Twitter Clone Report

Sean Egger and Aaron Millet

Data Model

Our Twitter clone uses five tables: Users, Tweets, Retweets, Favorites, and Following.
Users contains an ID as a primary key and a name for the user.
Tweets contains an ID as a primary key, a tweet author which is a foreign key to the ID in Users, a message containing the tweet text, a date which is the date and time the tweet was written, num_favorites as the number of times the tweet was favorited, and num_retweets for the number of times a tweet was retweeted.
Retweets contains a tweet ID as part of a composite primary key which is also a foreign key to the ID in Tweets that was retweeted, a user ID as part of a composite primary key which is also a foreign key to the ID in Users of who retweeted the tweet, and a date which is the date and time the tweet was retweeted.
Favorites contains a tweet ID as part of a composite primary key which is also a foreign key to the ID in Tweets that was favorited, a user ID as part of a composite primary key which is also a foreign key to the ID in Users of who favorited the tweet, and a date which is the date and time the tweet was favorited.
Following contains a follower ID as part of a composite primary key which is also a foreign key to the ID in Users of who is following someone, a following ID as part of a composite primary key which is also a foreign key to the ID in Users of who is being followed by someone, and a date which is the date and time the follower started following the other user.

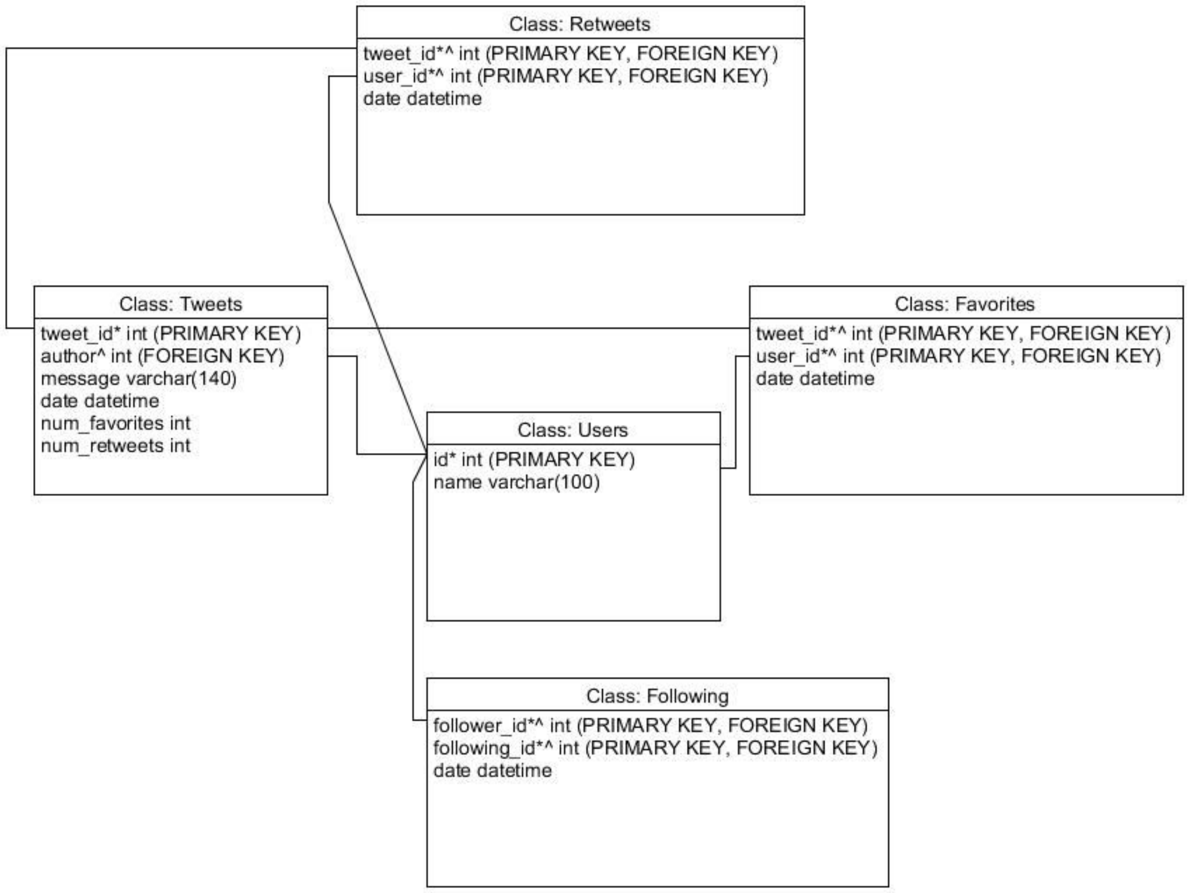
Justification

We believe this data model is suitable for the application laid out in the assignment. It allows all of the necessary functionality while keeping the data model fairly compact.
It should scale well, as every type of insertion be it adding a user, tweeting, retweeting, favoriting, or following another user adds at most one row to a single table. Additionally, most tables contain a fairly small amount of data or act primarily as intermediary tables, so the amount of data added for each time of relational addition is still quite small.
The queries needed to gather any type of data needed are typically only a few lines, with the exception of the timeline fetching query. Even the timeline query is essentially two smaller queries made into an aggregate table and is fairly simple.
Composite primary keys were used in cases where the only data needed in the table for key values consisted of two other always unique data points. This allows for easy searching of the data by column, while still making it simple to find an individual row if the two key data points are known.
Overall, this is a fairly compact and efficient solution to a basic Twitter data model.

Timeline Explanation

The timeline query is the most complex query needed for three reasons. It requires filtering data to make sure it only collects data points related to a single user. It also requires multiple join statements to link the ID for each user followed by a single user to their name, to each tweet by

all of those users, and to each retweet by all of those users. Lastly, and most complex, it has to include adding retweets into the timeline and sorting them them in with other tweets.
Most parts of the query are simple, but to accomplish the last part of the task, adding in retweets, using two subqueries and creating an aggregate table was necessary. This is because we felt it necessary to include both the user which retweeted the original tweet and the date and time they retweeted. Otherwise there would be no way to distinguish a retweet from a tweet, and there would end up being multiple instances of the same tweet grouped together in the timeline. Including the retweet date and time also allows sorting by when the tweet was retweeted if it is a retweet, rather than by when it was originally tweeted.
Note that the timeline could have included less data, such as by omitting ID values and using only user names, messages, and dates. It was left fairly verbose for simplicity and to check relevant data. When a frontend is created, we can display only data a user would want to see.


Class: Retweets
tweet_id*^ int (PRIMARY KEY, FOREIGN KEY)
user_id*^ int (PRIMARY KEY, FOREIGN KEY)
date datetime
Class: Tweets
tweet_id* int (PRIMARY KEY)
author^ int (FOREIGN KEY)
message varchar(140)
date datetime
num_favorites int
num_retweets int
Class: Favorites
tweet_id*^ int (PRIMARY KEY, FOREIGN KEY)
user_id*^ int (PRIMARY KEY, FOREIGN KEY)
date datetime
Class: Users
id* int (PRIMARY KEY)
name varchar(100)
Class: Following
follower_id*^ int (PRIMARY KEY, FOREIGN KEY)
following_id*^ int (PRIMARY KEY, FOREIGN KEY)
date datetime