Databases Project

Design issues with the giant table

The giant table containing many rows of data, each of which relating to all the information about one tweet, is initial in First Normal Form. Every attribute takes only a single value and every attribute is atomic. The giant table has quite a few functional dependencies that need to be taken into consideration in order to normalise this table into BCNF or 3NF form. Some of the FD's that have been ignored in this big table are listed below:

user_id $\Rightarrow$ user_name, user_screen_name, user_location, uset_utc_offset, user_time_zone, user_follower_count, user_friend_count, user_lang, user_description, user_status_count, user_created_at

user_time_zone $\Rightarrow$ user_utc_offset

tweet_id $\Rightarrow$ text, created_at, retweet_count (tweet_id is the primary key so I could list everything here technically)

When normalising this giant table, it would be better to normalise to 3NF instead of BCNF here as when we normalise to BCNF in this table it isn't possible to achieve dependency preservation. An example of this is that after we split the giant table into a relation called users (which I will explain more about later) we can further split this users relation with an FD user_location $\Rightarrow$ user_time_zone, user_utc_offset. Here the FD is not a super-key for the users relation but it could split the data. This FD isn't used to split the data as the user_location data isn't very clean as people can put non-geographic locations in this field making the contents of this field fairly useless.

Another clear design issue was the hashtags (1-6) attributes. Having the hashtags attributes in this form is very computationally inefficient and a waste of memory, due to there being many empty values. Storing hashtags in this way also makes it very hard to query over the hashtags to find certain information.

Redesign

Tweet(__tweet_id__, text, created_at, retweet_count, tweet_source, user_id) user_id FK to users(user_id)
Users(__user_id__, user_name, user_screen_name, user_location, user_follower_count, user_friend_count, user_lang, user_description, user_created_at, user_time_zone) user_time_zone FK to TimeZone(time_zone)
TimeZone(__time_zone__, utc_offset)

Hashtags(tweet_id, hashtag) tweet_id FK to tweet(tweet_id)

ReplyTweet(__tweet_id__, in_reply_to_status_id) tweet_id FK to tweet(tweet_id), (in_reply_to_status_id FK to tweet(tweet_id))

Retweet(__tweet_id__, retweet_of_tweet_id) tweet_id FK tweet(tweet_id), retweet_of_tweet_id KF to tweet(tweet_id)

In the schema ReplyTweet I have put the second FK in_reply_to_status_id in brackets as in theory this is correct, however it isn't possible to create this foreign key in practice as some of the tweets that are being replied to are not part of the big_giant_table database.

The Hashtags schema has the tweet_id's and hashtags, where each hashtag is given and the tweet_id that the hashtag came from is also given. This is done by adding the non-null values from all the hashtag tables into one table. The primary key of thius schema is an OID as there is no super-key in this schema.

Results from queries

Tweets Users and Languages

Total tweets = 110574

Retweeting habits

Fraction of tweets that are retweeted = 0.194

Average retweets per tweet = 121

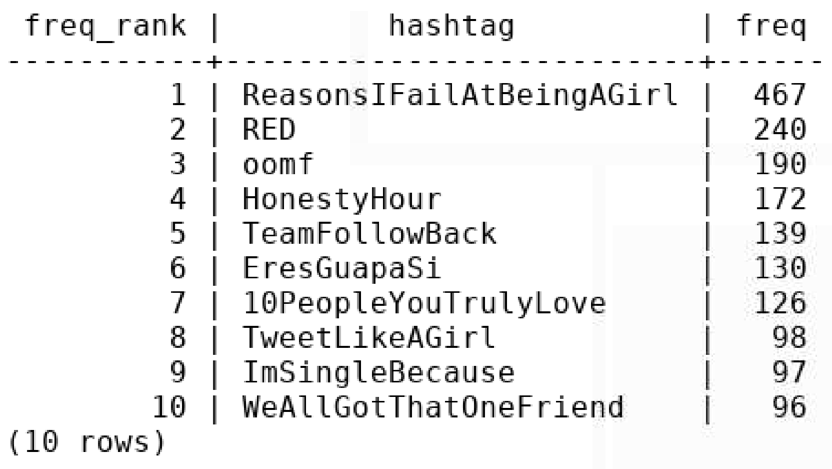
Fraction of tweets never retweeted = 0.669

Fraction of tweets that are retweeted fewer times than the average number of retweets = 0.936
Here the proportion of tweets that have less retweets than the average number of tweets shows that the distribution of the number of retweets has a right skew. Very few tweets have more retweets than the average, but because some tweets have large numbers of retweets, this right skew occurs.

Hashtags

Distinct hashtags = 10158

Top 10 most popular hashtags:



```
 freq_rank |         hashtag          | freq
-----------+--------------------------+------
         1 | ReasonsIFailAtBeingAGirl |  467
         2 | RED                      |  240
         3 | oomf                     |  190
         4 | HonestyHour              |  172
         5 | TeamFollowBack           |  139
         6 | EresGuapaSi              |  130
         7 | 10PeopleYouTrulyLove     |  126
         8 | TweetLikeAGirl           |   98
         9 | ImSingleBecause          |   97
        10 | WeAllGotThatOneFriend    |   96
(10 rows)
```

Replies

Tweets that are neither replies nor replied to = 0.794

Probability that user1 replies to user2 in same language = 0.882

Probability two arbitrary users have the same language setting = 0.424
The probability of two arbitrary users having the same language setting is much lower than the probability that user1 replying to user2 have the same language setting. This makes sense as there are many potential languages that two arbitrary users could speak however, if a user is replying to a tweet, that implies that they understand what the tweet says meaning they will probably reply in the same language.