Assignment 1 – Part 2

Table users:
userId varchar(12) primary key
fullName varchar(100)
nickName varchar(50)
city varchar(80)
state varchar(80)
country varchar(80)
emailId varchar(50)
age number

Table users_follows:
userId varchar(12) primary key foreign key references users(userId)
follows varchar(12) primary key

Table private_message
userIdS varchar(12) primary key foreign key references users(userId)
userIdR varchar(12) primary key foreign key references users(userId)
dateTime date primary key
message varchar(252)

Table public_message
tweetId varchar(20) primary key
tweetText varchar(252)
dateTime date
latitude char(12)
longitude char(12)
userId varchar(12) foreign key references users(userId)

Table reposts
originalTweetId varchar(20) primary key foreign key references public_message(tweetId)
repostTweetId varchar(20) primary key foreign key references public_message(tweetId)

Table public_message_hashTags
tweetId varchar(20) primary key foreign key references public_message(tweetId)
hashTag varchar(20) primary key

Assignment 1 – Part 4

We can have a separate table just to store the count of the reposted tweets.

The schema would look like:
Table repost_tweets_count
tweetId varchar(20) primary key foreign key references public_message(tweetId)
count number

This table will give us the count of how many times a tweet is reposted. This table will also include the tweets that have never been reposted, with their count = 0. So when, a tweet is reposted, we just need to increment the counter by 1 instead of having to perform an insertion.

Advantages:

- Easy to get the count of reposted tweets if that is what our application is mainly dealing with.
- Insertion cost reduces since all the tweets have an entry as soon as their posted for the first time and later only the count needs to be increased.

Disadvantages:

- Overhead of a new table will increase the overall overhead.
- If the application has nothing to do with the count of reposted tweets, maintaining this table would be meaningless.