Udiddit, A Social News Aggregator

Part: 1

Investing The Existing Schema

· In the given schema, there is no index and many constraints so it leads to duplication of datas.

· The provided tables bad_posts and bad_comments are in denormalized format,for example,the upvotes and downvotes contain comma separated values which is violating the first normal form.

· The schema does not facilitate the recording of timestamp associated with the posts, comments and topics, which is essential for any social news aggregator.

Part: 2

Create the DDL for New Schema

```
create table "users"(
"id" serial primary key,
"username" varchar(25) unique not null check
(length(trim("username")) <= 25),
"login" timestamp
);
```

```
create index "user by_username" on "users"("username")
```

```
create table "topics"(
"id" serial primary key,
"user_id" integer,
"topic_name" varchar(100) unique not null check
(length(trim("topic_name"))<=100 ),
"description" varchar(500),
"topic_created_on" timestamp);
```

```
create index "topics_by_name" on "topics"("topic_name");
```

Several users have created a topic under the same name. Hence, in order to make the topic_name unique, I have concatenated topic_name with the username. This has led to increasing the size of topic_name to 100.

```
create table "posts"(
"id" serial primary key,
"user_id" integer,
"topic_id" integer,
 "title" varchar(100) not null check(
 length(trim(title))<=100),
"url" varchar,
"text_content" varchar
  check(
("url" is null and "text_content" is not null)
     or
("url" is not null and "text_content" is null)
 ),
 "post_created_on" timestamp
);
```

```
alter table "posts" add  foreign key("topic_id")references "topics" on
delete cascade;
alter table "posts" add  foreign key("user_id") references "users" on
delete set null;
```

```
create index "posts_by_topic" on "posts"("topic_id");
```

```
create index "posts_by_users" on "posts"("user_id");

create index "posts_by_url" on "posts"("url");
```

```
create table "comments"(
 "id" serial,
  "user_id" integer,
  "post_id" integer,
  "parent_comment_id" serial,
   "comment_created_on" timestamp,
    primary key("id","parent_comment_id")
 );
```

```
  alter table "comments" add  foreign key ("post_id")references "posts" on
delete cascade;

    alter table "comments" add foreign key("user_id") references "users" on
delete set null;
```

```
create index "comments_by_post"on comments("post_id");

create index "followers_of_parentcomment" on comments("parent_comment_id");

create index "comments_by_users" on comments("user_id");
```

```
 create table "vote"(
"id" serial primary key,
"user_id" integer,
"post_id" integer,
 "vote" integer check("vote"= 1 or "vote"= -1)
);
```

```
 alter table "vote" add  foreign key("user_id")references"users" on delete
set null;
  alter table "vote" add foreign key("post_id")references"posts" on delete
cascade;
```

Part:3

Migration of Data

Users table

```
insert into users(username, login)
select distinct t.name, now() from (
select distinct username as name from bad_posts
union
select distinct regexp_split_to_table("upvotes",',') as name
from bad_posts
union
select distinct regexp_split_to_table("upvotes",',') as name
from bad_posts
union
select distinct username  as name from bad_comments)t;
```

Topics table

```
insert into topics (user_id,topic_name,topic_created_on)
select t.user_id, t.topic_name ||'_by_'||t.username,now()
from(
select distinct u.id as user_id, u.username as username,
```

```
bp.topic as topic_name
from bad_posts bp
join users u on u.username =bp.username)t;
```

Posts table

```
insert into posts
("user_id","topic_id","title","url","text_content",
"post_created_on")
select y.user_id, y.topic_id,
case when length(trim(y.title)) > 100 then
substring(trim(y.title), 1, 100)
     else y.title
     end,

y.url, y.text_content,now() from
(
     select id as topic_id, x.user_id, x.title, x.url,
x.text_content from topics t join (
     select u.id as user_id, title, url, text_content,
topic||'_by_'||p.username as topic_user from bad_posts p join
users u on p.username = u.username)x
     on t.topic_name = x.topic_user

)y;
```

Comments table

```
 insert into
"comments"("user_id","post_id","comment_created_on")
select u.id as user__id,p.id as post_id,now()
from bad_comments bc
join users u
```

```
on u.username = bc.username
join posts p
on p.id= bc.post_id;
```

Vote table

```
insert into vote("user_id","post_id","vote")
select u.id, t.id, t.vote from
(select id, regexp_split_to_table(upvotes, ',') as user, 1 as
vote from bad_posts
union
select id, regexp_split_to_table(downvotes, ',') as user, -1 as
vote
from bad_posts )t
join users u
on u.username= t.user;
```