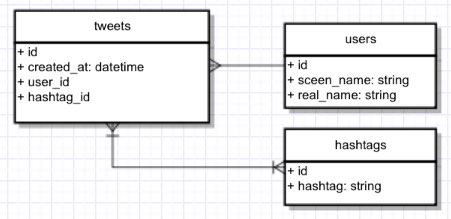
**Task #1: UML ERD**

This is the general UML ERD for how I would model the information extracted:

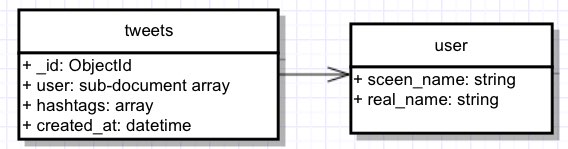


The idea is that there is three main categories of information: the tweet, the user, and the hashtag. The ‘id’, ‘user\_id’, and ‘hashtag\_id’ implementation can be omitted depending on how the database is structured.

**Task #2A: NoSQL**

(i) Implementation Model

Below is the collection structure for modeling information extracted from the raw tweets:



There will only be one collection of “tweets”. No foreign key is used as “user” displayed above is not a separate collection. “User” is actually embedded in sub-document array within the “tweets” collection. Similarly, “hashtags” is also embedded within the “tweets” collection as a standard array.

The above collection structure is based on using MongoDB with the example JSON structure below:

//tweets collection

[

{

"\_id": ObjectId(xxxxx),

"user": {

screen\_name: "evlist",

real\_name: "Eric van der Vlist"

},

"hashtags": ["xmlprague", "marklogic",...],

"created\_at": ISODate()

},

…

]

(ii) Process for Storing Information

1. Create a python script that can load the 2 JSON files in-memory to form dictionary format so that python can easily manipulate
2. For each JSON file, the python script extracts only the needed information (tweet created time, user’s screen name, user’s real name, and hashtags) and format the data in the JSON structure described above. Special attention should be paid on formatting the date
3. In the same python script using "pymongo” package to connect to localhost and to load the dictionary data into MongoDB.
4. Run “mongod” command to initiate mongo server and run the python script that contains step 1-3. The end result is that all the tweets will be stored in MongoDB under a single collection called “tweets”

(iii) Pseudo Code / Procedures for Answering Questions

Q1: Who tweeted the most during the conference?

* Install node module “mongodb”
* Create nodejs app that connects to the database at localhost
* Run the following query inside the nodejs app:

Inside app.js, run “aggregate” function for MongoDB with the following conditions:

1. Match this condition:

(

(“created\_at” >= day 1’s start time of 9am CET Feb 14)

and (“created\_at” <= day 1’s end time of 9am CET Feb 14)

)

or

(

(“created\_at” >= day 2’s start time of 9am CET Feb 15)

and (“created\_at” <= day 2’s end time of 9am CET Feb 15)

)

)

2. Group by screen name of each tweet and count no. of tweet per screen name

3. Sort the result by descending order

4. Output only the first element in the array (since the output is an array of js

objects of the screen name and number of tweets arranged in descending

order, the first element of the array will give us the person’s screen name

who has the highest number of tweets).

Q2: What were the top 10 hash tags used?

Inside app.js, run “find” function for MongoDB with the following conditions:

1. Match this condition:

(

(“created\_at” >= day 1’s start time of 9am CET Feb 14)

and (“created\_at” <= day 1’s end time of 9am CET Feb 14)

)

or

(

(“created\_at” >= day 2’s start time of 9am CET Feb 15)

and (“created\_at” <= day 2’s end time of 9am CET Feb 15)

)

)

2. Only get the “hashtags” result

3. At this point, the “hashtags” result will be in a messy form where it’s

wrapped an array of objects which each contains an array of hashtags.

A function needs to be applied to count each unique value of hashtags. This

function needs to do the following:

i. Extract object sequentially

ii. Within object, get the array of hashtags

iii. Inside each array of hashtags, count these hashtags in a new object

with these rules:

a. If the hashtag key already exists, add 1 to the count

b. Else, create a new hashtag key and make the value to be 1

iv. Convert this object into an array of objects such that each object

contains only the hashtag key and the count

v. Sort this array in descending order by the count value and return the

resulting array

4. After getting the array, just loop through the array to print only the first 10

elements in the array (corresponding to the 10 most used hashtags)

Q3: For a particular hour, how many tweets were produced?

Inside app.js, set the following variables:

* Current time t0 (set to be start time of day 1)
* t1 (one hour after t0).
* day (set to be 1 to signify Day 1 or Feb 14)
* endDay (set to be end time for Day 1)

Create an immediate function (call it loop() ).

Inside this immediate function, run “aggregate” function for MongoDB with the

following conditions:

1. Match this condition:

(“created\_at” >= t0) and (“created\_at” < t1)

2. Group by the time t0 and count number of tweets in that hour

3. Output the number of tweets for that hour

4. Determine next step based on “t0” and “day” variables

If t0 < endDay: 🡨 check if time reached end of day

t0 = t1

t1 = t1 + 1 hour

run the loop() as callback function (note that we cannot just do a

for loop but need to rely on callbacks because of the asynchronous nature of javascript)

Else if day == 1 🡨 check if we are still on day 1

t0 = start time of day 2

t1 = t0 + 1 hour

endDay = end time of day 2

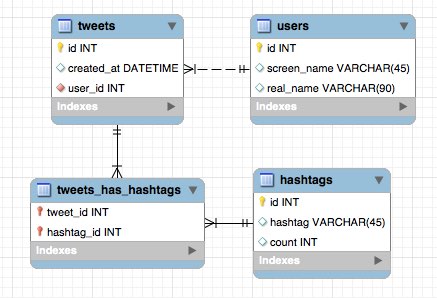
day += 1

run the loop() as callback function

**Task #2B: SQL**

(i) Implementation Model

Below shows the full schema of organizing the data:



(ii) Process for Storing Information

1. Use MySQL workbench to create the tables of the implementation model above with all variables and their value types properly defined. (I prefer to use MySQL if I were to do this exercise).
2. Create python script with SQLAlchemy python library that can perform the following actions:
   1. Load the 2 JSON files in-memory
   2. Loop through each object and get only extract needed info
   3. Within the loop, perform insertion into the database with either standard SQL command or use ORM from SQLAlchemy:

* Insert user info with the following details:
  + First check whether user already exists (can have a list variable in python in-memory to keep track of each unique user’s screen\_name)
  + If the user’s screen\_name does not exist:
    - Add that screen\_name to the list
    - SQL command to insert new user with the screen\_name and real\_name into the users table
    - SQL command to read the last insert id for the new user entry and store in-memory
    - SQL command to insert new tweet with created\_at and user\_id info. The user\_id is the last insert id for the new user
  + Else:
    - SQL command to read the user id with the screen\_name match in users table, and store in-memory
    - SQL command to insert new tweet with created\_at and user\_id info. The user\_id is the user id that matches the screen\_name in the users table
* SQL command to read the last insert id for the new tweet entry and store in-memory
* Insert hashtag info with the following details:
  + First check whether hashtag already exists (can have a list variable in python in-memory to keep track of each unique hashtag)
  + If the hashtag does not exist:
    - Add that hashtag to the list
    - SQL command to insert new hashtag with the hashtag and count (being 1) into the hashtags table
    - SQL command to insert new relationship of the last inserted tweet id and the last inserted hashtag id in the tweets\_has\_hashtags table
  + Else :
    - SQL command to update count of the specific hashtag row in the MySQL that matches this new hashtag in the hashtags table (return the hashtag id that has just been updated)
    - SQL command to insert new relationship of the last inserted tweet id and the last updated hashtag id in the tweets\_has\_hashtags table
  1. Repeat (b) and (c) until going through every tweet object in the 2 JSON files

1. Use MAMP if using a Mac (or WAMP if using a PC) program to start a local

server for running the MySQL database

1. Run the python script while MAMP is running the server.

(iii) Pseudo Code / Procedures for Answering Questions

Q1: Who tweeted the most during the conference?

Type the following command in MySQL Workbench for the query:

SELECT screen\_name, COUNT(tweets.id) AS num\_tweets

FROM users

LEFT JOIN tweets ON users.id = tweets.user\_id

GROUP BY users.id

ORDER BY num\_tweets DESC

LIMIT 1;

Q2: What were the top 10 hash tags used?

Type the following command in MySQL Workbench for the query:

SELECT hashtag, count

FROM hashtags

ORDER BY count DESC

LIMIT 10;

Q3: For a particular hour, how many tweets were produced for the query?

Type the following command in MySQL Workbench:

SELECT DATE(created\_at) AS date, HOUR(created\_at) AS hour, COUNT(tweets.id) AS num\_tweets

FROM tweets

WHERE

(DAY(created\_at) = 14 AND HOUR(created\_at) >= 8)

AND (DAY(created\_at) = 14 AND HOUR(created\_at) <= 15)

OR

(DAY(created\_at) = 15 AND HOUR(created\_at) >= 8)

AND (DAY(created\_at) = 15 AND HOUR(created\_at) <= 15)

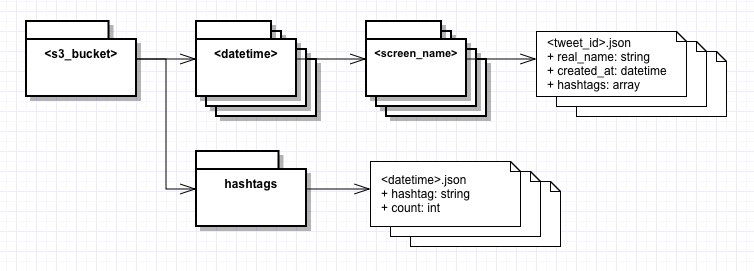
GROUP BY DATE(created\_at), HOUR(created\_at);

Note that I restraint the hours of each day from 8am to 3pm (rather than 9am to 4pm). That is because we are using UTC and not converting time into CET.

**Task #2B: SQL**

(i) Implementation Model

The model is shown below:

****

The model is represented by a series of folders on S3. Within the <s3\_bucket> folder, there are multiple <datetime> subfolders. Each <datetime> subfolder represents an hour (eg. 2015-02-14-0800). Within each <datetime> subfolder, there are <screen\_name> subfolders, one for each unique screen\_name within that hour. Within each <screen\_nme> subfolder, there are a number of json files that is named by the tweet\_id (eg. 566739270405087232.json). The tweet\_id is a tweet identifier created by Twitter.

There is another branch of subfolder called “hashtags”. This subfolder contains a list of files with <datetime>.json as its filename, which are listed in chronological order. Each of these files contain the unique hashtag name and the count of each hashtag. The hashtags will be ordered by count value in descending order. Each time the hashtag count is updated, a new <datetime>.json file is created. This way for the query, the user only needs to look at the latest <datetime>.json file for the count of each hashtag.

Based on the above file structure, one can locate a particular tweet file by accessing the following file directory:

s3://<s3\_bucket>/<datetime>/<screen\_name>/<tweet\_id>.json

One can locate a particular hashtag count file by accessing the following file directory:

s3://<s3\_bucket>/hashtags/<datetime>.json

(ii) Process for Storing Information

1. Setup s3 bucket on AWS S3
2. Run python script that does the following:
   1. Load the 2 JSON files in-memory
   2. Loop through each tweet and get only extract needed info and upload data to S3 by:
      1. Create a dictionary with the following info: real name, created\_at, and the hashtags. Use json.dump() function to output the json file
      2. Use boto to check if folder on S3 <datetime> folder for the corresponding created\_at for the tweet already exists:

* If <datetime> folder already exists, check if <screen\_name> folder for the corresponding screen\_name for the tweet already exists
  + If <screen\_name> folder already exists, put that json output into that folder in S3 using boto
  + Else, create a new folder with the corresponding <screen\_name> and upload the json output file into that folder in S3 sing boto
* Else, create a new folder with the corresponding <datetime>
  + If <screen\_name> folder already exists, put that json output into that folder in S3 using boto
  + Else, create a new folder with the corresponding <screen\_name> and upload the json output file into that folder in S3 sing boto
    1. Update hashtags in <hashtags> folder by:
* Use boto to check the latest json file in the <hashtags> folder
  + If no json file in that folder, create a temporary dictionary variable in-memory (call this “hashtags\_temp”)
  + Else if there is at least one json file, store the key-value of latest json file in a dictionary variable in-memory (call this “hashtags\_temp”)
* Loop through the hashtags within the array of the tweet to update the “hashtags\_temp” in the following way:
  + If the hashtag already exists in “hashtags\_temp”, add one to the count for that hashtag
  + Else, create a new object with the hashtag word and set the count to be 1.
* This results in the latest count of hashtags in the “hashtags\_temp” variable. Perform a merge-sort such that the hashtags is listed in descending order of the count value.
* Use the combination of json.dump() and boto to output and upload this new hashtag-counter json file back into the <hashtags> folder in S3. The filename should be set to the creation time of this new output. That way, we know which is the latest updated hashtags file in <hashtags>

(iii) Pseudo Code / Procedures for Answering Questions

Q1: Who tweeted the most during the conference?

Create a python script that does the following:

* Have an empty dictionary variable called “screen\_name\_temp” to store the screen\_name count
* Loop through each <datetime> folder:
  + Loop through each <screen\_name> folder:
    - Count the number of files in that folder and update “screen\_name\_temp” in the following way:
    - If the screen\_name key already exists in “screen\_name\_temp”, add the “number of files in the screen\_name folder” to the value for that key
    - Else, create a new key using that screen\_name word and set the value to be the “number of files in the screen\_name folder”.
* Now that “screen\_name\_temp” has the count of screen\_name and its corresponding count of tweets, you simply need to loop through “screen\_name\_temp” and find the screen\_name key with the maximum value.
  + For this loop you can use two temporary variables. One variable for storing the tempory screen\_name value, one for storing the temporary count value. As you go through each screen\_name in “screen\_name\_temp”, if the count value is bigger than the one stored in the temporary count value, update the temporary screen\_name variable and temporary count variable to the new values. That way in the end, you will leave with the screen\_name variable with the highest count value.

Q2: What were the top 10 hash tags used?

Create a python script that does the following:

* Use boto to go to <hashtags> folder in S3 and retrieve the lastest json file (the file has the count of each hashtag) and store the json file into dictionary variable in-memory
* Since the hashtags are already sorted in descending order, simply loop through the dictionary variable to find the first 10 hashtag keys.

Q3: For a particular hour, how many tweets were produced?

Since <datetime> folders are already chronologically arranged, this simplifies things a lot. Create a python script that does the following:

* Only loop through all the <datetime> folders that is within the 9am CET to 4pm CET each day:
  + Loop through each <screen\_name> folder
    - Count the number of files.
  + Sum up all the counts for each <screen\_name> folder within that <datetime> folder. That sum corresponds to the total number of tweets for that hour.
  + Print the sum of tweets for that hour, then move to the next <datetime> folder

**Task #3**

Instructions:

* Setup mongo by doing the following:
* Run “sudo mongod”
* Run “mongo” on separate terminal to open mongo console
* Inside mongo console, type:
  + use w205assignment3
  + db.createCollection('tweets')
* Run the command “python load.py” to load the 2 JSON files into MongoDB (you can type “db.tweets.find()” in mongo console to double-check if data actually got loaded into the MongoDB)
* Run “app.js” to conduct the 3 queries for the 3 questions asked. Below is the output that answer the 3 questions after running app.js:

Answer to Question 1: xmlprague tweeted the most with 68 tweets

Answer to Question 2: These are the top 10 most used hashtags among all the tweets:

{ hashtag: 'xmlprague', count: 746 }

{ hashtag: 'XMLPrague', count: 38 }

{ hashtag: 'thetransformationsong', count: 27 }

{ hashtag: 'oxygenxml', count: 19 }

{ hashtag: 'XProc', count: 18 }

{ hashtag: 'RDFa', count: 15 }

{ hashtag: 'XML', count: 14 }

{ hashtag: 'BRILLIANT', count: 14 }

{ hashtag: 'FUCKYEAH', count: 14 }

{ hashtag: 'xproc', count: 12 }

Answer to Question 3: Hourly tweets in chronological order:

On Feb 14 2015 between 09:00 and 10:00 (CET), there are 46 tweets

On Feb 14 2015 between 10:00 and 11:00 (CET), there are 55 tweets

On Feb 14 2015 between 11:00 and 12:00 (CET), there are 19 tweets

On Feb 14 2015 between 12:00 and 13:00 (CET), there are 42 tweets

On Feb 14 2015 between 13:00 and 14:00 (CET), there are 9 tweets

On Feb 14 2015 between 14:00 and 15:00 (CET), there are 24 tweets

On Feb 14 2015 between 15:00 and 16:00 (CET), there are 24 tweets

On Feb 15 2015 between 09:00 and 10:00 (CET), there are 21 tweets

On Feb 15 2015 between 10:00 and 11:00 (CET), there are 56 tweets

On Feb 15 2015 between 11:00 and 12:00 (CET), there are 16 tweets

On Feb 15 2015 between 12:00 and 13:00 (CET), there are 66 tweets

On Feb 15 2015 between 13:00 and 14:00 (CET), there are 13 tweets

On Feb 15 2015 between 14:00 and 15:00 (CET), there are 13 tweets

On Feb 15 2015 between 15:00 and 16:00 (CET), there are 41 tweets