Report

Design issues with the giant table design:

1. The User information and Tweet Information is not directly dependent, still, it is given altogether in one row, which can slow down query execution and can lead to inconsistency.
2. The table can lead to huge data redundancy because storing the tweet and user data together.
3. All the Tweet data depends on tweet_id so needs to be a separate table.
4. Similarly, all the User data depend on the user_id so needs to be in another table.
5. If we want to add another tweet with the same user_id we will need to enter all the details again of the same user again, causing a lot of data repetition.
6. On the other hand, if we want to delete a particular tweet, and only one user_id is associated with it, we will lose all the user data along with the tweet data.
7. A user is allowed to use as many hashtags as he wants but the table can store only 6, and if he does not enter all the hashtags the columns will contain null values causing a lot of memory wastage.
8. The retweet of tweet id can be null if the tweet is not retweeted.
9. If a tweet is replied, it is associated with 3 attributes depend on it, so should be maintained in a separate table to avoid null values.
10. If the user does not add location and user description it contains null values.
11. If time zones are provided then only it has offset, so offset contains null values again if time zones are not provided.

Functional Dependencies:

tweet_id -> created_at, text, retweet_count, tweet_source,
retweet_of_tweet_id, in_reply_to_screen_name,
in_reply_to_status_id, in_reply_to_user_id,
hashtags.

user_id -> user_name, user_screen_name, user_followers_count,
user_friends_count, user_lang, user_status_count,
user_created_at, user_location, user_time_zone

Normalized Design:

1. The new design is in 3NF removing all the inconsistencies in the previous database.
2. The tweets table consists of only the not null values created_at, text, tweet_source, and the retweet_count maintaining the tweets data using tweet_id as a primary key.
3. If a tweet is retweeted then only it contains retweet_of_tweet_id, so a separate table named retweet is created having tweet_id referencing to tweet_id in tweets table

4. If a tweet is replied then it maintains 3 attributes, in_reply_to_screen_name, in_reply_to_status_id, in_reply_to_user_id, so they are in a separate table tweetReplys referencing to tweet_id in tweets table.
5. We have a separate table for hashtags that stores every hashtag with a tweet_id. For example: If a tweet with tweet_id X has 2 hashtags then 2 rows are entered in the table. First row with X and Hashtag1, second with X and Hashtag2. So, this stores only provided hashtags and there are no null values at all which saves a lot of memory and is an efficient way to store this multivalued attribute.
6. We just maintain the basic user information in the users table with user_id as the primary key
7. Not every user specifies his location so being optional attribute, we use a separate table userLocations to store location referencing to user_id, so that we store only the location of users who specify it.
8. We store time zones and corresponding offset in userTimeZones table. Attribute offset is dependent on time zone, so these 2 attributes are maintained in a different table to avoid null values.
9. All the users do not specify a description, so we store only the ones who specify their descriptions in a table userDesc.
10. Finally, in the tweets_users, we link the tweet_id with respective user_id showing the relation between 2 in a very simplified manner, where user_id and tweet_id are foreign keys referencing the tweets and users table

Conclusion:

- The table is in 3NF because we separated all the attributes which are not directly dependent on the primary key.
- For an instance, the offset is dependent on the time zone, so should be better stored in the time zones table.
- If not, the offset will be duplicated in the database if we have the same time zone.
- We maintain all the optional attributes in a separate table to avoid all the null values.
- Every relation schema in the database has a trivial dependency, for example, the tweet table has superkey tweet_id that contains created_at, text, tweet_source, and retweet_count. Tweet_id is superkey and each attribute in set difference forms a candidate key. Thus, we can say that the schema is in 3NF. Similarly, all other schemas are normalized.