Databases for text storage

Jonathan Ronen

New York University

jr4069@nyu.edu

December 1, 2014

Overview

Why Databases?

- Structured way to store your data
- Accessible, shareable
- Manage growing volumes of data
- You cannot keep all of your data in working memory...
- **indexing**

Basic issues with databases

- Inserting data
- Schema
- Querying
- Indexing

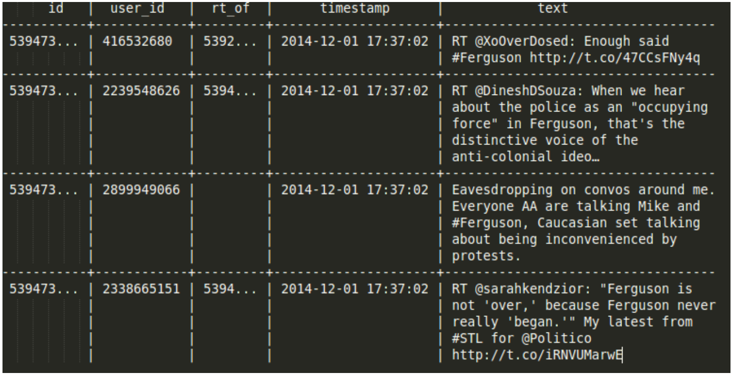
I'll show you how to do this in

- PostgresSQL
- MongoDB

PostgreSQL

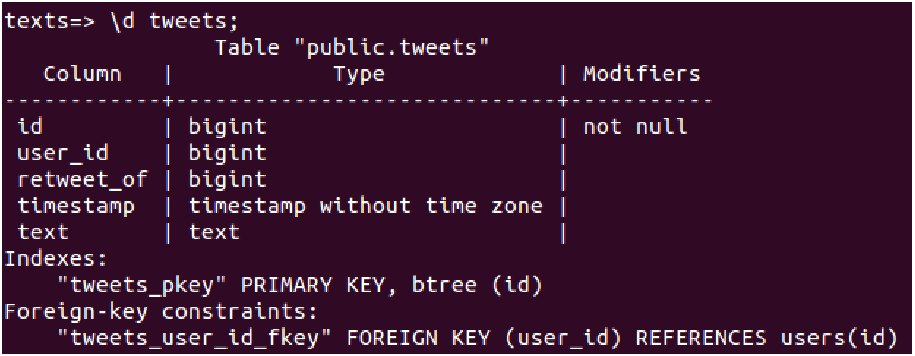
- Relational DB
- Which means we define tables with columns **and relations**
- Queried using Structured Query Language
- ES-QUE-ELL, or SEQUEL, but not SQUEAL
- opensource, free, very fast, advanced text search capabilities
- Friendly elephant logo

Basics of SQL



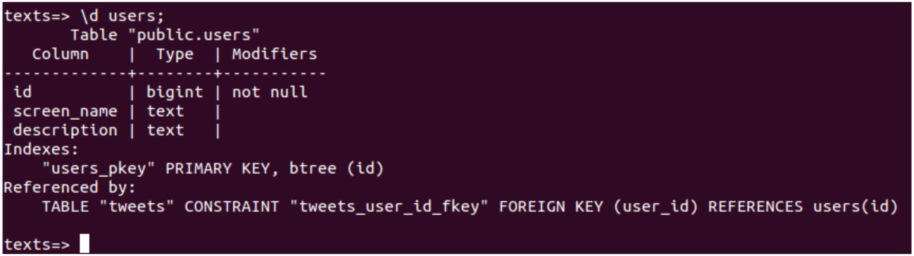
```
    id     |  user_id   | rt_of  |     timestamp       |                text
-----------+------------+--------+---------------------+-----------------------------------
539473... | 416532680  | 5392... | 2014-12-01 17:37:02 | RT @XoOverDosed: Enough said
          |            |         |                     | #Ferguson http://t.co/47CCsFNy4q
-----------+------------+--------+---------------------+-----------------------------------
539473... | 2239548626 | 5394... | 2014-12-01 17:37:02 | RT @DineshDSouza: When we hear
          |            |         |                     | about the police as an "occupying
          |            |         |                     | force" in Ferguson, that's the
          |            |         |                     | distinctive voice of the
          |            |         |                     | anti-colonial ideo…
-----------+------------+--------+---------------------+-----------------------------------
539473... | 2899949066 |         | 2014-12-01 17:37:02 | Eavesdropping on convos around me.
          |            |         |                     | Everyone AA are talking Mike and
          |            |         |                     | #Ferguson, Caucasian set talking
          |            |         |                     | about being inconvenienced by
          |            |         |                     | protests.
-----------+------------+--------+---------------------+-----------------------------------
539473... | 2338665151 | 5394... | 2014-12-01 17:37:02 | RT @sarahkendzior: "Ferguson is
          |            |         |                     | not 'over,' because Ferguson never
          |            |         |                     | really 'began.'" My latest from
          |            |         |                     | #STL for @Politico
          |            |         |                     | http://t.co/iRNVUMarwE
```

Basics of SQL



```
texts=> \d tweets;
                 Table "public.tweets"
   Column    |            Type             | Modifiers
-------------+-----------------------------+-----------
 id          | bigint                      | not null
 user_id     | bigint                      |
 retweet_of  | bigint                      |
 timestamp   | timestamp without time zone |
 text        | text                        |
Indexes:
    "tweets_pkey" PRIMARY KEY, btree (id)
Foreign-key constraints:
    "tweets_user_id_fkey" FOREIGN KEY (user_id) REFERENCES users(id)
```

Basics of SQL



```
texts=> \d users;
       Table "public.users"
   Column    |  Type  | Modifiers
-------------+--------+-----------
 id          | bigint | not null
 screen_name | text   |
 description | text   |
Indexes:
    "users_pkey" PRIMARY KEY, btree (id)
Referenced by:
    TABLE "tweets" CONSTRAINT "tweets_user_id_fkey" FOREIGN KEY (user_id) REFERENCES users(id)

texts=> 
```

Basics of SQL

SELECT statement

```
SELECT * FROM tweets WHERE user_id=2170941466;
```

SELECT statement with time range

```
SELECT * FROM tweets WHERE timestamp >'2014-12-2';
```

SELECT statement with LIKE

```
SELECT * FROM tweets WHERE lower(text) LIKE '%obama%';
```

Indexing

Imagine searching through a table:

id	user_id	timestamp	text
1	1	2014-11-30 10:23:40	I love the biebsssss!
2	2	2014-11-30 11:33:44	Bieberboy make me a baby!
3	1	2014-11-30 10:23:23	God if biebs dont come i shoot myself!
4	3	2014-11-30 9:12:11	I love bieber so much i have bieber san
5	2	2014-11-30 12:33:10	RT if you love biebsbs as much ias me!

Find me all tweets since noon.

Indexing

Imagine searching through a table:

id	user_id	timestamp	text
4	3	2014-11-30 9:12:11	I love bieber so much i have bieber san
3	1	2014-11-30 10:23:23	God if biebs dont come i shoot myself!
1	1	2014-11-30 10:23:40	I love the biebsssss!
2	2	2014-11-30 11:33:44	Bieberboy make me a baby!
5	2	2014-11-30 12:33:10	RT if you love biebsbs as much ias me!

Easy! Sort by time!

Indexing

An index is a sorted copy of a column.

timestamp	id
2014-11-30 9:12:11	4
2014-11-30 10:23:23	3
2014-11-30 10:23:40	1
2014-11-30 11:33:44	2
2014-11-30 12:33:10	5

(Or really, it's usually a btree...)

Text search in postgres

SELECT statement using PG text search

SELECT * FROM tweets WHERE to_tsvector('english', text) @@ to_tsquery('obama');

- to_tsvector
- to_tsquery
- (show these in the terminal...)

Text indexing

CREATE INDEX statement

```
CREATE INDEX text_idx ON tweets USING gin(to_tsvector('english', text));
```

SELECT statement using text index

```
SELECT * FROM tweets WHERE to_tsvector('english', text) @@ to_tsquery('obama');
```

Aggregation

GROUP BY statement

```
SELECT user_id, count(*) FROM tweets GROUP BY user_id;
```

MongoDB

- Document store
- noSQL doesn't mean query language isn't structured (but it's different..)
- opensource, free, really fast (sometimes)

JSON documents

```
{
    "created_at": "Wed Aug 13 15:20:46 +0000 2014",
    "lang": "en",
    "retweet_count": 0,
    "text": "Pennsylvania USA Philadelphia \u00bb Mike
    "user": {
        "name": "Jeff",
        "screen_name": "jeffersondol",
        "statuses_count": 207845,
        "description": "#android, #androidgames,#iphon
        "followers_count": 810,
        "lang": "en",
        "geo_enabled": false,
        "location": "Florida",
    }
}
```

MongoDB is a document database

- MongoDB lets you store these documents directly
- No need to flatten to tabular form!
- Comes with its own query syntax
- Also uses indexing to speed queries

SQL	**Mongo**
Database	Database
Table	Collection
Row	Document
Index	Index

MongoDB Query Syntax

Regex matching

```
db.collection.find({'text': /obama/})
```

Date range

```
db.collection.find({timestamp: {
        $gt: new Date(2014,10,6)
      }
   })
```

Text search in MongoDB

Creating a text index

```
db.tweets.ensureIndex({text: "text" })
```

Using text search

db.tweets.findOne({*text* : {search: "obama"}})

Aggregation in MongoDB

Aggregation framework

```
db.tweets.aggregate({$group: {
     _id:     "$user.screen_name",
     number: { $sum: 1 }
    }
  })
```

SMAPP

Some info on the smapp backend:

- MongoDB with index on tweet id, timestamp, random number (for sampling)
- No text index (yet!)
- New!: multiple collection for smappler indexes (smapptoolkit)

The End