments etc.) needs to be retrieved as quickly and efficiently as possible. Retrieving by other means will consume a lot of time and users will not have that much patience too. Storing these data in an excel will be a time consuming and a slow process. As well we need to filter the data stored to retrieve only the necessary particular tuples according to user query. Searching and indexing these data becomes a huge cumbersome task.To make this task easier we have used a database like Postgres.

III. TARGET USER

The dataset contains all Question and Answers where there are two types of communities/portals this Application DB will provide:

- Public communities: A community where any users are allowed to join, participate and add a post.
- Private communities: A community where only certain users are allowed to be joined as per the community author or the administrator.

IV. DATASET DESCRIPTION

We took data from existing application (Stack overflow) We used an online query tool from stack exchange to obtain the data. The following dataset is prepopulated in our DB:

Top 500 questions along with top 5 answers based on score(votes) for a total of 3000 questions

5000 users from stack overflow whose info are publicly available

Random comments, votes , review are generated to populate and test various aspect of the db.

V. LIST OF RELATION AND ATTRIBUTES

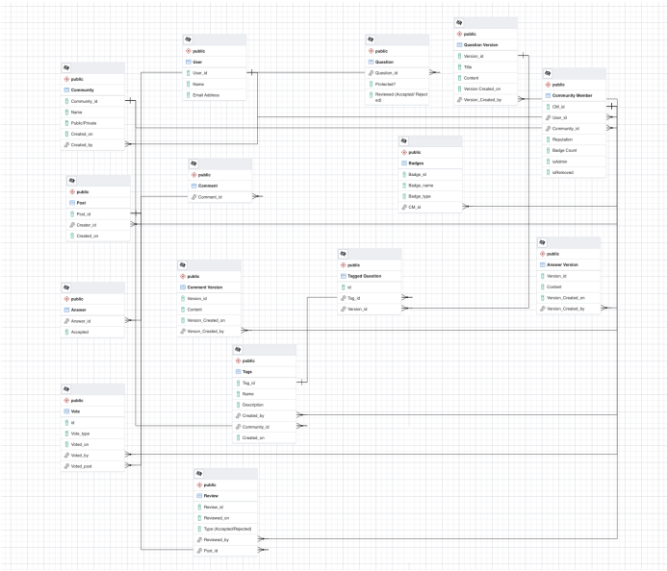
- Community (Community_id : bigint PRIMARY KEY, name : VARCHAR, type VARCHAR, created_on : timestamp, created_by : bigint.)
- User(User_id : bigint PRIMARY KEY, Name : varchar, Email Address : varchar)
- Community_Member(CM_id : bigint PRIMARY KEY, User_id : bigint, Community_id : bigint, Reputation: int, BadgeCount : int, isAdmin : Boolean, isRemoved : Boolean)
- Vote(id : bigint PRIMARY KEY,
- Vote_type : varchar, Voted_on : timestamp, Voted_by : bigint, Voted_post :bigint)
- Question(Question_id : bigint PRIMARY KEY, Protected : Boolean, Reviewed : varchar)
- Question_Version(Version_id : bigint PRIMARY KEY, Question_id : bigint, Title : text, Content : text, Version_Created_on : timestamp, Version_Created_by : bigint)
- Answer(Answer_id : bigint PRIMARY KEY, Accepted : Boolean)
- Answer_Version(Version_id : bigint PRIMARY KEY, Answer_id : bigint, Content : text, Version_Created_on : timestamp, Version_Created_by : bigint)
- Comment(Comment_id : bigint PRIMARY KEY)
- Comment Version (Version_id : bigint PRIMARY Key, Comment_id : bigint, Content : text, Version_Created_On : timestamp, Version_Created_by : bigint)
- Tags(Tag_id : bigint PRIMARY KEY, Name : varchar, Description : text, Created_by : bigint, Community_id : bigint, Created_on : timestamp)
- Tagged Question(id : bigint PRIMARY KEY, tag_id : bigint, Version_id : bigint)
- Post(Post_id : bigint PRIMARY KEY, Created_by: bigint, Created_on :

timestamp)

- Review(Reaview_id : bigint PRIMARY KEY, Reviewed_on timestamp, Type : varchar, Reviewed_by : bigint, Post_Id : bigint)

- Badges (Badge_id : bigint PRIMARY KEY, Badge_name : varchar, CM_id : bigint)

VI. ER DIAGRAM



VII. BULK LOADING FROM CSV

To test the db we have written a script to load the data into the Database. This is done with the following files:

- QuestionDetails.json: Contains info of top 500 question rom stackoverflow
- AnswerDetails.Json: Contains answer info for the top 500 question that were populated
- Tags.Json: contains top 1000 tags from stackoverflow based on the usage to be linked with questions.

VIII. CREATING TABLES

1) COMMUNITY

```
CREATE TABLE IF NOT EXISTS Community
(
    "Community_id" bigint
    PRIMARY  KEY,
    "Name" varchar(40) NOT NULL,        "type"
    varchar(30) check ("type" in ('Public',
    'Private')),
    "Created_on" timestamp WITH TIME ZONE
    default CURRENT_TIMESTAMP,
    "Created_by" bigint
);
```

2) USER

```
CREATE TABLE IF NOT EXISTS "User"
(
        "User_id" bigint PRIMARY KEY,
        "Name" varchar(60) NOT NULL,
        "Email Address" varchar(80) NOT NULL
);
```

3) COMMUNITY MEMBER

```
CREATE TABLE IF NOT EXISTS
"Community Member"
(
        "CM_id" bigint PRIMARY KEY,
        "User_id" bigint NOT NULL,
        "Community_id" bigint NOT NULL,
        "Reputation" int DEFAULT 0,
        "Badge Count" int DEFAULT 0,
        "isAdmin" boolean  DEFAULT false,
        "isRemoved" boolean DEFAULT false
);
```

4) VOTE

```
CREATE TABLE IF NOT EXISTS Vote
(
        “id” bigint PRIMARY KEY,
        "Vote_type" varchar(5),
        "Voted_on" timestamp WITH TIME
            ZONE DEFAULT
            CURRENT_TIMESTAMP,
        "Voted_by" bigint,
        "Voted_post" bigint
);
```

5) QUESTION

```
CREATE TABLE IF NOT EXISTS Question
(
        "Question_id" bigint PRIMARY KEY,
```

```
        "Protected?" boolean,
        "Reviewed" varchar(30) check
        ("Reviewed" in ('Accepted', 'Rejected'))
    );
```

6) QUESTION_VERSION

```
    CREATE TABLE IF NOT EXISTS "Question
    Version"
    (
          "Version_id" bigint PRIMARY KEY,
          "Question_id" bigint NOT NULL,
          "Title" text, "Content" text,
        "Version Created_on" timestamp WITH
              TIME ZONE DEFAULT
              CURRENT_TIMESTAMP,
        "Version_Created_by" bigint
    );
```

7) ANSWER

```
    CREATE TABLE IF NOT EXISTS Answer
    (
        "Answer_id" bigint PRIMARY KEY,
        "Accepted" boolean
    );
```

8) ANSWER_VERSION

```
    CREATE TABLE IF NOT EXISTS "Answer
Version"
    (
        "Version_id" bigint PRIMARY KEY,
        "Answer_id" bigint NOT NULL,
        "Content" text,
        "Version_Created_on" timestamp
              WITH TIME ZONE DEFAULT
              CURRENT_TIMESTAMP,
        "Version_Created_by" bigint
   );
```

9) COMMENT

```
    CREATE TABLE IF NOT EXISTS Comment
    (
         "Comment_id" bigint PRIMARY KEY
    );
```

10) COMMENT_VERSION

```
      CREATE TABLE IF NOT EXISTS
      "Comment Version"
      (
        "Version_id" bigint PRIMARY KEY,
        "Comment_id" bigint NOT NULL,
        "Content" text,
        "Version_Created_on" timestamp
              WITH TIME ZONE DEFAULT
              CURRENT_TIMESTAMP,
        "Verson_Created_by" bigint
      );
```

11) TAGS

```
    CREATE TABLE IF NOT EXISTS Tags
    (
        "Tag_id" bigint PRIMARY KEY,
        "Name" varchar(20),
        "Description" text,
        "Created_by" bigint,
        "Community_id" bigint,
        "Created_on" timestamp
              WITH TIME ZONE DEFAULT
              CURRENT_TIMESTAMP
    );
```

12) TAGGED QUESTION

```
    CREATE TABLE IF NOT EXISTS
    "Tagged Question"
    (
        “id” bigint PRIMARY KEY,
        "Tag_id" bigint,
        "Version_id" bigint
     );
```

13) POST

```
    CREATE TABLE IF NOT EXISTS Post
    (
        "Post_id" bigint PRIMARY KEY,
        "Created_by" bigint,
        "Created_on" timestamp
              WITH TIME ZONE DEFAULT
              CURRENT_TIMESTAMP
     );
```

14) REVIEW

```
CREATE TABLE IF NOT EXISTS Review
(
        "Review_id" bigint PRIMARY KEY,
        "Reviewed_on" timestamp
              WITH TIME ZONE DEFAULT
              CURRENT_TIMESTAMP,
        "Type" varchar(20),
        "Reviewed_by" bigint,
        "Post_id" bigint
);
```

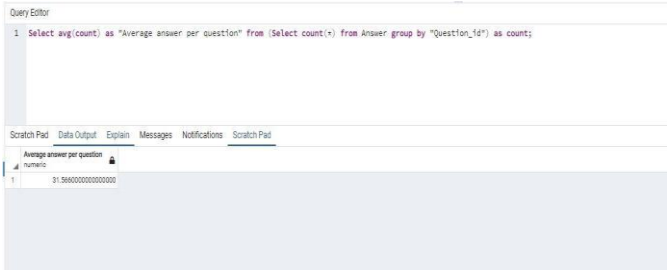
15) BADGES

```
CREATE TABLE IF NOT EXISTS Badges
(
        "Badge_id" bigint PRIMARY KEY,
        "Badge_name" varchar(30),
        "Badge_type" varchar(30),
        "CM_id" bigint
 );
```

IX. QUERY EXECUTION

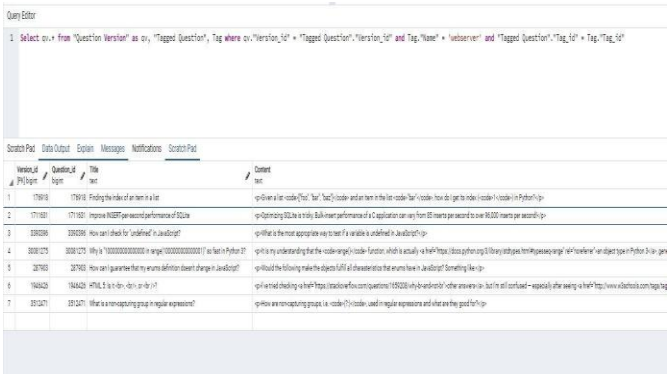
1. Average answers per questions

Select avg (count) as "Average answer per question" from (Select count(*) from Answer group by "Question_id") as count;



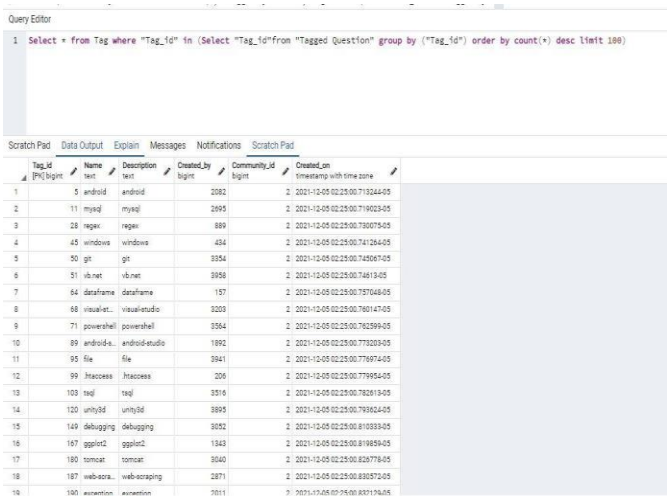
2. Question with the tag 'webserver' and inner join

Select qv.* from "Question Version" as qv, "Tagged Question", Tag where qv. "Version id" = "Tagged Question". "Version id" and Tag. "Name"= 'webserver" and "Tagged Question". "Tag_id" = Tag. "Tag_id"



3. Top 100 tags based on count/usage

Select * from Tag where "Tag_id" in (Select "Tag id"from "Tagged Question" group by ("Tag id") order by count(*) desc limit 100)



4. User with most number of answer posted

Select u.*, cm.* from "Community Member" as cm, "User as u where u."user id"= cm."user_id" and cm. "cm_id in
(Select "Answer Version". "Version_Created_ by" from "Answer Version" group by ("Version_Created_by") order by (count (*)) limit 1)

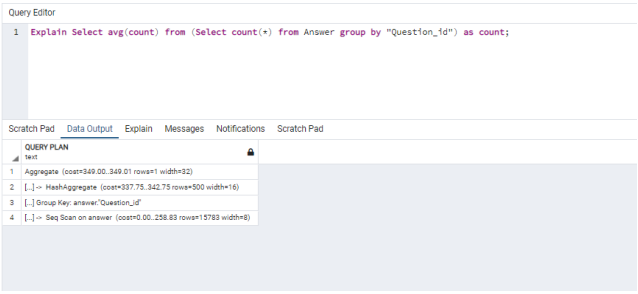
X. BOYCE-CODD NORMAL FORM

Boyce–Codd Normal Form (BCNF) is based on functional dependencies that take into account all candidate keys in a relation; however, BCNF also has additional constraints compared with the general definition of 3NF.None of the attributes are dependent on any other column except for the primary key, hence primary key only functionally determines dependency on every other attribute in the relation. The FD in our data model are as follows

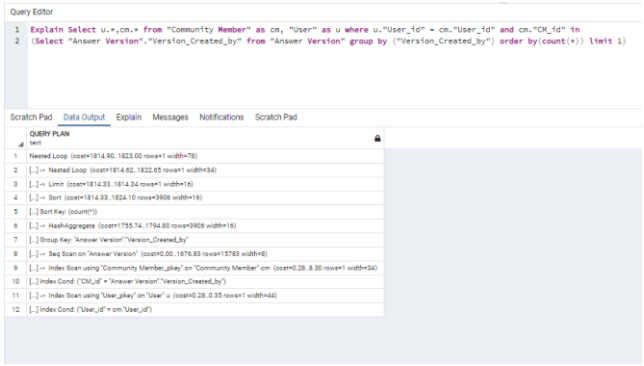
- Community: id → Community name, Public/Private, Created_on, created_by
- User : User_id → Name, Email Address
- Community Member : CM_id → User_id, Community_id, Reputation, Badge Count, isAdmin, isRemoved.
- Vote : id → Vote_type, Voted_on, Voted_by, voted_post.
- Question : Question_id → Protected?, Reviewed
- Question_version : version_id → question_id, title, content, version_created_on, version_created_by
- Answer : Answer_id → Accepted
- Answer_version : version_id → answer_id, title, content, version_created_on, version_created_by
- Comment : comment_id
- Comment_version : version_id → comment_id, title, content, version_created_on, version_created_by
- Tags : tag_id → Name, Description, Created_by, Community_id, Created_on
- Tagged_Question

Hence, all tables in the schema are BCNF Compliant.

XI. Query Execution Analysis

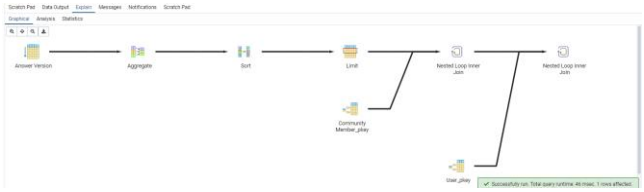


The above query is used to get the average number of answers for every question. Analyzing the query, we can see that the aggregate operation has the highest cost of 349. The sequential scan of answers cost is negligible for 15783 rows

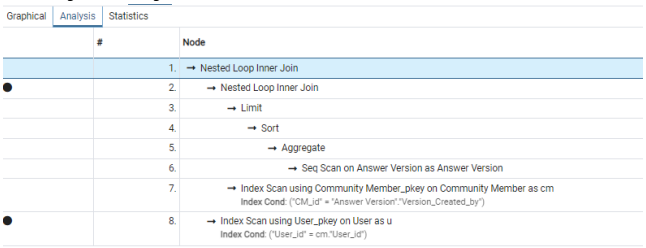


The above query is to get the top user in terms of number of answers posted. In this we can see that subquery execution has a high cost of operation and can be improved. Since all the primary keys are indexed (All Ids), the operation cost for those are very small

The graphical view of the query execution plan



Query Analysis:



XII. CONTRIBUTION OF TEAM MEMBERS

ARAVIND BALAKRISHNAN -

- Creating tables for the dataset
- Inserting data in tables created
- Report writing
- ER Diagram
- PowerPoint presentation content

SAI PRASHANTH SELVAGANAPATHY -

- Data model design
- Bulk loading
- Writing queries
- Query execution and analysis
- Report writing
- Testing and debugging queries

XIII. FUTURE WORK

1. Our Database will be developed with a UI built using HTML and CSS to showcase the use of our database.
2. Will be available for different use cases