Datenbanksysteme
Projekt Iteration 2: Datenimport

Adrian Gruszczynski
Pit Ronk
Remi Toudic
Tutor: Toni Draßdo
Tutorium: Donnerstag 12-14

June 1, 2017

Aufgabe 1: Datenbankschema erstellen

```
1 CREATE TABLE tweet (tweet_id serial primary key, handle
      varchar(20) NOT NULL, body varchar(200) NOT NULL,
     time timestamp NOT NULL, retweet_count int,
     favorite_count int);
2
3 CREATE TABLE hashtag(hash_id serial primary key,
     tweet_id int, text varchar(100) NOT NULL);
```

Private repository:
`https://gitlab.spline.inf.fu-berlin.de/h4rrytrum4n/DBS_PR/blob/master/it_02/src/a1.sql`

Aufgabe 2: Datenbereinigung

```
1  import csv
2  import re
3
4  def cleanData():
5      tweetID, hashCount, ahshWritten = 0, 0, 0;
6      hashtags = []
7
8      #Open files for read/write
9      rawData=open('../data/tweets_raw.csv', 'rt')
10     hashtagCSV=open('../data/hashtag.csv', 'wt')
11     tweetCSV=open('../data/tweet.csv', 'wt')
12
13     try:
14         #Prepare csv read/write iterators
15         reader = csv.reader(rawData, delimiter=';')
16         writerHashtag = csv.writer(hashtagCSV,
               delimiter=';')
17         writerTweet = csv.writer(tweetCSV, delimiter=';
               ')
18         #Omit the first row in the given data
19         next(reader)
20         #Iterate through each row of the given data
21         for row in reader:
22             #Extract all the relevant tweet
                   informations
23             #handle, text, time, retweet_count,
                   favorite_count
24             writerTweet.writerow((row[0], row[1], row
                   [4], row[7], row[8]))
25             #See if there're any #'s at all in the
                   tweet
26             #Count every occurence if true
27             match = len(re.findall('#', row[1]))
28             if(match):
29                 hashCount = hashCount + match
```

```
30                  #Substitute whitespaces after #
31                  row[1] = re.sub('# ', '#', row[1])
32                  #Extract every # into an temporary
                        array
33                  hashtags = re.findall(r'#\w+', row[1])
34                  #Save the extracted #'s and their
                        corresponding TweedID
35                  #Count each # that is written
36                  for ahsh in hashtags:
37                      writerHashtag.writerow((tweetID,
                            ahsh))
38                      ahshWritten = ahshWritten +1
39              tweetID = tweetID+1
40      finally:
41              #Close all filedescriptors
42              rawData.close()
43              hashtagCSV.close()
44              tweetCSV.close()
45
46      print '%s %d' % ('Occurences of #\'s in the given
            data: ', hashCount)
47      print '%s %s' % ('Extracted #\'s count: ',
            ahshWritten)
```

Private repository:
`https://gitlab.spline.inf.fu-berlin.de/h4rrytrum4n/DBS_PR/blob/master/it_02/src/clean_data.py`

Aufgabe 3: Daten import

```
1  import psycopg2
2  import csv
3  import re
4
5  def importTweets():
6      tweetsImported = 0;
7
8      #Open and prepare the csv data for read
9      tweetCSV = open('../data/tweet.csv', 'rt')
10     tweetReader = csv.reader(tweetCSV, delimiter = ';')
11
12     #Open connection to the PostgreSQL DB.
13     try:
14         conn = psycopg2.connect("dbname='election' user
               ='postgres' host='localhost' password='
               postgres'")
15         print "Connection sucessful"
16     except:
17         print "Connection failed"
18         return
19
20     #Create a cursor which will be used for DB
           programming
21     cur = conn.cursor()
22     #Prepare the insertion query
23     queryTweet = """INSERT INTO tweet (handle, body,
           time, retweet_count, favorite_count)
24         VALUES (%s, %s, %s, %s, %s);"""
25     #Iterate through each row in the given data
26     #Prepare the values that are going to be inserted
27     for entry in tweetReader:
28         vals = (entry[0], repr(entry[1]), entry[2],
               entry[3], entry[4], )
29         try:
30             #Execute query and commit changes
```

```
31              cur.execute(queryTweet, vals)
32              conn.commit()
33              tweetsImported = tweetsImported + 1
34          except:
35              print "Query failed!"
36      #Close the cursor and the connection
37      cur.close();
38      conn.close();
39      print '%s %d' % ("Imported tweets: ",
            tweetsImported)
40
41
42  def importHashtags():
43      hashImported = 0;
44
45      #Open and prepare the data for reading
46      hashCSV = open('../data/hashtag.csv', 'rt')
47      hashReader = csv.reader(hashCSV, delimiter = ';')
48
49      #Open connection to the PostgreSQL DB.
50      try:
51          conn = psycopg2.connect("dbname='election' user
                ='postgres' host='localhost' password='
                postgres'")
52          print "Connection sucessful"
53      except:
54          print "Connection failed"
55          return
56
57      #Create a cursor which will be used for DB
            programming
58      cur = conn.cursor()
59      #Prepare the insertion query
60      queryHash = """INSERT INTO hashtag (tweet_id, text)
            VALUES (%s, %s);"""
61
62      #Iterate through each row of the given data
63      #Prepare values that are going to be inserted
```

```
64      for entry in hashReader:
65          vals = (entry[0], entry[1], )
66          try:
67              #Execute the query and commit changes
68              cur.execute(queryHash, vals)
69              conn.commit()
70              hashImported = hashImported + 1
71          except:
72              print "Query failed!"
73      #Close the cursor and the connection
74      cur.close()
75      conn.close()
76      print '%s %d' % ("Imported hashtags: ",
            hashImported)
```

Private repository:
`https://gitlab.spline.inf.fu-berlin.de/h4rrytrum4n/DBS_PR/blob/master/it_02/src/import_data.py`

Aufgabe 4: Webserver

Für die Webanwendung haben wir uns entschieden Django Framework zu benutzen.

Django ist ein Python basiertes Webframework das einem Model-View-Present Schema folgt. Es verfügt über eine integrierte Schnittstelle für die Anbindung von mehreren Datenbanken u.a PostgreSQL. Das Webframework an sich läuft auf einem Apache Webserver.

Sobald unsere Datenbank eingerichtet wurde, könenn wir mit Installation von Django anfangen. Für verbesserte Flexibilität werden wir die Einrichtung von Django und dessen Abhängigkeiten in einem virtual enviroment einrichten.

Zunächst werden folgende packages installiert:

```
sudo apt install python-pip python-dev libpq-dev
sudo pip install virtualenv
```

Als nächstes wird ein Projektordner erstellt:

```
mkdir django-postgresql
cd django-postgresql
```

Nun wird ein virtual enviroment für unseres Django Projekt angelegt und aktiviert:

```
virtualenv django-postgresql
source django-postgresql/bin/activate
```

Sobald unser virtual enviroment aktiv ist, kan mann mit der Installation von *psycopg*2 anfangen. Das Paket ermöglicht uns den Aufbau einer Verbindung zu unserer Datenbank.

```
pip install django psycopg2
```

Als nächstes kann das Projekt angelegt werden:

```
django-admin.py startproject django-postgresql .
```

Wenn unser Projekt erfolgreich angelegt wurde, kann man mit der Konfiguration der Datenbank beginnen. Dafür öffnen wir die folgende Konfigurationsdatei und suchen nach dem *DATABASES* Abschnitt:

```
vim django-postgresql/settings.py
```

Die ursprüngliche Konfiguration muss folgendermaßen angepasst werden:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'election',
        'USER': 'postgres',
        'PASSWORD': 'postgres',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

Wenn die Konfiguration der Datenbankverbindung erfolgreich gelaufen ist kann man jetzt die bestehende Relationen aus unserer Election Datenbank als Modell importieren:

```
python manage.py inspectdb > models.py
pthon manage.py makemigrations
python manage.py migrate
```

Nachdem die Struktur für die Datenbank aufgebaut wurde kann man ein administratives Konto erstellen:

```
python manage.py createsuperuser
```

Zum Schluss starten wir unseren Webserver mit folgendem Befehl:

```
python manage.py runserver localhost:8888
```