Learn Twitter API Programming

Adam Green, 140Dev.com, 140dev@gmail.com, @140dev

Building Multiple Sites from a Tweet Database

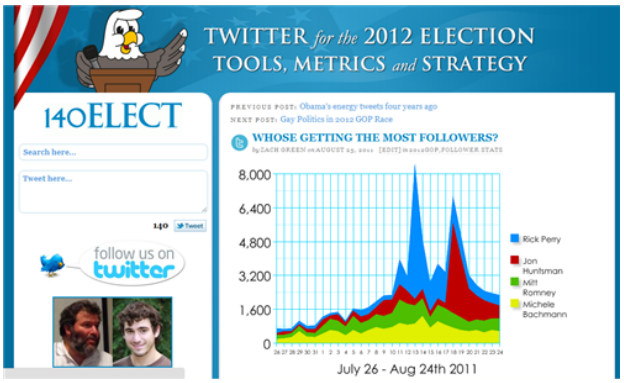
These three sites all use data from a common MySQL database that collects tweets by and about Barack Obama and the GOP presidential candidates. The page delivery and database code is PHP, and the UI is jQuery.

Tweets are collected with the Twitter streaming API, which delivers about 60,000 tweets a day. The database now contains 5.2 million tweets from 1.4 million different accounts.

2012Twit.com (@2012twit)

140Townhall.com (@140townhall)

140Elect.com (@140elect)



Twitter API Overview

There are several components to the Twitter API. All of them are free, but there are limits on the number of times they can be called in an hour, and the amount of data that can be requested.

You are allowed to to cache this data in a local database, and use it for display on a website or app, or in various forms of analysis. You are not allowed to build a secondary API that resells this data.

Some portions of the API require no authentication, some use basic authentication (account name and password), and some use OAuth authentication.

Most parts of the API use a REST model, except for the Streaming API, which requires a continuous connection.

Data is returned in both XML and JSON, but over time XML is being phased out.

Search API - REST access to up to 1,500 past tweets based on queries. Only last 5 - 7 days of tweets are available. No authentication required.

Streaming API - Continuous connection that receives tweets in real-time. Up to 400 keywords and 5,000 user accounts may be used for tweet collection. Basic and OAuth authentication.

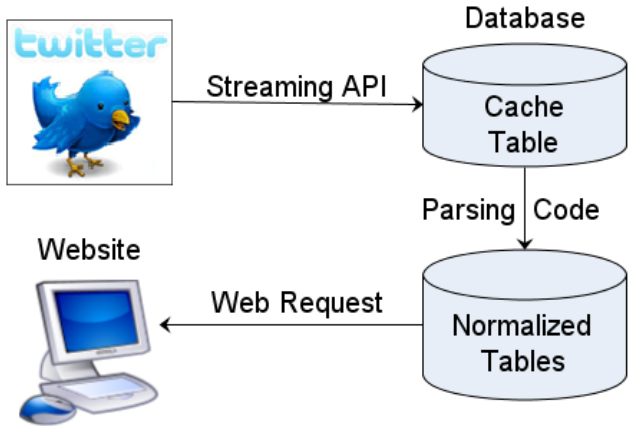
REST API - A wide collection of API calls that allow the creation of all Twitter features on a website or mobile app. OAuth authentication.

There are a variety of rate limits: 1,000 tweets per day per account, 250 Direct Messages per day per account, and 350 REST API requests per hour.

Twitter is well known to have reliability and scaling problems. Any app based on the API, must allow for frequent errors.

Tweet Aggregation Architecture

The foundation of a Twitter aggregation site is collecting tweets in a normalized database and then delivering them from the database to the website.



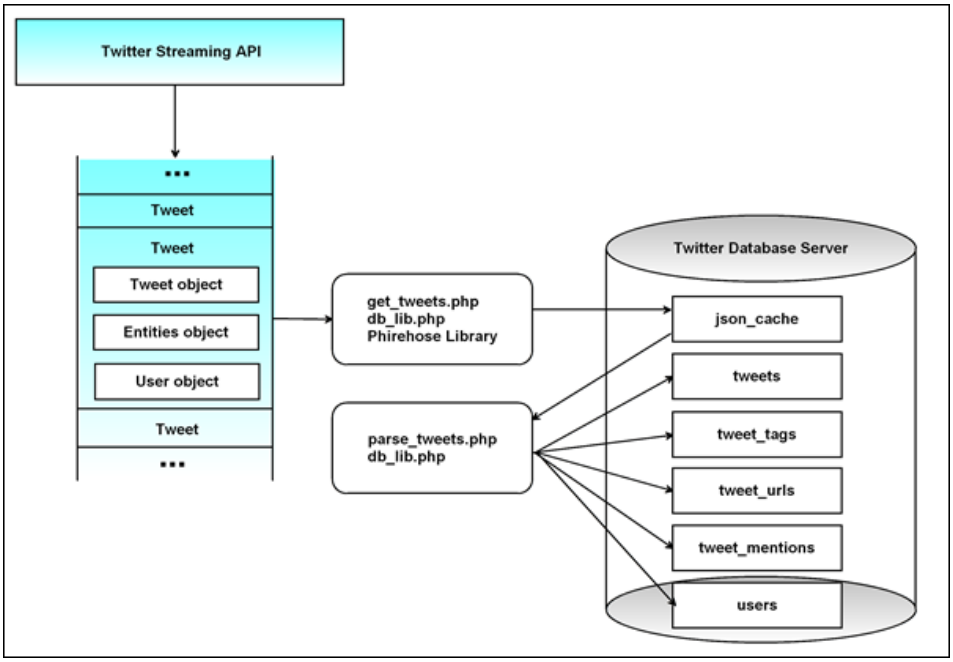
Benefits:

- Rate limit compliance - Any number of web pages requests can be served without using any valuable API calls.
- Performance optimization - The database server can be scaled to deliver a desired performance level without waiting for the API to respond.
- Reliability - Even if the Twitter API site fails completely, the website will continue delivering data from the database.
- Multiple site delivery - Once the database is in place, any number of sites and apps can share this data.
- Long-term archive and datamining - Over time the database becomes an ever more valuable asset. Datamining can be done to extract trends, leads, and search results going much further back in time than the Twitter search feature.

http://140dev.com/twitter-api-programming-tutorials/twitter-api-database-cache/

140Dev Twitter Framework

The following pages are based on an open source PHP framework available on the 140dev.com site: http://140dev.com/free-twitter-api-source-code-library/



http://140dev.com/free-twitter-api-source-code-library/twitter-database-server/code-architecture/

It uses the Phirehose PHP library to make a continuous connection to the streaming API. This requires a background process to be run on the server.

https://github.com/fennb/phirehose

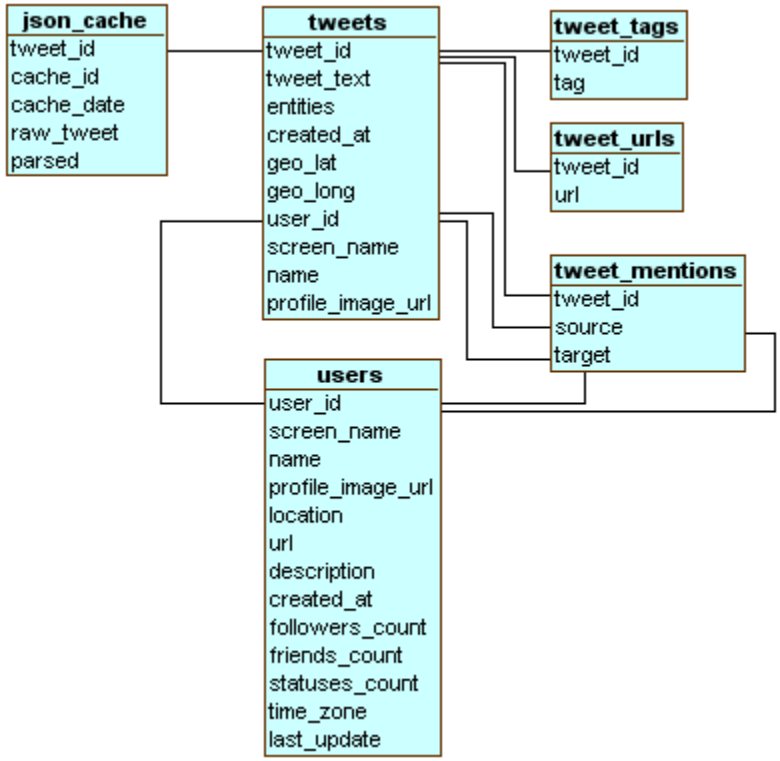
The 140 Dev Framework runs two background processes:

- get_tweets.php connects to the streaming API and stores an entire tweet package into a cache tabble.

- parse_tweets.php reads new tweets out of the cache table and distributes the data into a collection of normalized tables.

Twitter Database Schema

This is a minimal schema for a tweet database used by the 140 Dev Framework. For actual applications additional tables should be created to add meta data and speed queries.



Common tables added to tweet aggregation apps:

- Excluded words - common offensive words that are used to block tweets from being stored in the database.
- Excluded users - screen names of abusive or spam accounts.
- Hourly stats - summary of statistical data accumulated each hour to allow rapid calculations later.
- Daily stats
- API log - Log of all API calls with amount of data returned and possible errors.

http://140dev.com/free-twitter-api-source-code-library/twitter-database-server/mysql-database-schema/

Sample Twitter Queries

Identify key influencers

```
SELECT screen_name, COUNT( * ) AS cnt
FROM tweets
GROUP BY screen_name
ORDER BY cnt DESC
```

Identify leads with specific attributes

```
SELECT screen_name, name, description, url, followers_count
FROM users
WHERE description LIKE '%lawyer%'
ORDER BY followers_count DESC
```

Create frequency counts for keyword usage

```
SELECT DATE( created_at ) , COUNT( * )
FROM tweets
WHERE tweet_text LIKE '%jobs%'
GROUP BY DATE( created_at )
```

Collect data for a tag cloud

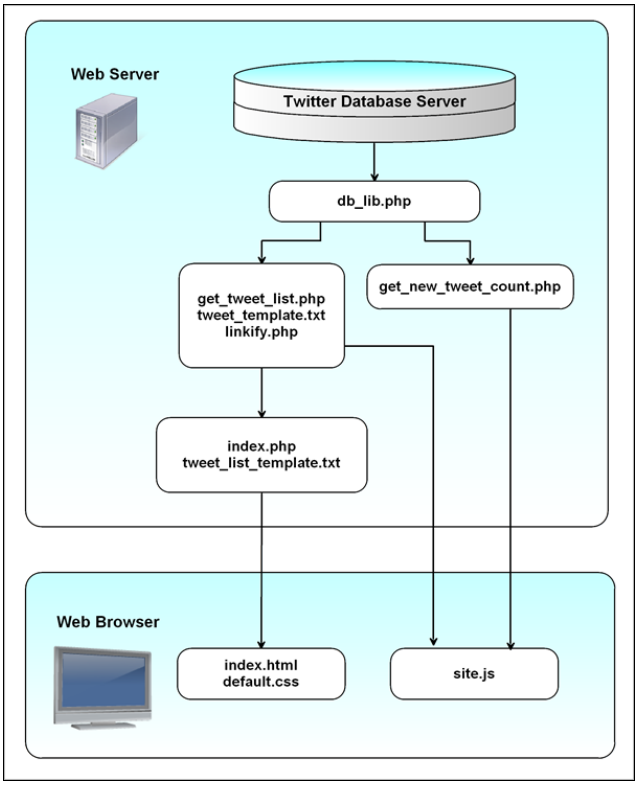
```
SELECT tag, COUNT( * ) AS cnt
FROM tweet_tags
GROUP BY tag
ORDER BY cnt DESC
LIMIT 50
```

SEO Architecture for Tweet Display

Google loves tweets and gives high ranking to text it recognizes as coming from tweets. This makes tweet aggregation on web pages a simple way to add Google juice to any site. By collecting tweets for specific keywords, and displaying them on your pages you can add steadily changing text rich in keywords.

A common mistake is to deliver tweets to a web page after the page loads with Javascript. This makes the tweets invisible to Google.

The better model is to use PHP code on the server to inject a first set of tweets into the page when it loads as HTML. Then after the page is loaded, you can use Javascript to load new tweets in real-time.



http://140dev.com/free-twitter-api-source-code-library/twitter-display/code-architecture/

Automated Account Management

The Twitter API allows a website to read and modify a Twitter account in many ways: Tweeting, Retweeting, Sending direct messages, Following and Unfollowing, etc.

All of these operations require OAuth tokens for the account being changed. There are 2 models for these types of apps:

- Single account app - When you create an app on http://dev.twitter.com, you are given a set of OAuth tokens that can be used to change the account that owns that app. These can be used without any additional authentication programming.

  http://140dev.com/twitter-api-programming-tutorials/hello-twitter-oauth-php/

- Multiple account app - If you want multiple users to log into your site and allow their accounts to be changed, you must use the OAuth login system to allow users to give you app permission. This can be done with a variety of PHP libraries.

  https://dev.twitter.com/docs/twitter-libraries#php

Managing a Twitter account is an extremely time consuming activity. The best way to develop an influential Twitter account is to build an automated system that:

- Tweets 10 to 15 times a day based on a database of pre-written tweets or an algorithm that generates stats or selects tweets for retweeting

- Selects key influencers from the database and follows them at a steady rate.

- Unfollows users who have been followed but failed to follow back.

- When automating following and unfollowing a log must be maintained to make sure that no account is followed more than once. Repeatedly following the same accounts will eventually lead to suspension off your app.

Twitter Resources

Twitter Developer Site

Discussion forums - https://dev.twitter.com/discussions

Documentation - https://dev.twitter.com/docs

Phirehose Streaming API Library

https://github.com/fennb/phirehose

Boston Twitter API Developers Meetup

http://www.meetup.com/twitter-api-developers/

Adam Green - Twitter API Consulting Services

- Tweet database collection servers
- Tweet aggregation websites
- Twitter client mobile apps
- Integration with existing sites or creation of new sites

140dev@gmail.com, 781-879-2960