

# CS 327E Lecture 13

## Shirley Cohen

November 21, 2016

# Plan for Today

- Reading Quiz
- MySQL + JSON
- Final Project Assignment

# Readings for Today

- JSON Data Type from the MySQL Reference Manual (section 12.6)
- JSON Functions from the MySQL Reference Manual (section 13.16)

# Question 1

Which of the following statements is **false** about MySQL support for JSON:

- A. MySQL has a native JSON datatype
- B. MySQL converts JSON documents to a binary format
- C. MySQL indexes JSON documents directly
- D. MySQL supplies a number of functions for operating on JSON data
- E. MySQL automatically validates JSON documents that are stored in JSON columns

## Question 2

In the context of JSON, normalization means the process of removing duplicate keys (or names) from a document.

- A. True
- B. False

# Question 3

```
1  {
2      "text": "RT @ESW_UTEXAS: Meeting tonight at 7pm in JGB 2.216! We'll
3          have Chick-fil-A and a rep from @airliquidegroup will present!
4          #utexas",
5      "retweeted_status":
6      {
7          "entities":
8          {
9              "user_mentions":
10             [
11                 {
12                     "id": 104481993,
13                     "indices":
14                         [
15                             75,
16                             91
17                         ]
18                 }
19             }
20 }
```

What does the path expression below evaluate to?

`$.retweeted_status.entities.user_mentions[0].indices[0]`

- A. {}
- B. 75
- C. NULL
- D. 91
- E. "id" : 104481993

# Question 4

```
1  {
2      "text": "RT @ESW_UTEXAS: Meeting tonight at 7pm in JGB 2.216! We'll
3          have Chick-fil-A and a rep from @airliquidegroup will present!
4          #utexas",
5      "retweeted_status":
6      {
7          "entities":
8          {
9              "user_mentions":
10             [
11                 {
12                     "id": 104481993,
13                     "indices":
14                         [
15                             75,
16                             91
17                         ]
18                 }
19             }
20 }
```

Let  $j$  be to the JSON document shown. What does the select statement below evaluate to?

```
select JSON_EXTRACT(j,
    '$.retweeted_status.entities.
    user_mentions[0].id');
```

- A. "indices" : [75, 91]
- B. 75
- C. 91
- D. 104481993
- E. None of the above

# Question 5

```
1  {
2      "text": "RT @hiangieeee: The students of @UTAustin took it to the streets today.  
        #utprotest #Morningafter https://t.co/JvTLajTqT8",  
3      "retweet_count": 6236,  
4      "id_str": "796541058192736256",  
5      "favorited": false,  
6      "user": {  
7          "profile_background_image_url_https": "https://pbs.twimg.com  
        /profile_background_images/569242807726915584/WDTmWx2F.jpeg",  
8          "entities": {  
9              "description": {  
10                  "urls": []  
11              }  
12          },  
13          "followers_count": 73,  
14          "utc_offset": -21600  
15      },  
16      "metadata": {  
17          "iso_language_code": "en",  
18          "result_type": "recent"  
19      }  
20 }
```

Let  $j$  be the JSON document shown. What does the select statement below evaluate to?

```
select JSON_SEARCH(j, 'one', 'RT');
```

- A. "\$.text"
- B. "\$.\*"
- C. {}
- D. NULL
- E. None of the above

# Demo 1

JSON Editor Online

New Open ▾ Save ▾ Settings ▾ Help

The screenshot shows the JSON Editor Online interface. On the left, a code editor displays a JSON document with line numbers 51 to 59. The document contains a single object with various properties like 'favorited', 'retweeted\_status', 'text', etc. On the right, a tree view shows the structure of the JSON object, with expanded sections for 'retweeted\_status' and 'text'. Below the tree view, the evaluated result for the expression '\$.retweeted\_status.id' is shown as the value '794164880169238500'. A 'Fork me on GitHub' button is visible at the bottom right.

```
51 "favorited": false,
52 "retweeted_status": {
53     "contributors": null,
54     "truncated": false,
55     "text": "Meeting tonight at 7pm in JGB 2.216!
      We'll have Chick-fil-A and a rep from
      @airliquidegroup will present! #utexas",
56     "is_quote_status": false,
57     "in_reply_to_status_id": null,
58     "id": 794164880169238500,
59     "favorite_count": 1,
```

```
retweet_count : 1
id_str : 794164880169238500
favorited : false
retweeted_status {24}
contributors : null
truncated : false
text : Meeting tonight at 7pm in JGB 2.216!
```

JSONPath Online Evaluator - jsonpath.com

## Inputs

### JSONPath Syntax

```
$.retweeted_status.id
```

Example '\$.phoneNumbers[\*].type' See also [JSONPath expressions](#)

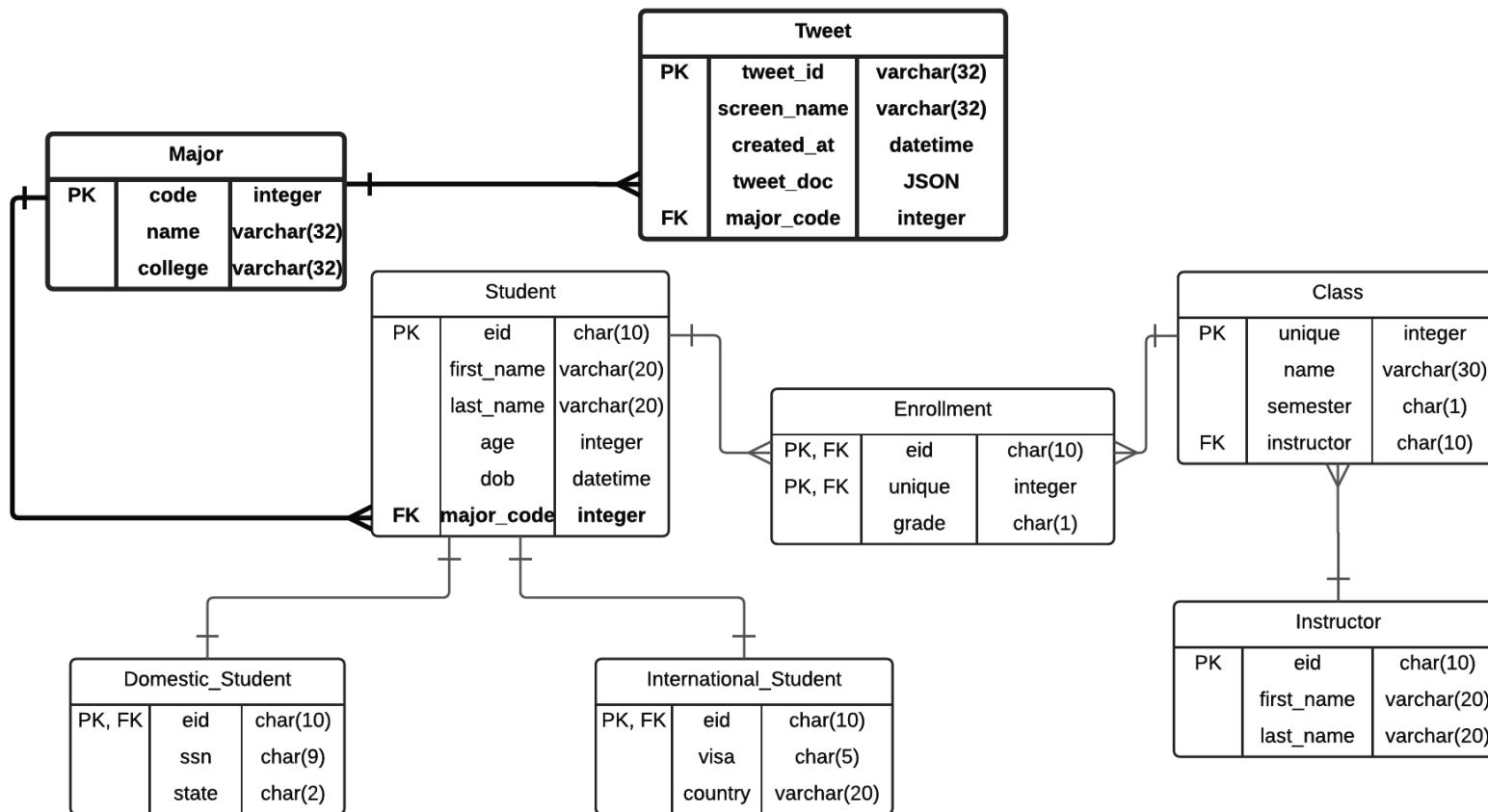
### JSON

```
1 [
2     "contributors": null,
3     "truncated": false,
4     "text": "RT @ESW_UTEXAS: Meeting tonight at 7pm in JGB 2.216! We'll have Chick
5     "is_quote_status": false,
6     "in_reply_to_status_id": null,
7     "id": 794173468631674881,
8     "favorite_count": 0,
9     "entities": {
10         "symbols": [],
11         "user_mentions": [
12             {
13                 "id": 769031310182641700,
14                 "indices": [
15                     3,
16                     14
```

## Evaluation Results

```
1 '0' => "794164880169238500"
2
```

# 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 );
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
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);
29
30
31
32
33
34
35
36
37
38
39
40
41
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')), '%a %b %d %H:%i:%s +0000 %Y')) stored,
50   tweet_doc json,
51   major_code int,
52   foreign key (major_code) references Major(code)
53 );
54 );
```

# 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
```

# Demo 2

The screenshot shows a desktop environment with a command prompt window, a code editor window, and a database management interface.

**Command Prompt:**

```
C:\utcswork\cs327e_fall_2016\snippets>python twitter_client.py
twitter_query: (#UTexas OR @UTAustin OR url:utexas.edu) AND 'Biology'
found a Biology tweet
found a Biology tweet
found a Biology tweet
twitter_query: (#UTexas OR @UTAustin OR url:utexas.edu) AND 'Biomedical Engineering'
twitter_query: (#UTexas OR @UTAustin OR url:utexas.edu) AND 'Chemistry'
twitter_query: (#UTexas OR @UTAustin OR url:utexas.edu) AND 'Computer Science'
found a Computer Science tweet
found a Computer Science tweet
found a Computer Science tweet
twitter_query: (#UTexas OR @UTAustin OR url:utexas.edu) AND 'Economics'
found a Economics tweet
twitter_query: (#UTexas OR @UTAustin OR url:utexas.edu) AND 'Electrical Engineering'
```

**Code Editor:**

```
2 ✘
3 ✘ select m.name, m.code, count(t.tweet_id) as tweet_count
4 ✘ from Major m left outer join Tweet t
5 ✘ on m.code = t.major_code
6 ✘ group by m.name, m.code
7 ✘ order by tweet_count desc;
```

**Database Management Interface:**

The interface displays a tree view of database schemas and tables on the left, and a result grid on the right.

**Schemas:**

- scratch
- sxsw
- sys
- teams
- test
- test\_results
- utexas

**Tables:**

- utexas
- Tables
  - class
  - domestic\_student
  - enrollment
  - instructor
  - international\_student
  - major
    - Columns
      - code
      - name
      - college
    - Indexes
    - Foreign Keys
    - Triggers
  - student
  - tweet
    - Columns
      - tweet\_id
      - screen\_name
      - created\_at
      - tweet\_doc
      - major\_code
  - Indexes

**Result Grid:**

	name	code	tweet count
▶	Plan II	14	31
	Journalism	3	27
	Math	8	10
	Film	4	6
	Psychology	16	6
	Physics	7	5
	Electrical Engineering	11	3
	Computer Science	9	3
	Biology	5	3
	Economics	13	1

# Concept Question 1

We want to extend the Twitter Client to check for duplicate tweets before doing the insert into MySQL. Assume that in Python we extract the id of the tweet and store the value in the variable \$id. How can we formulate a SQL query that checks for duplicate tweets given \$id?

```
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')), '%a %b %d %H:%i:%s +0000 %Y')) stored,
50    tweet_doc json,
51    major_code int,
52    foreign key (major_code) references Major(code)
53  );
54
```

- A. select count(\*) from Tweet where tweet\_id = \$id
- B. select \* from Tweet order by tweet\_id
- C. select count(distinct tweet\_id) from Tweet where tweet\_id = \$id
- D. select \* from Tweet where tweet\_id = \$id
- E. select count(tweet\_id) from Tweet where tweet\_id = \$id

# Concept Question 2

We want to implement a more accurate count of tweets per UT major. More specifically, we want to filter out all retweets and only add up the origin tweets. Assume that for all tweets, we extract the origin tweet id from the tweet and we store this value in a new field called `Tweet.origin_tweet_id`. How can we modify the query below to only count unique tweets?

```
select m.name, m.code, count(t(tweet_id) as tweet_count  
from Major m left outer join Tweet t  
on m.code = t.major_code  
group by m.name, m.code  
order by tweet_count desc;
```

- A. Replace: `count(t(tweet_id)` with: `count(t.origin_tweet_id)`
- B. Replace: `count(t(tweet_id)` with: `count(distinct t.origin_tweet_id)`
- C. Change the outer join to an inner join
- D. Change the left outer join to a right outer join
- E. None of the above

# Final Project

[http://www.cs.utexas.edu/~scohen/project/final\\_project.pdf](http://www.cs.utexas.edu/~scohen/project/final_project.pdf)