1. SQL Basics

October 25, 2013

Directions

- Work in groups of 2 students.
- Write-up a short note (max 2 pages long) to sum up your experiments and focus on singular difficulties of the subject.
- Please, add at the end of your report, the following sentence:

  > "We, ⟨student1⟩ and ⟨student2⟩, attest this is our own work and accept the consequences if it is not."

- Deliverables must be packaged into a *tar.gz* or *zip* file containing a single directory having your short report (PDF file) and the SQL scripts to play back the entire story. Name of the directory must satisfy the pattern:

  ```
  hw1_<name of student1>_<name of student2>
  ```

- The archive must be uploaded on the Moodle web site up to the due date.

Requirements

PostgreSQL (>=9) database system is required to achieve this assignment. It is an Open Source software available for download at `http://www.postgresql.org/download/`. The abundant documentation and active community could help you installing and setting the system if necessary. The *PgAdmin III* client included into the regular install package is the preferred way for the connection to the database.

Due date

2013-10-11 11:59pm

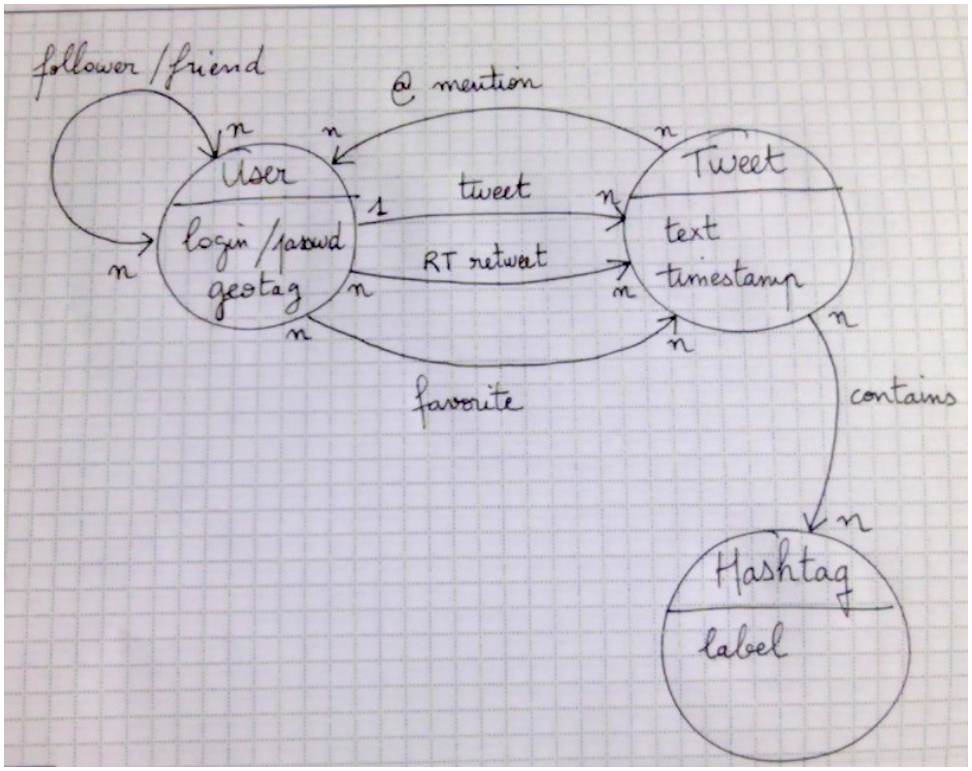

Figure 1: *TinyTwitter* — A (Not So) Simplified Model of Twitter

Introduction

The playground is Twitter, the famous micro-blogging platform. We are trying to develop a restricted copy of Twitter, coined *TinyTwitter*.

The drawing on Figure gives a model for *TinyTwitter*. There are three main entities (user, tweet, hashtag) and several binary relationships among them. A registered user (with username/password) post tweets, i.e. short messages of less than 80 characters each. Tweets may actually be retweets (`RT <tweetid>`) of previous messages from registred users. RT is idempotent, i.e. `RT RT <tweetid>` = `RT <tweetid>`. Users might follow any other registered users, so-called friends. Friend's tweets are highlighted. Tweets admit hashtags and mentions. Hashtags (`#Nantes`) are basically keywords for tweet categorization purpose. Mentions such like `@me` are references to registered users and allow for instance to answer a previous tweet or to focus on your audience.

The above model translates into the set of following relations:

```
User       (uid, username, password, geotag)
Tweet      (tid, text, timestamp, user)
Hashtag    (hid, label)
Tagintweet (tag, tweet)
Mention    (tweet, user)
Retweet    (tweet, retweet)
Follower   (user, friend)
Favorite   (user, tweet)
```

1 DDL part of SQL and more

1. Write SQL `CREATE TABLE` statements to implement the eight relations of the *TinyTwitter* project. Propose accurate types for every field and incorporate integrity constraints such like:
   - *primary keys*, *foreign keys*;
   - field is *not null*;
   - field is *unique*;
   - domain-based and tuple-based constraints (*check*);
2. Write SQL `INSERT` statements to populate the database. The resulting dataset is required to return a non-empty answer set to any query below.

2 DML part of SQL

Write SQL statements to translate the following queries against the above *TinyTwitter* database.

1. (a) List all tweets of user `me` in descending timestamp order.
   (b) List the 10 more recent tweets from friends of `me`.
   (c) Give location of users that tweet about `#Nantes`.
2. (a) Give the average number of followers by user.
   (b) Give the number of tweets a day for the entire last month.
3. (a) List in descending timestamp order both tweets that mention user `me`, and retweets from `me`.
   (b) Find users that never mention their friends.
   (c) Find users that have retweeted all their friends.

Resources & References

1. Chapter *Relational Model* of Lecture Notes (and SQL counterparts)
2. Basic SQL tutorials and quick reference guide
   `http://www.w3schools.com/sql/`
3. Homepage of the PostgreSQL 9.3 documentation
   `http://www.postgresql.org/docs/9.3/static/index.html`
4. Data integrity PostgreSQL document page
   `http://www.postgresql.org/docs/9.3/static/ddl-constraints.html`