Freie Universität Berlin

Prof. Dr. Agnès Voisard, Nicolas Lehmann

Datenbank Systeme, SoeSe 2017

Project 1

TutorIn: Hoffman Christian
Tutorium Tutorium 3

Ingrid Tchilibou, Emil Milanov, Boyan Hristov

30. Mai 2017

---

1 Aufgabe: Projektdokumentation

Link zum Projekt: https://github.com/gancia-kiss/dbs_projekt

Das Team: Emil Milanov, Boyan Hristov, Ingrid Tchilibou
Wir sind alle Bachelor Informatik Studenten im 4. Semester.

Projektziel
Unserer Meinung nach hat dieses Projekt als Ziel Verhältnisse zwischen Tweets und Hashtags oder zwischen Paaren von Hashtag zu analysieren und auch grafisch darzustellen. Laut der Aufgabenstellung soll das eine Web-Anwendung sein, was wir uns merken müssen, ist aber in der ersten Projektiteration nicht wichtig.

Aufgabenverteilung
Alle drei von uns haben gründlich das ER Modell besprochen und mit Aufmerksamkeit eine Eintscheidung getroffen. Ingrid hat zusätzlich die Daten gründlich analysiert, Emil und Ingrid haben das relationales Modell erstellt und Boyan hat das ER Modell gezeichnet und bereinigt, die Datenbank mithilfe von Postgres erstellt und eine Möglichkeit gefunden, eine Dumpdetei zu erstellen, als Beweis, dass wir eine Datenbank fertig haben. Zusätzlich haben Emil und Ingrid die Dokumentationn verfasst.

2 Aufgabe:Explorative Datenanalyse

Wir haben eine Tabelle mit 10 Attributen von verschiedenen Tweets von der 2016 America Election.
Jeder Tupel beschreibt genau :

Von wem wurde den Tweet gepostet? Dafür gibt es den Attribut *Handle*
Was war die Nachricht / Inhalt von diesem Tweet? Dafür ist den Attribut *Text*
Manche posts von Trump und Clinton waren eigentlich Retweets von Tweets von anderen Personen. Dafür wird den Attribut Original Author erstelt. Außerdem hat jeder Tweet einen *Source-url* als Attribut (D.h Was für ein Platform oder Gerät benutzt wurde, um den Tweet zu veröffentlichen)
Wie häufig wurde diesen Tweet weitergeleitet? *retweet-count*

Wie häufig wurde diesen Tweet ğelikedöder gespeichert. *favorite-count* ?

Außerdem gibt es Tweets, die ein Zitat enthalten. Dieses Zität wird als Anhang im Browser angezeigt, aber wird nicht vom API genommen. Solche Tweets sind auch truncated oder abgeschnitten. *is-quote status*
Zusätzlich gibt es auch manche Tweets, die 'truncated' worden sind. Das heißt, dass die ein zitiertes Tweet oder irgendwelche Media enthalten, deswegen überschritten sie das Twitter Zeichenlimit und werden abgeschnitten.

Für uns wären *text*, *handle*, *time*, *retweet-count* und *favourite-count* besonders wichtig, da man mit diesen Daten alle von den gegebenen möglichen Anfragen bearbeiten kann. Das Attribut *text* enthält alle HashTags und es gibt leider kein anderes Attribut dazu, deswegen werden wir die HashTags von diesem Attribut nehmen. Wie schon erwähnt, müssen wir hier auf das Attribut *truncated* aufpassen.

Für die Wichtigkeit des Tweets könnte im unseren Fall auch wichtig sein, ob Trump oder Clinton Tweets weitergeleitet haben. Dafür ist das Attribut *original author*. Öftmals sind das Tweets mit 'trending' Messages/Aussagen.

3 Aufgabe: ER-Modelierung

Alle drei von uns haben dieses Modell gründlich besprochen und eine Entscheidung getroffen. Wir glauben, dass dieses Modell das beste für unsere Vorstellung des Projekts ist.

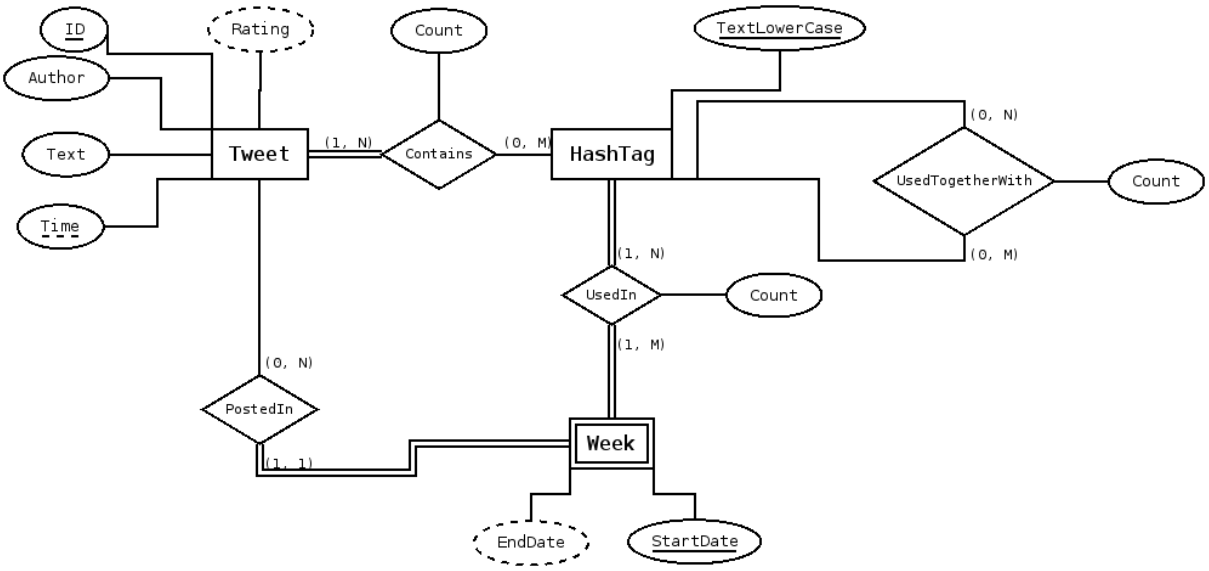
Probleme und Lösungen

- Um 'Welche Hashtags werden am meistens verwendet' zu beantworten, muss man für jeden Hashtag überprüfen, in welchen Tweets er vorkommt und dann diese aufzählen. Deswegen haben wir uns entschieden, in dem Hashtah Entitätstyp ein Attribut TotalCount zu speichern. So werden wir auch weniger Server-Anfragen bearbeiten müssen.
- Der Datenumfang ist über mehreren Monaten. Um die Fragen 'Wann traten insgesammt am meisten Hashtags auf?' und 'Wie hat sich die Häufigkeit der Verwendung eines speziellen Hashtags im Laufe der Zeit entwickelt?' zu beantworten müssen wir jeden Tweet zu einem zeitlichen Bereich zuordnen.

  Wir haben uns deswegen entschieden, einen Entitätstyp 'Week' zu haben, wo wir an einer Woche benutzten Hashtags speichern werden. (Natürlich auch einen Attribut "Count"für höhere effizienz verwenden)
- Unserer Meinung nach, wäre eine Metrik für wichtige Tweets die Summe von Retweets und Favourites. So haben wir einen Attribut 'Rating', der nach der folgende Schema berechnet wird;

  $$retweets * weight_1 + favourites * weight_2 = rating$$

  wobei $weight_1$ und $weight_2$ später genau bestimmt werden.
- Es wird ziemlich aufwändig herauszufinden, welche Paar von Hashtags am häufigsten gemeinsam auftritt, weil wir über alle Tweets iterieren müssen. Deswegen haben wir eine N:M rekursive Relation auf 'Hashtag' erstellt.
- Eigentlich haben wir zu fast jedem Entitätstyp einen Attribut 'count' hinzugefügt, weil die Berechnung von Anzahl viel einfacher zu Parse-Zeit wäre, als jedes mal eine Anfrage zu machen, und dann alle verschiedene Entitäten aufzuzählen.



4 Aufgabe: Relationales Modell

Nachdem wir die ER Diagramm hatten, war es einfach ein relationelles Modell zu erstellen. Aufgrund der Aufgabestellung und unsere Interpretation, sind nicht alle Attribute aus der .csv Datei für uns Relevant. Im 3. Teil der Projektdokumentation sind unsere Probleme, Lösungen und Entscheidungen um den Entitätstypen und deren Attributen zu sehen. Trotzdem war es für uns sehr einfach ein relationelles Modell zu erstellen.

Einige Entscheidungen, die wir bei der Erstellung des relationalen Modell treffen müssten waren z.B in welcher Tabelle die verschiedene 'Count' Attributen sein müssen.

Unsere Haupttabellen sind TWEET, HASHTAG und WEEK, jede von denen einen ID Attribut hat, und HASHTAG.Text wird nur kleine Buchstaben haben.

Wir vermuten auch zum Beispiel, dass Hashtags maximal 40 Zeichen lang sind. Natürlich gibt es längere Hashtags, aber wir glauben dass Trump und Clinton keine solche Hashtags verwenden. Für Handle brauchen wir nicht mehr als 20 Zeichen. Normalerweise sind Tweets maximal 200 Zeichen lang, es gibt aber manche, die Truncated worden sind und es zusätzlich eine Link enthalten, was die Grenze überschritten kann. Deswegen haben wir uns entschieden 200 Zeichen für jeden Tweet zu speichern.

Tweet(ID :: integer, Handle :: character varying(20), Text :: character varying(200),
Time :: date, Rating :: integer, Count :: integer, replyTo :: character varying(20),
isRetweet :: boolean, isQuote :: boolean, isTruncated :: boolean)

Hashtag(TextLowerCase :: char varying(40), TotalCount :: integer)
Week(StartDate :: date, EndDate :: date)

Contains(Tweet.ID, Hashtag.TextLowerCase)
UsedIn(Week.StartDate, Hashtag.ID, Count :: integer)
WeeklyTweers(Week.StartDate , Hashtag.ID)
UsedTogetherWith(HT1.ID, HT2.ID)

5 Aufgabe: Datenbank erstellen

Wir sind nicht sicher wie wir beweisen können, dass wir die Datenbank erstellt haben, deswegen wenn man den Link öffnet, sind ein paar Screenshots zu finden. Außerdem steht im Github-repo eine .sql Datei, wo mann alle bei der Erstellung der Datenbank ausgeführte Anweisungen finden kann.

Trotzdem, nachdem wir ein relationales Modell hatten, war es einfach die Datenbank zu erstellen. Das schwierigste bei diesem Teil der Aufgabe war zu lernen, wie man mit Postgres umgeht.

Wir haben alle Tabellen, als auch die Relationen dazwischen erstellt. Falls das aber nicht notwendig ist, hier sind die notwendige Befehle, die wir ausgeführt haben, um unsere Datenbank zu erstellen aus dem Arch Linux wiki:

sudo -u postgres -i initdb –locale $LANG -E UTF8 -D '/var/lib/postgres/data'

Das wechselt zu dem default postgre Nutzer und erstellt den Datenbank cluster

createuser –interactive hristov

Das erstellt ein postgresql Nutzer. Hier ist es klever das selbe Nutzername als das eigene auf dem Rechner zu nutzen um Authentifizierung zu erleichtern.

createdb Election

Das erstellt ein Datenbank mit dem Namen 'Election'

psql -d Election

So geht man in dem Shell von dem Datenbank rein. Danach kann man SQL Befehle benutzen um Tabellen zu erstellen, Daten aufzurufen und im großen und ganzen alles was man sonst mit SQL machen darf. Am Ende kann man auch die für die Erstellung benutze Befehle exportieren und das ganze Datenbank migrieren. Das haben wir auch als Beweis gemacht.

pg_dump Election > Election.sql