CS 327E Lecture 12

Shirley Cohen

November 16, 2016

Plan for Today

- Reading Quiz
- APIs
- JSON
- MySQL + JSON

Readings for Today

- Chapter 13 from our Data Wrangling text
- JSON Data Interchange Format

Question 1

What is an API?

A. It’s a service for pulling data

B. It’s a service for pushing data

C. It’s a service for searching data

D. It’s a service that outputs responses in JSON format

E. All of the above

Question 2

A REST API client uses HTTP GET to pull data from the API service and HTTP POST to push data to the API service:

A. True

B. False

Question 3

Which of the following statements is **false**:

A. JSON is a collection of name/value pairs

B. JSON is a self-describing data format

C. JSON is used mostly with JavaScript

D. JSON is used for data exchange

E. JSON can represent nested data

Question 4

Which of the following is a **valid** JSON construct:

A. Array

B. Loop

C. Table

D. Function

E. Parameter

Question 5

How many nested objects are in this JSON document?


```
{
  "text": "RT @hiangieee: The students of @UTAustin took it to the streets today.
    #utprotest #Morningafter https://t.co/JvTLajTqT8",
  "retweet_count": 6236,
  "id_str": "796541058192736256",
  "favorited": false,
  "user": {
    "profile_background_image_url_https": "https://pbs.twimg.com
      /profile_background_images/569242807726915584/WDTmWx2F.jpeg",
    "entities": {
      "description": {
        "urls": []
      }
    },
    "followers_count": 73,
    "utc_offset": -21600
  },
  "metadata": {
    "iso_language_code": "en",
    "result_type": "recent"
  }
}
```


A. 1 B. 2 C. 3 D. 4 E. > 5

JSON Format

```
{
    "firstName": "Jason",
    "middleInitial": "M",
    "lastName": "Jones",
    "address":
    {
        "street": "444 Guadalupe",
        "city": "Autin",
        "state": "TX",
        "zip": 78705
    },
    "email":
    [
      "jason@utexas.com",
      "jjones@gmail.com"
    ],
    "phone":
    [
      {"work": "512-555-1212"},
      {"cell": "512-222-1234"}
    ]
}
```

object
- **{}**
- **{** *members* **}**

members
- *pair*
- *pair* **,** *members*

pair
- *string* **:** *value*

array
- **[]**
- **[** *elements* **]**

elements
- *value*
- *value* **,** *elements*

value
- *string*
- *number*
- *object*
- *array*
- **true**
- **false**
- **null**

Source: http://www.json.org

Concept Question 1

Suppose we wanted to add a second person to this document. How can we modify the JSON structure to represent *n* people?

```json
{
    "firstName": "Jason",
    "middleInitial": "M",
    "lastName": "Jones",
    "address":
    {
        "street": "444 Guadalupe",
        "city": "Autin",
        "state": "TX",
        "zip": 78705
    },
    "email":
    [
      "jason@utexas.com",
      "jjones@gmail.com"
    ],
    "phone":
    [
      {"work": "512-555-1212"},
      {"cell": "512-222-1234"}
    ]
}
```

A. Add a Person array, each person = an element in the array and each element = a nested object

B. Make each person = a nested object. The individual person objects are separated by a “,”

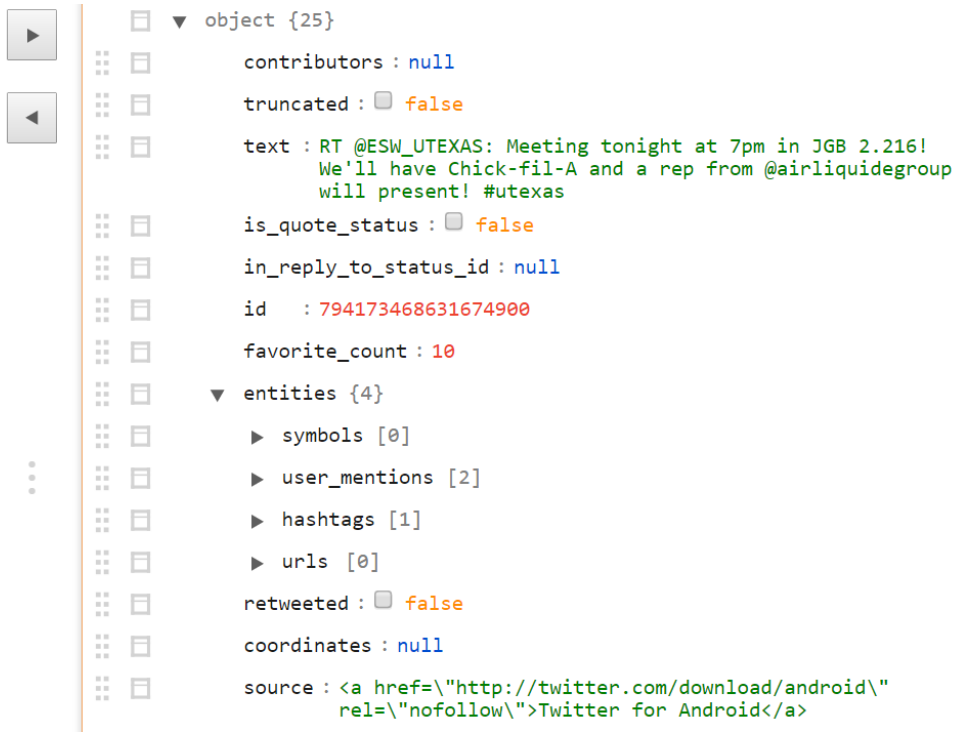
C. B + add a label for each person object. The label = a unique identifier such as an ssn.

D. All of the above

E. None of the above

Anatomy of a (partial) Tweet


```
{"contributors": null,
  "truncated": false,
   "text": "RT @ESW_UTEXAS: Meeting tonight at 7pm in JGB 2.216!
     We'll have Chick-fil-A and a rep from @airliquidegroup will
     present! #utexas", "is_quote_status": false,
      "in_reply_to_status_id": null,
      "id": 794173468631674881,
      "favorite_count": 10,
      "entities":
      {"symbols": [],
      "user_mentions":
        [{"id": 769031310182641665,
         "indices": [3, 14],
         "id_str": "769031310182641665",
         "screen_name": "ESW_UTEXAS",
         "name": "ESW at UT"},
        {"id": 104481993,
        "indices": [91, 107],
        "id_str": "104481993",
        "screen_name": "airliquidegroup",
        "name": "Air Liquide Group"}],
        "hashtags": [{"indices": [122, 129],
        "text": "utexas"}],
        "urls": []},
        "retweeted": false,
        "coordinates": null,
        "source": "<a href=\"http://twitter.com/download/android\"
```

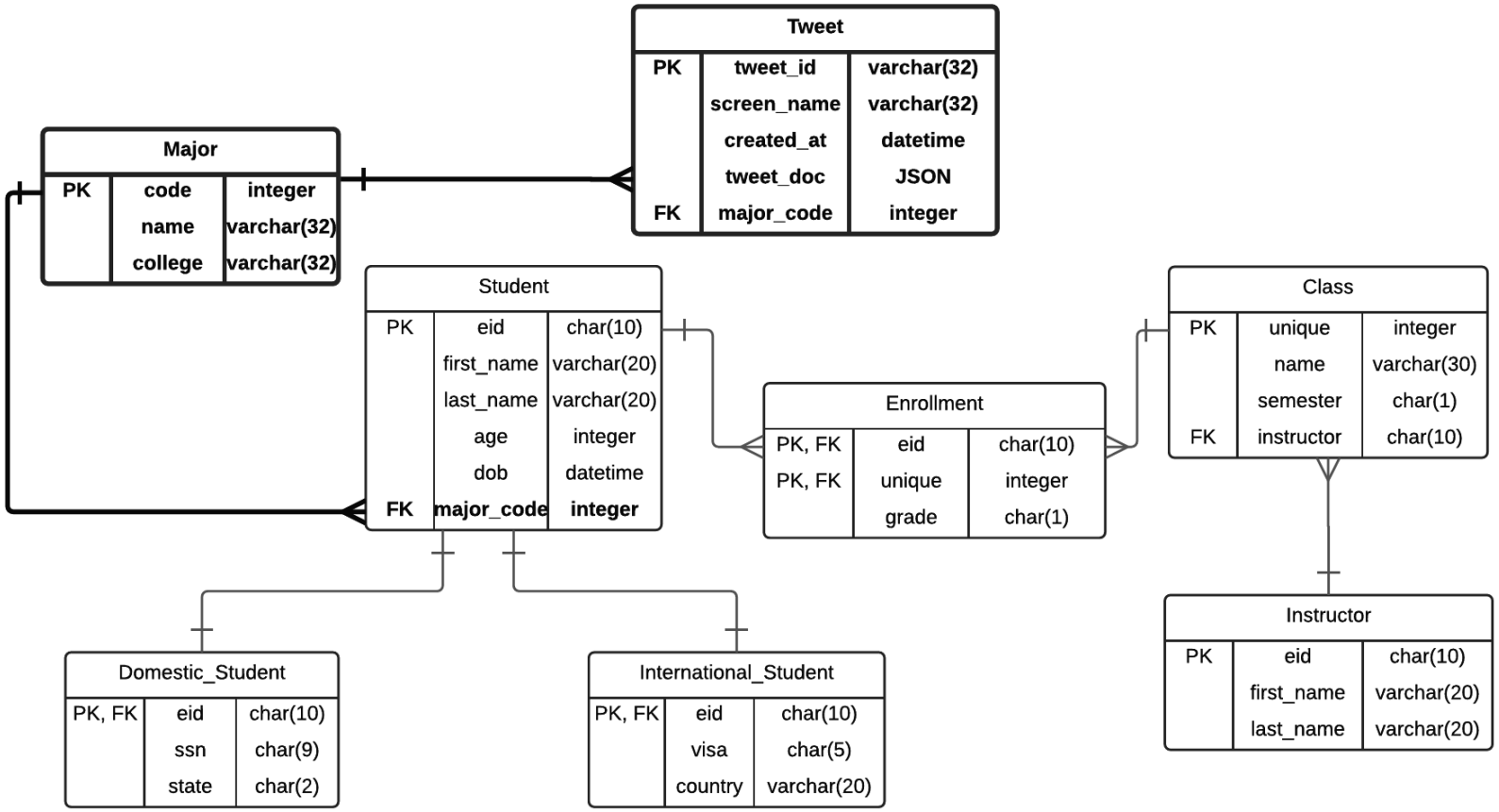




```
▼ object {25}
    contributors : null
    truncated : false
    text : RT @ESW_UTEXAS: Meeting tonight at 7pm in JGB 2.216!
           We'll have Chick-fil-A and a rep from @airliquidegroup
           will present! #utexas
    is_quote_status : false
    in_reply_to_status_id : null
    id : 794173468631674900
    favorite_count : 10
  ▼ entities {4}
    ▶ symbols [0]
    ▶ user_mentions [2]
    ▶ hashtags [1]
    ▶ urls [0]
    retweeted : false
    coordinates : null
    source : <a href=\"http://twitter.com/download/android\"
             rel=\"nofollow\">Twitter for Android</a>
```


Twitter API Field Guide: https://dev.twitter.com/overview/api/tweets

JSON Editor Online: https://chrome.google.com/webstore to download the Chrome plug-in.

UT Class Enrollment & Twitter



Logical ERD - UT Class Enrollment - CS 327E Fall 2016

New DDL

```
3  drop table if exists Major;
4  create table Major (
5      code int auto_increment primary key,
6      name varchar(32) not null,
7      college varchar(32) not null
8  );
```

```
26 alter table Student add column major_code int;
27 alter table Student add constraint fk_major_code
28     foreign key (major_code) references Major(code);
```

```
42 drop table if exists Tweet;
43 create table Tweet (
44     tweet_id varchar(32) generated always
45         as (json_unquote(json_extract(tweet_doc, '$.id_str'))) stored primary key,
46     screen_name varchar(32) generated always
47         as (json_unquote(json_extract(tweet_doc, '$.user.screen_name'))) stored,
48     created_at datetime generated always
49         as (str_to_date(json_unquote(json_extract(tweet_doc, '$.created_at')),
50         '%a %b %d %H:%i:%s +0000 %Y')) stored,
51     tweet_doc json,
52     major_code int,
53     foreign key (major_code) references Major(code)
54 );
```

Concept Question 2

Suppose we want to store the `favorite_count` from a tweet it in its own field because we plan to query it frequently. How can we formulate the JSON path expression to extract the `favorite_count` value from a tweet?

```
create table Tweet (
    tweet_id varchar(32) generated always
        as (json_unquote(json_extract(tweet_doc, '$.id_str'))) stored primary key,
    screen_name varchar(32) generated always
        as (json_unquote(json_extract(tweet_doc, '$.user.screen_name'))) stored,
    created_at datetime generated always
        as (str_to_date(json_unquote(json_extract(tweet_doc, '$.created_at')),
        '%a %b %d %H:%i:%s +0000 %Y')) stored,
    tweet_doc json,
    major_code int,
    foreign key (major_code) references Major(code)
);
```

```
{"contributors": null,
  "truncated": false,
   "text": "RT @ESW_UTEXAS: Meeting tonight at 7pm in JGB 2.216!
     We'll have Chick-fil-A and a rep from @airliquidegroup will
     present! #utexas", "is_quote_status": false,
      "in_reply_to_status_id": null,
      "id": 794173468631674881,
      "favorite_count": 10,
      "entities":
```

Path expressions:

A. `$.contributors.favorite_count`

B. `$.favorite_count`

C. `$.text.favorite_count`

D. `$.id.favorite_count`

Twitter API

```
1  import tweepy
2
3  API_KEY = 'm6BbBvpoik7CqNgz6mPai7Mta'
4  API_SECRET = 'ZzZpRxsn4EXpnKyIP1rSQ4xUBKONrXeQ4VQ9yn3uxdoTlRxQkq'
5  TOKEN_KEY = '795474433448218624-11jeAlxRQKX5qG1VTABHzF33PWtReD7'
6  TOKEN_SECRET = 'fMVQFRPL2PYceAKKq8DbQgnlwiNnazEZ9ceqwae3WGxEI'
7
8  auth = tweepy.OAuthHandler(API_KEY, API_SECRET)
9  auth.set_access_token(TOKEN_KEY, TOKEN_SECRET)
10 api_inst = tweepy.API(auth)
```

`tweepy` = Twitter API wrapper, makes OAuth a lot simpler

Visit https://apps.twitter.com to obtain API key and secret along with TOKEN key and secret for your application. Note: API key is also called Consumer key.

Concept Question 3

What does this code sample do?

```
10  def store_tweet(item):
11      db = dataset.connect('sqlite:///data_wrangling.db')
12      table = db['tweets']
13      item_json = item._json.copy()
14      for k, v in item_json.items():
15          if isinstance(v, dict):
16              item_json[k] = str(v)
17      table.insert(item_json)
18
19
20  auth = tweepy.OAuthHandler(API_KEY, API_SECRET)
21  auth.set_access_token(TOKEN_KEY, TOKEN_SECRET)
22
23  api = tweepy.API(auth)
24
25  query = '#childlabor'
26  cursor = tweepy.Cursor(api.search, q=query, lang="en")
27
28  for page in cursor.pages():
29      for item in page:
30          store_tweet(item)
```

A. Searches the database for tweets related to “child labor”.

B. Searches Twitter for tweets containing #childlabor, pulls those tweets, and stores each one into its own table in SQLite.

C. Answer B except that the tweets are all stored in the same `tweets` table.

D. Searches Twitter for tweets containing #childlabor and generates a web page for each tweet.

E. None of the above.

Source: https://github.com/jackiekazil/data-wrangling/blob/master/code/chp13-apis/advanced_data_pull.py

Twitter Client

```
16  def do_data_pull(api_inst):
17
18      sql_query = "select code, name from Major order by name"
19
20      try:
21          conn = create_connection()
22          db_cursor = conn.cursor()
23          query_status = run_stmt(db_cursor, sql_query)
24          resultset = db_cursor.fetchall()
25
26          for record in resultset:
27              major_code = record[0]
28              major_name = record[1]
29
30              utexas_query = "(#UTexas OR @UTAustin OR url:utexas.edu) AND "
31              twitter_query = utexas_query + "'" + major_name + "'"
32              print "twitter_query: " + twitter_query
33              twitter_cursor = tweepy.Cursor(api_inst.search, q=twitter_query, lang="en")
34
35              for page in twitter_cursor.pages():
36                  for item in page:
37                      json_str = json.dumps(item._json)
38                      print "found a " + major_name + " tweet"
39                      insert_stmt = "insert into Tweet(tweet_doc, major_code) values(%s, %s)"
40                      run_prepared_stmt(db_cursor, insert_stmt, (json_str, major_code))
41                      do_commit(conn)
42
43      except pymysql.Error as error:
44          is_success = False
45          print "do_data_pull: " + e.strerror
```

Plan for Next Week

- Last Quiz on Monday: Readings will come straight from the MySQL Reference Guide. See class web page for details.
- Final Project: Assignment and rubric will be out on Monday.
- No class on Wednesday: Thanksgiving break!