Queries

Folgende Abfragen soll das System unterstützen:

1. Auflisten der Posts, die von einem Account gemacht wurden, bzw. ihm zugeordnet wurden
2. Finden der 100 Accounts mit den meisten Followern
3. Finden der 100 Accounts, die den meisten der Accounts folgen, die in 1) gefunden wurden
4. Auflisten der Informationen für die persönliche Startseite eines beliebigen Accounts (am besten mit den in 2) gefundenen Accounts ausprobieren); die Startseite soll Folgendes beinhalten (als getrennte Queries umsetzen):
   - die Anzahl der Follower
   - die Anzahl der verfolgten Accounts
   - wahlweise die 25 neusten oder die 25 beliebtesten Posts der verfolgten Accounts (per DB-Abfrage)
5. Caching der Posts für die Startseite (vgl. 4), erfordert einen sog. Fan-Out in den Cache jedes Followers beim Schreiben eines neuen Posts
6. Auflisten der 25 beliebtesten Posts, die ein geg. Wort enthalten (falls möglich auch mit UND-Verknüpfung mehrerer Worte)

1. SELECT * FROM twitter.tweets_by_authors WHERE author = 'katyperry';

2. -> Extraktion der Top 100 per Skript und Schreiben in Time-Series-Table
   -> Abfrage dann auf dieser Table
   SELECT follower_json FROM twitter.top100_accounts_by_time LIMIT 1;

3. -> Extraktion der Top 100 "Mainstream-Follower" per Skript und Schreiben in Time-Series-Table
   -> Abfrage dann auf dieser Table
   SELECT follower_json FROM twitter.top100_followers_by_time LIMIT 1;

Das Skript für Query 2 und Query 3 (/runtime_scripts/periodic_script.py) ist als periodisch laufender Batch-Job (z.B unter Verwendung eines Jenkins-Automatisierungsservers) gedacht.

4.
   1. SELECT username, COUNT(*) AS followers FROM twitter.follower_relations_by_users WHERE username = 'katyperry' AND rel_type = 'follower';

   2. SELECT username, COUNT(*) AS follows FROM twitter.follower_relations_by_users WHERE username = 'katyperry' AND rel_type = 'follows';

   3. Mit Python Skript (**/runtime_scripts/init_cache_for_user.py**) den Cache für einen beliebigen User initialisieren. Danach mit folgender Abfrage die 25 neusten Post der verfolgten Accounts des jeweiligen Users aus dem Cache abrufen:
      SELECT * FROM twitter.caches_by_users WHERE username = <username> LIMIT 25;

5. In C* leider nicht mit Queries alleine umsetzbar. Lösung daher mit Python Skript (**/runtime_scripts/insert_new_tweet.py**).

- Neuen Tweet in twitter.tweets_by_authors schreiben
- Follower des Autors aus twitter.follower_relations_by_user extrahieren
- Neuen Tweet jeweils in den Cache jedes Followers in twitter.caches_by_users schreiben

6. Eigentlich bietet Lucene hier die Filter-Types **phrase** und **contains** an, um nach den Vorkommnissen gewisser Keywords zu suchen. Leider gab es hierbei insoweit Probleme, dass unter Verwendung besagter Filter-Types ein **Equals** statt **Contains** abgebildet wurde. Da wir die Ursache des Problems nicht identifizieren konnten, haben wir hier stattdesssen die Filter-Types **wildcard** und **regexp** verwendet, um die gewünschte Funktionalität abzubilden.

   Für einen Begriff:
   ```
   SELECT * FROM twitter.tweets_by_authors WHERE expr(tweets_index,
   '{filter: {type: "wildcard", field: "content", value: "\*hello\*"}, sort: {field:
   "number_of_likes", reverse: true}}') LIMIT 25;
   ```

   Für zwei - x Begriffe:
   ```
   SELECT * FROM twitter.tweets_by_authors WHERE expr(tweets_index,
   '{filter: {type: "regexp", field: "content", value:
   "(.\*\n\*from.\*\n\*hello.\*\n\*)|(.\*\n\*hello.\*\n\*from.\*\n\*)"}, sort: {field:
   "number_of_likes", reverse: true}}') LIMIT 25;
   ```