INDIANA UNIVERSITY BLOOMINGTON

B669/I590: Management, Access, and Use of Big and Complex Data

Project Report [System Track]

Bipra De

Masters of Science in Computer Science[First year]

23rd November,2014.

TABLE OF CONTENTS

1. Problem Statement

1) Design two different data models. See http://docs.MongoDB.org/manual/data-modeling/ for data modeling reference.

2) Ingest all Twitter dataset (profile, networks, and tweets) into both data models of Mongo DB, using the data models you defined in step 1.

3) Perform the following query/aggregation/update operations against the data for both data models.

a) Return all the user IDs from the tweets, which contain keyword KEYWORD in their text fields. Set the KEYWORD to a high-frequency word (e.g., “good”) first, then set it to a low-frequency word (e.g., “qwertyuiopasdfghjkl”) and run the query again. That is, running two queries for each data model.

b) Return cumulated retweet counts of all tweets, each of which has at least one hashtag.

c) Select a user/users who has/have the largest number of followers, find all the followers in the network dataset, and return all the names of the followers (if these names can be retrieved from the profile dataset).

d) Add a follower to a user, update all the necessary collections.

4) Performance Evaluation:

a) What’s being measured?

- You will measure response time of each operation in 2) and 3), for both data models that you designed.

b) How to measure?

- Each operation response time can be measured at the Mongo DB client side. Write your Mongo DB client code that implements all the operations in 2) and 3). Wrap each of the operations with start and finish timestamps.

- To grab timestamp, we recommend that you embed the timestamp related code in the same process/thread of your Mongo DB client code. Although you can measure timestamps by invoking Linux commands (/bin/date) before and after invoking your Mongo DB client code, the response time will be less accurate for faster operations. However, the /bin/date command is still acceptable in this project.

2. System Specifications

2.1 Hardwares Used:

1. MacBook Pro:
    a. Processor: 2.5 GHz Intel Core i5
    b. Memory: 8 GB 1600 MHz DDR3

2. External Hard Drive to host Mongo DB data.

2.2 Softwares Used:

1. Mongo DB version 2.6.5
2. Eclipse IDE with MonjaDB plugin

2.3 Data Set:

This dataset is a subset of Twitter. It contains 284 million following relationships, 3 million user profiles and 50 million tweets. The dataset was collected at May 2011

The dataset was created for the following research work:

- Rui Li, Shengjie Wang, Kevin Chen-Chuan Chang: Multiple Location Profiling for Users and Relationships from Social Network and Content PVLDB 5(11): 1603-1614, 2012
- Rui Li, Shengjie Wang, Hongbo Deng, Rui Wang, Kevin Chen-Chuan Chang: Towards social user profiling: unified and discriminative influence model for inferring home locations. KDD 2012:1023-1031

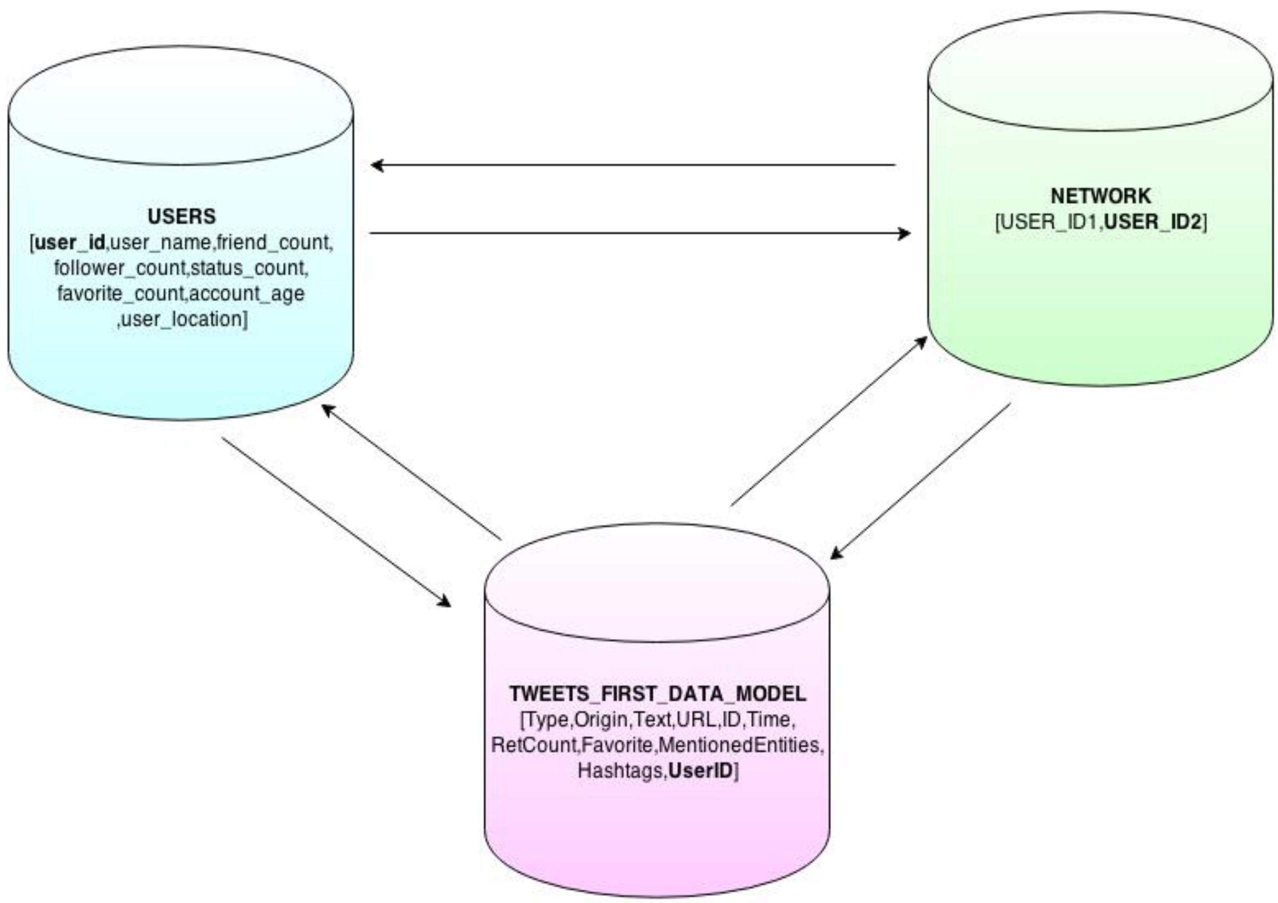
3. Data Model 1 (Reference data model):

3.1 Data Model Design

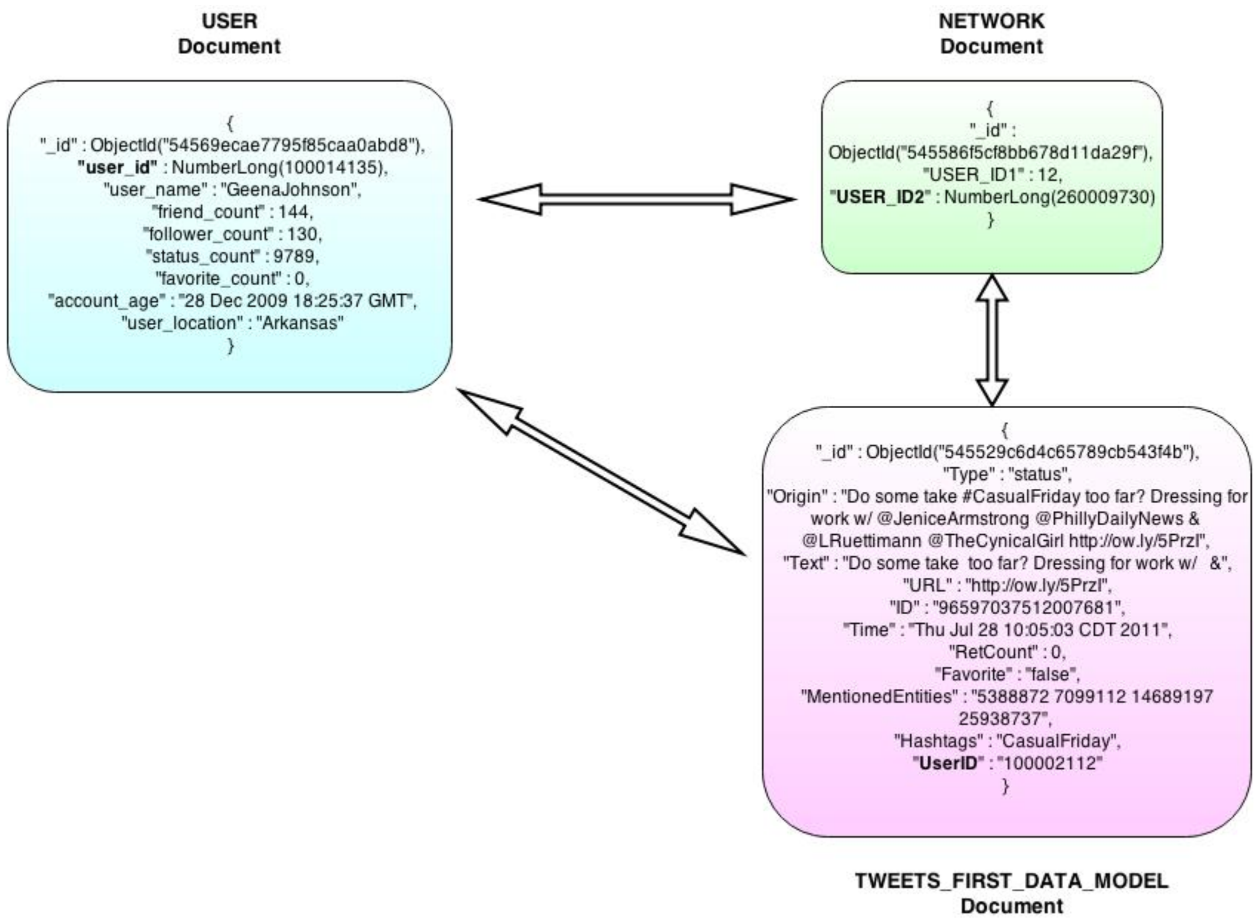
- In Reference data model, users' profile data, their network data and their tweet data are stored in three separate collections – USERS, NETWORK and TWEETS_FIRST_DATA_MODEL respectively.

- User ID is a common field across all the three collections and is used as a reference to link one document of a collection to a document in another collection.

3.2 Collection level representation diagram

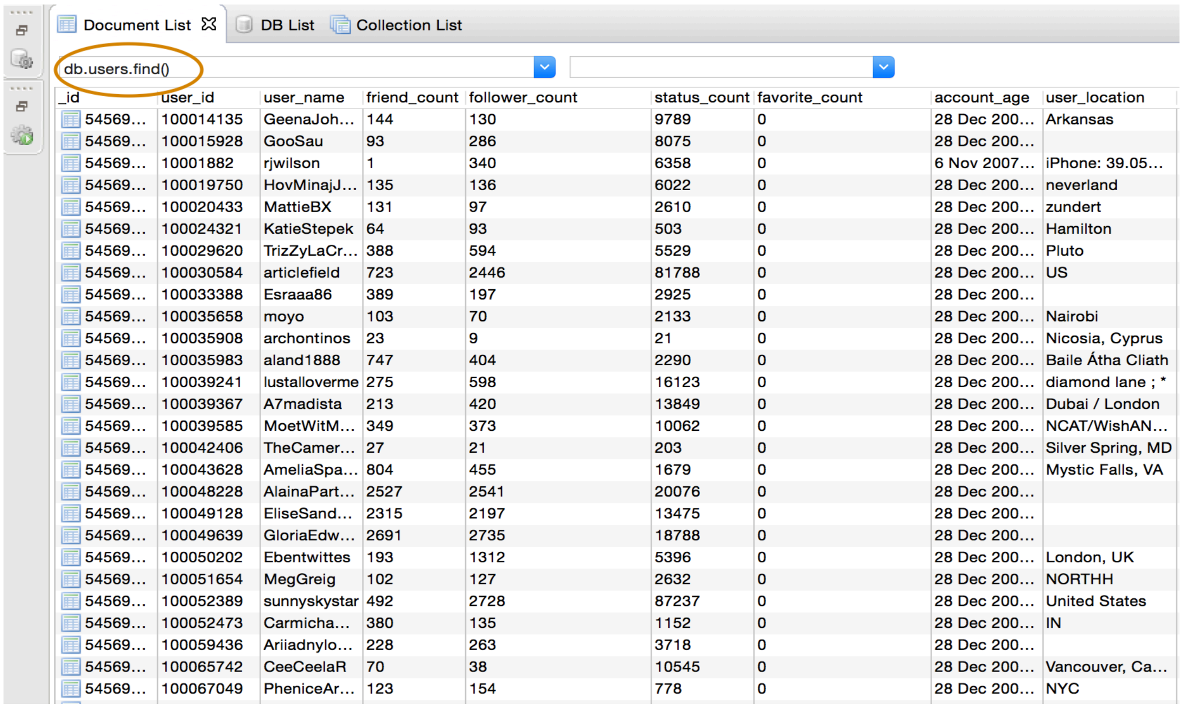


3.3 Document level representation diagram



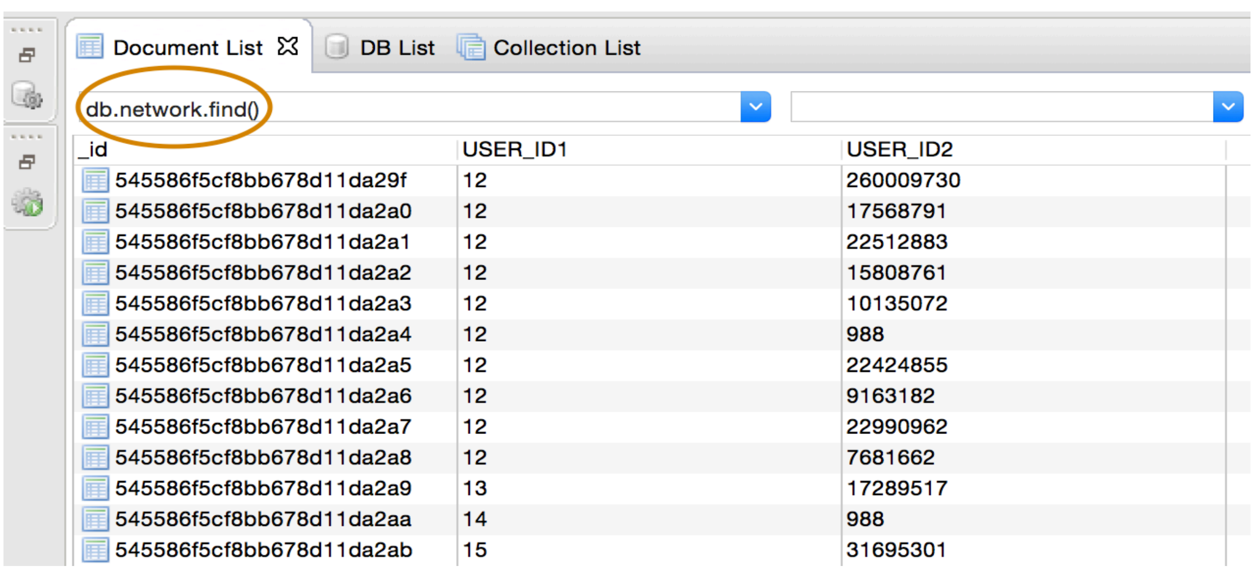
3.4 Tabular view representation

1.USER Collection:



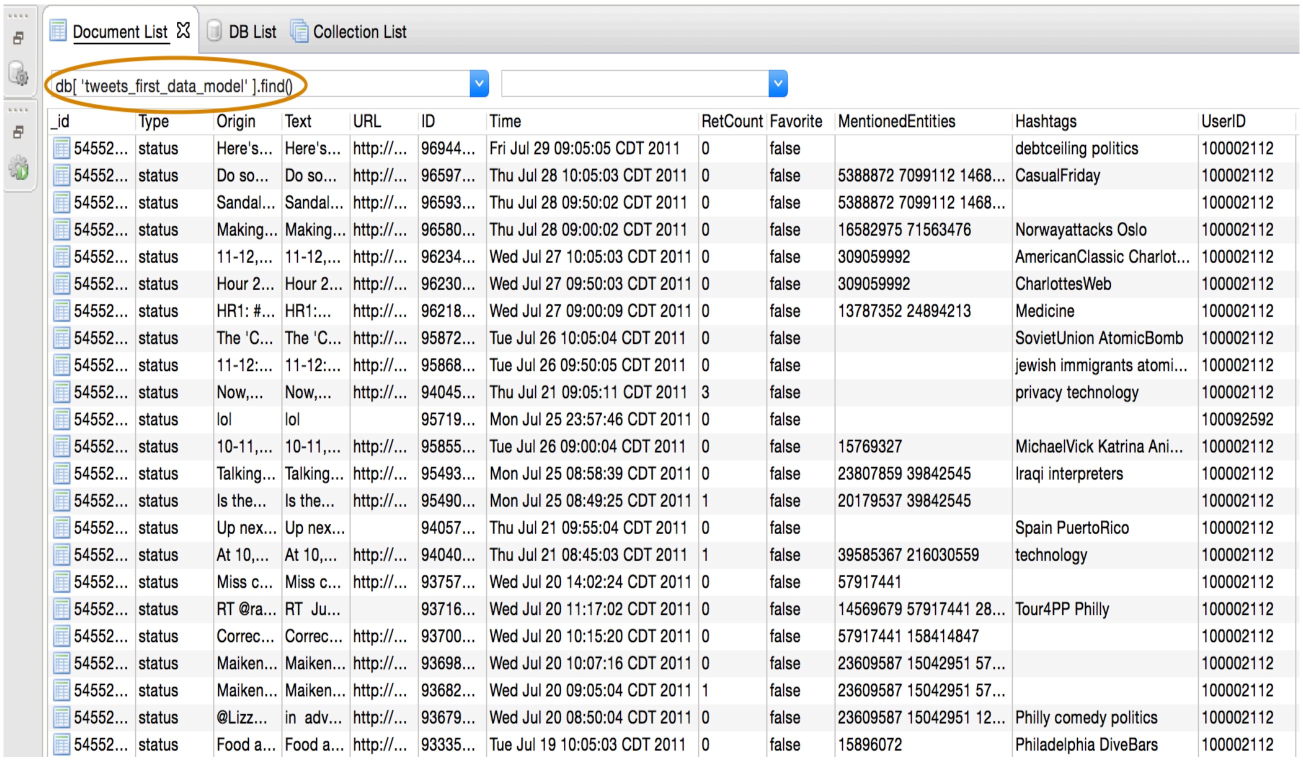
_id	user_id	user_name	friend_count	follower_count	status_count	favorite_count	account_age	user_location
54569…	100014135	GeenaJoh…	144	130	9789	0	28 Dec 200…	Arkansas
54569…	100015928	GooSau	93	286	8075	0	28 Dec 200…	
54569…	10001882	rjwilson	1	340	6358	0	6 Nov 2007…	iPhone: 39.05…
54569…	100019750	HovMinajJ…	135	136	6022	0	28 Dec 200…	neverland
54569…	100020433	MattieBX	131	97	2610	0	28 Dec 200…	zundert
54569…	100024321	KatieStepek	64	93	503	0	28 Dec 200…	Hamilton
54569…	100029620	TrizZyLaCr…	388	594	5529	0	28 Dec 200…	Pluto
54569…	100030584	articlefield	723	2446	81788	0	28 Dec 200…	US
54569…	100033388	Esraaa86	389	197	2925	0	28 Dec 200…	
54569…	100035658	moyo	103	70	2133	0	28 Dec 200…	Nairobi
54569…	100035908	archontinos	23	9	21	0	28 Dec 200…	Nicosia, Cyprus
54569…	100035983	aland1888	747	404	2290	0	28 Dec 200…	Baile Átha Cliath
54569…	100039241	lustalloverme	275	598	16123	0	28 Dec 200…	diamond lane ; *
54569…	100039367	A7madista	213	420	13849	0	28 Dec 200…	Dubai / London
54569…	100039585	MoetWitM…	349	373	10062	0	28 Dec 200…	NCAT/WishAN…
54569…	100042406	TheCamer…	27	21	203	0	28 Dec 200…	Silver Spring, MD
54569…	100043628	AmeliaSpa…	804	455	1679	0	28 Dec 200…	Mystic Falls, VA
54569…	100048228	AlainaPart…	2527	2541	20076	0	28 Dec 200…	
54569…	100049128	EliseSand…	2315	2197	13475	0	28 Dec 200…	
54569…	100049639	GloriaEdw…	2691	2735	18788	0	28 Dec 200…	
54569…	100050202	Ebentwittes	193	1312	5396	0	28 Dec 200…	London, UK
54569…	100051654	MegGreig	102	127	2632	0	28 Dec 200…	NORTHH
54569…	100052389	sunnyskystar	492	2728	87237	0	28 Dec 200…	United States
54569…	100052473	Carmicha…	380	135	1152	0	28 Dec 200…	IN
54569…	100059436	Ariiadnylo…	228	263	3718	0	28 Dec 200…	
54569…	100065742	CeeCeelaR	70	38	10545	0	28 Dec 200…	Vancouver, Ca…
54569…	100067049	PheniceAr…	123	154	778	0	28 Dec 200…	NYC

2.NETWORK Collection:



_id	USER_ID1	USER_ID2
545586f5cf8bb678d11da29f	12	260009730
545586f5cf8bb678d11da2a0	12	17568791
545586f5cf8bb678d11da2a1	12	22512883
545586f5cf8bb678d11da2a2	12	15808761
545586f5cf8bb678d11da2a3	12	10135072
545586f5cf8bb678d11da2a4	12	988
545586f5cf8bb678d11da2a5	12	22424855
545586f5cf8bb678d11da2a6	12	9163182
545586f5cf8bb678d11da2a7	12	22990962
545586f5cf8bb678d11da2a8	12	7681662
545586f5cf8bb678d11da2a9	13	17289517
545586f5cf8bb678d11da2aa	14	988
545586f5cf8bb678d11da2ab	15	31695301

3.TWEETS_FIRST_DATA_MODEL Collection:



Document List | DB List | Collection List

db['tweets_first_data_model'].find()

_id	Type	Origin	Text	URL	ID	Time	RetCount	Favorite	MentionedEntities	Hashtags	UserID
54552...	status	Here's...	Here's...	http://...	96944...	Fri Jul 29 09:05:05 CDT 2011	0	false		debtceiling politics	100002112
54552...	status	Do so...	Do so...	http://...	96597...	Thu Jul 28 10:05:03 CDT 2011	0	false	5388872 7099112 1468...	CasualFriday	100002112
54552...	status	Sandal...	Sandal...	http://...	96593...	Thu Jul 28 09:50:02 CDT 2011	0	false	5388872 7099112 1468...		100002112
54552...	status	Making...	Making...	http://...	96580...	Thu Jul 28 09:00:02 CDT 2011	0	false	16582975 71563476	Norwayattacks Oslo	100002112
54552...	status	11-12,...	11-12,...	http://...	96234...	Wed Jul 27 10:05:03 CDT 2011	0	false	309059992	AmericanClassic Charlot...	100002112
54552...	status	Hour 2...	Hour 2...	http://...	96230...	Wed Jul 27 09:50:03 CDT 2011	0	false	309059992	CharlottesWeb	100002112
54552...	status	HR1: #...	HR1:...	http://...	96218...	Wed Jul 27 09:00:09 CDT 2011	0	false	13787352 24894213	Medicine	100002112
54552...	status	The 'C...	The 'C...	http://...	95872...	Tue Jul 26 10:05:04 CDT 2011	0	false		SovietUnion AtomicBomb	100002112
54552...	status	11-12:...	11-12:...	http://...	95868...	Tue Jul 26 09:50:05 CDT 2011	0	false		jewish immigrants atomi...	100002112
54552...	status	Now,...	Now,...	http://...	94045...	Thu Jul 21 09:05:11 CDT 2011	3	false		privacy technology	100002112
54552...	status	lol	lol		95719...	Mon Jul 25 23:57:46 CDT 2011	0	false			100092592
54552...	status	10-11,...	10-11,...	http://...	95855...	Tue Jul 26 09:00:04 CDT 2011	0	false	15769327	MichaelVick Katrina Ani...	100002112
54552...	status	Talking...	Talking...	http://...	95493...	Mon Jul 25 08:58:39 CDT 2011	0	false	23807859 39842545	Iraqi interpreters	100002112
54552...	status	Is the...	Is the...	http://...	95490...	Mon Jul 25 08:49:25 CDT 2011	1	false	20179537 39842545		100002112
54552...	status	Up nex...	Up nex...		94057...	Thu Jul 21 09:55:04 CDT 2011	0	false		Spain PuertoRico	100002112
54552...	status	At 10,...	At 10,...	http://...	94040...	Thu Jul 21 08:45:03 CDT 2011	1	false	39585367 216030559	technology	100002112
54552...	status	Miss c...	Miss c...	http://...	93757...	Wed Jul 20 14:02:24 CDT 2011	0	false	57917441		100002112
54552...	status	RT @ra...	RT Ju...		93716...	Wed Jul 20 11:17:02 CDT 2011	0	false	14569679 57917441 28...	Tour4PP Philly	100002112
54552...	status	Correc...	Correc...	http://...	93700...	Wed Jul 20 10:15:20 CDT 2011	0	false	57917441 158414847		100002112
54552...	status	Maiken...	Maiken...	http://...	93698...	Wed Jul 20 10:07:16 CDT 2011	0	false	23609587 15042951 57...		100002112
54552...	status	Maiken...	Maiken...	http://...	93682...	Wed Jul 20 09:05:04 CDT 2011	1	false	23609587 15042951 57...		100002112
54552...	status	@Lizz...	in adv...	http://...	93679...	Wed Jul 20 08:50:04 CDT 2011	0	false	23609587 15042951 12...	Philly comedy politics	100002112
54552...	status	Food a...	Food a...	http://...	93335...	Tue Jul 19 10:05:03 CDT 2011	0	false	15896072	Philadelphia DiveBars	100002112

3.5 Data Loading

- **USERS Collection**:
  - Step 1: The users.txt (tsv) raw data file was reformatted from ISO-8859-1 to UTF-8 format using the reformat.sh script provided.
  - Step 2: The reformatted file was then prepended with the below header.

"*user_id user_name friend_count follower_count status_count favorite_count account_age user_location* "

  - Step3: The file from step 2 above is then imported directly into Mongo DB using import_mongodb.sh script provided.

- **NETWORK Collection**:
    - Step 1: The network.txt (tsv) raw data file was reformatted from ISO-8859-1 to UTF-8 format using the reformat.sh script provided.
    - Step 2: The reformatted file was then prepended with the below header.

      "*USER_ID1 USER_ID2* "

    - Step3: The file from step 2 above is then imported directly into Mongo DB using import_mongodb.sh script provided.

- **TWEETS_FIRST_DATA_MODEL Collection**:
    - Step 1: Fetch a file from the twitter data set directory where each file is a collection of tweets from a particular user and the file name is the User ID of the user.
    - Step 2: Extract all the tweets from a file in a String array using the delimiter "***\n***\n"

      ```
      String[] tweets = StringUtils.substringsBetween(content,
                                  "***\n***\n", "***\n***\n");
      ```

    - Step 3: Iterate over the tweets array and extract the below fields from each tweet string.
        - Type
        - Origin
        - Text
        - URL
        - ID
        - Time
        - RetCount
        - Favorite
        - MentionedEntities
        - HashTags

- Step 4: Prepare a BSON object using the data extracted in steps above.

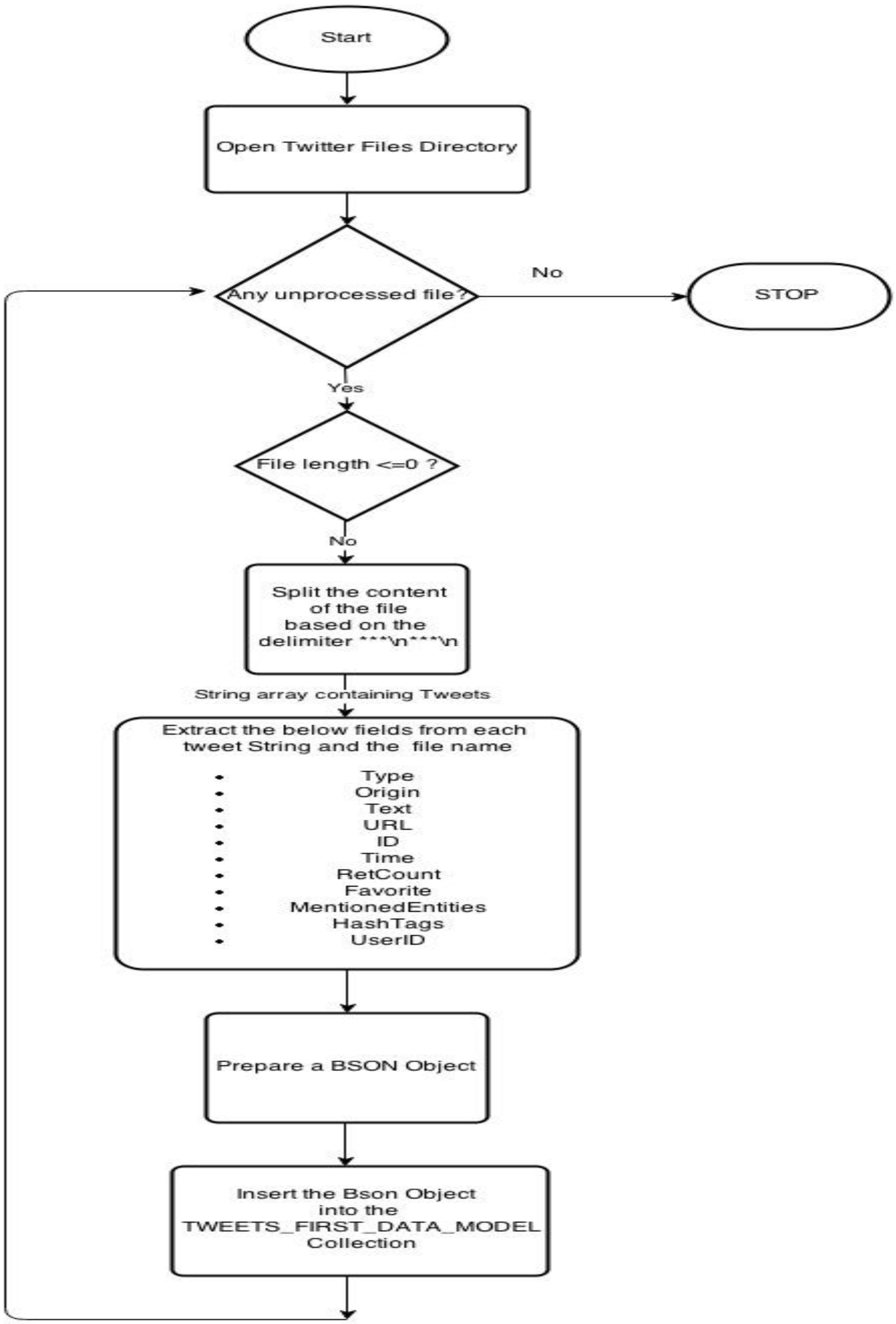
```
BasicDBObject document = new BasicDBObject("Type",type)
.append("Origin", origin)
.append("Text",text)
.append("URL", URL)
.append("ID",ID)
.append("Time",time)
.append("RetCount",retCount)
.append("Favorite",favourite)
.append("MentionedEntities",mentionedEntities)
.append("Hashtags",hashTags)
.append("UserID",fileName);
```

- Step 5: Insert the BSON object into the TWEETS_FIRST_DATA_MODEL Collection.

```
Collection.insert(document)
```

Repeat the Steps 1 to 5 till all the files in the twitter data set directory are processed.

3.6 Data loading flowchart for TWEETS_FIRST_DATA_MODEL Collection



3.7 Challenges faced during data loading and query operations

- *PROBLEM*: Identifying a proper delimiter to separate each tweets in a file from each other.
    - *SOLUTION*: “*****\n***\n**” was selected as the delimiter to separate tweets
- *PROBLEM*: 0 byte tweet files were present in the data set resulting in NULL pointer exception during file read operation.
    - *SOLUTION*: Included proper checks in the code to prevent processing 0 byte files to avoid NullPointerException.
- *PROBLEM*: Duplicate tweet data were present in the data set.
    - *SOLUTION*: Allowed Mongo DB generate unique id “_id” for each document.
- *PROBLEM*: Since the volume of data was huge and the Mongo DB was hosted on an external hard disk, data read operations were very slow.
    - *SOLUTION*: Created indices on relevant fields of the collections as shown below.

```
  show dbs
BIGDATA  99.905GB
admin     (empty)
local     0.078GB
testDB    0.078GB
> use BIGDATA
switched to db BIGDATA
> show collections
network
second_data_model
system.indexes
tweets_first_data_model
users
> db.system.indexes.find()
{ "v" : 1, "key" : { "_id" : 1 }, "name" : "_id_", "ns" : "BIGDATA.network" }
{ "v" : 1, "key" : { "USER_ID1" : 1, "USER_ID2" : 1 }, "name" : "USER_ID1_1_USER_ID2_1", "ns" : "BIGDATA.network" }
{ "v" : 1, "key" : { "_id" : 1 }, "name" : "_id_", "ns" : "BIGDATA.tweets_first_data_model" }
{ "v" : 1, "key" : { "UserID" : 1 }, "name" : "UserID_1", "ns" : "BIGDATA.tweets_first_data_model" }
{ "v" : 1, "key" : { "_id" : 1 }, "name" : "_id_", "ns" : "BIGDATA.users" }
{ "v" : 1, "key" : { "user_id" : 1 }, "name" : "user_id_1", "ns" : "BIGDATA.users" }
```

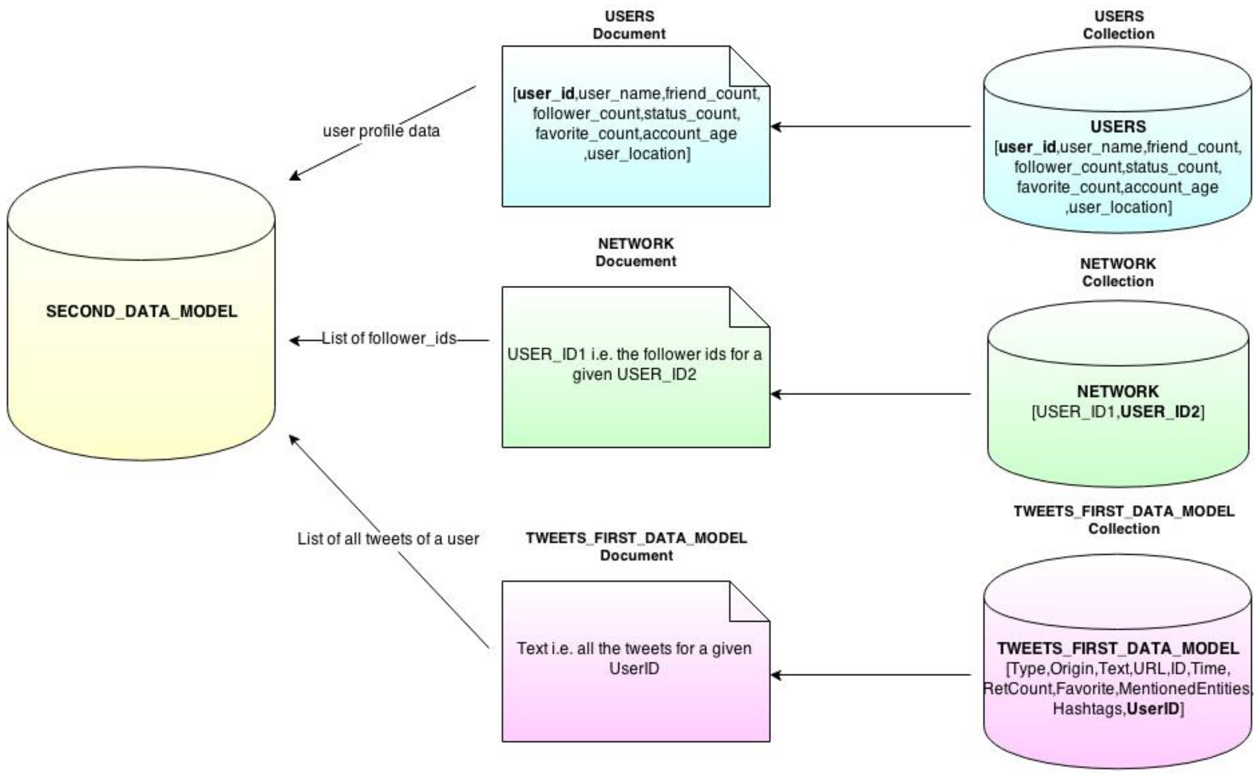
3.8 Data Loading Time

Collection Name	Method used	Time Taken
Tweets	Through MongoDB-JAVA driver	Approximately 3 hours
User Profile	Through mongo import	Approximately 4 minutes
Networks	Through mongo import	Approximately 5 hours

3.9 Query Time

Query	Execution time
a) Return all the user IDs from the tweets, which contain keyword KEYWORD in their text fields. Set the KEYWORD to a high-frequency word (e.g., “good”) first, then set it to a low-frequency word (e.g., “qwertyuiopasdfghjkl”) and run the query again. That is, running two queries for each data model.	For high frequency word “good”: **12 minutes** For low frequency word “qwertyuiopasdfghjkl”: **8 minutes**
b) Return cumulated retweet counts of all tweets, each of which has at least one hashtag.	Result : [{ "cumulatedRetweetCount" : 11953303}] : **7 minutes**
c) Select a user/users who has/have the largest number of followers, find all the followers in the network dataset, and return all the names of the followers (if these names can be retrieved from the profile dataset)	**19 minutes**
d) Add a follower to a user, update all the necessary collections.	**25 seconds**

4. Data Model 2 (Embedded):



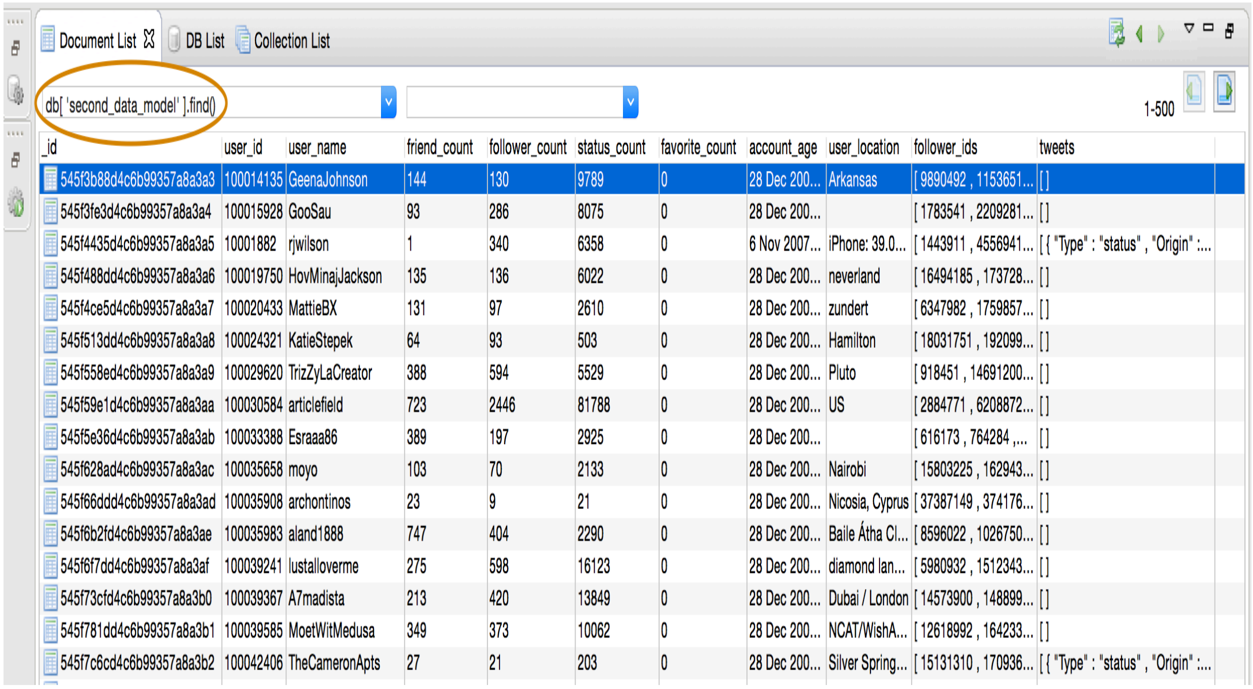
4.1 Data Model Design

- In Embedded data model, each document is a combination of documents from the USERS, NETWORK and TWEETS_FIRST_DATA_MODEL.
    - All the fields from USERS table are extracted for a given user document.
    - All the follower ids i.e. all the USER_ID1 values for a given user id from the USERS collection (USER_ID2 of NETWORK=user_id of USERS) is extracted from NETWORK collection and stored in a list. This list is then embedded into the “follower_ids” field of a document in the SECOND_DATA_MODEL collection.
    - All the tweets for a given user id from the USERS collection (UserID of TWEETS_FIRST_DATA_MODEL=user_id of USERS) are extracted from TWEETS_FIRST_DATA_MODEL collection and stored in a list. This list is then embedded into the “tweets” field of a document in the SECOND_DATA_MODEL collection.

```
{
   "user_id": 100014135,
   "user_name": "GeenaJohnson",
   "friend_count": 144,
   "follower_count": 130,
   "status_count": 9789,
   "favorite_count": 0,
   "account_age": "28 Dec 2009 18:25:37 GMT",
   "user_location": "Arkansas",
   "follower_ids": [⊞],
   "tweets": [⊞]
}
```

Embedded Documents

4.2 Tabular view representation



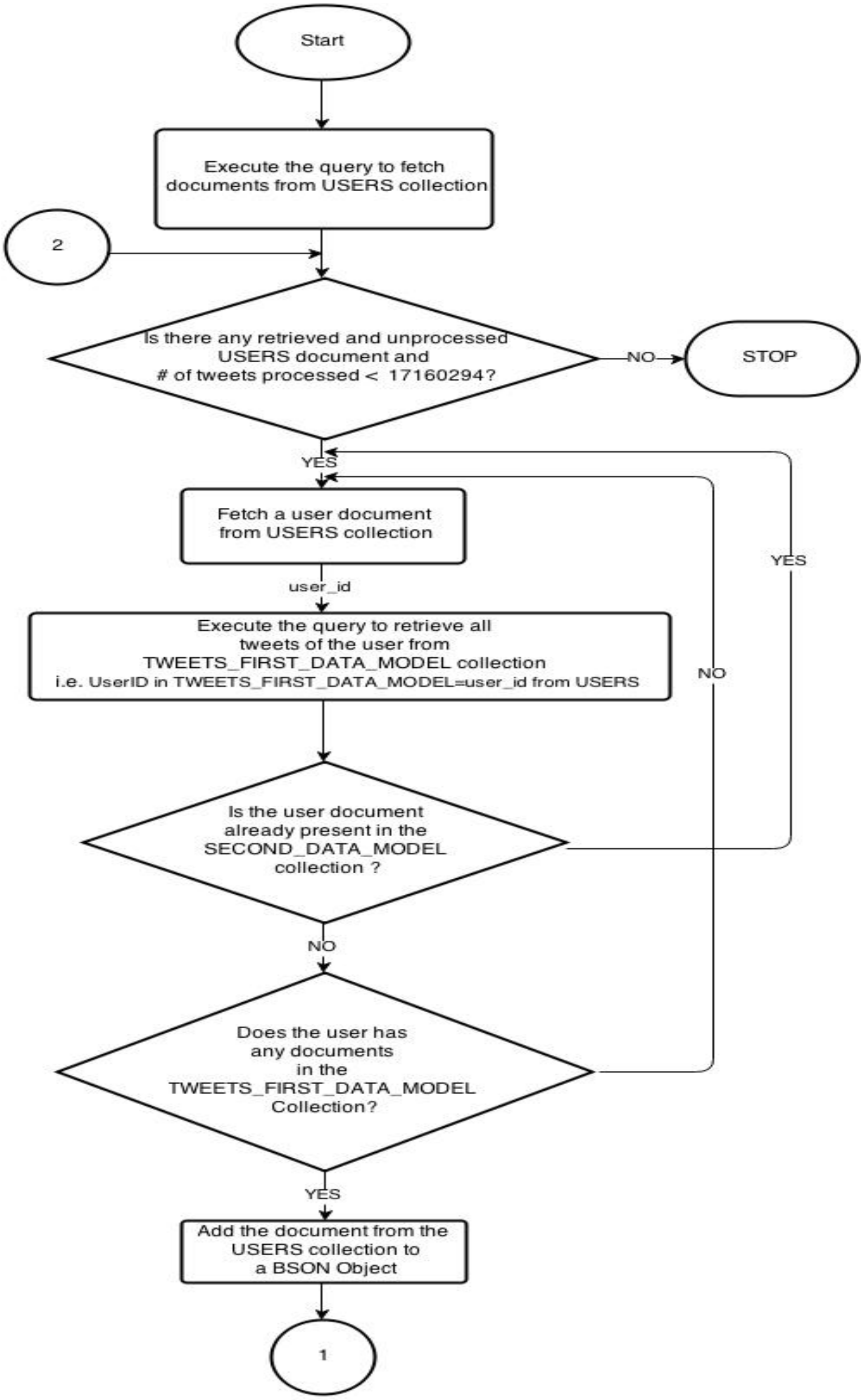
_id	user_id	user_name	friend_count	follower_count	status_count	favorite_count	account_age	user_location	follower_ids	tweets
545f3b88d4c6b99357a8a3a3	100014135	GeenaJohnson	144	130	9789	0	28 Dec 200...	Arkansas	[9890492 , 1153651...	[ ]
545f3fe3d4c6b99357a8a3a4	100015928	GooSau	93	286	8075	0	28 Dec 200...		[1783541 , 2209281...	[ ]
545f4435d4c6b99357a8a3a5	10001882	rjwilson	1	340	6358	0	6 Nov 2007...	iPhone: 39.0...	[1443911 , 4556941...	[{ "Type" : "status" , "Origin" :...
545f488dd4c6b99357a8a3a6	100019750	HovMinajJackson	135	136	6022	0	28 Dec 200...	neverland	[16494185 , 173728...	[ ]
545f4ce5d4c6b99357a8a3a7	100020433	MattieBX	131	97	2610	0	28 Dec 200...	zundert	[6347982 , 1759857...	[ ]
545f513dd4c6b99357a8a3a8	100024321	KatieStepek	64	93	503	0	28 Dec 200...	Hamilton	[18031751 , 192099...	[ ]
545f558ed4c6b99357a8a3a9	100029620	TrizZyLaCreator	388	594	5529	0	28 Dec 200...	Pluto	[918451 , 14691200...	[ ]
545f59e1d4c6b99357a8a3aa	100030584	articlefield	723	2446	81788	0	28 Dec 200...	US	[2884771 , 6208872...	[ ]
545f5e36d4c6b99357a8a3ab	100033388	Esraaa86	389	197	2925	0	28 Dec 200...		[616173 , 764284 ,...	[ ]
545f628ad4c6b99357a8a3ac	100035658	moyo	103	70	2133	0	28 Dec 200...	Nairobi	[15803225 , 162943...	[ ]
545f66ddd4c6b99357a8a3ad	100035908	archontinos	23	9	21	0	28 Dec 200...	Nicosia, Cyprus	[37387149 , 374176...	[ ]
545f6b2fd4c6b99357a8a3ae	100035983	aland1888	747	404	2290	0	28 Dec 200...	Baile Átha Cl...	[8596022 , 1026750...	[ ]
545f6f7dd4c6b99357a8a3af	100039241	lustalloverme	275	598	16123	0	28 Dec 200...	diamond lan...	[5980932 , 1512343...	[ ]
545f73cfd4c6b99357a8a3b0	100039367	A7madista	213	420	13849	0	28 Dec 200...	Dubai / London	[14573900 , 148899...	[ ]
545f781dd4c6b99357a8a3b1	100039585	MoetWitMedusa	349	373	10062	0	28 Dec 200...	NCAT/WishA...	[12618992 , 164233...	[ ]
545f7c6cd4c6b99357a8a3b2	100042406	TheCameronApts	27	21	203	0	28 Dec 200...	Silver Spring...	[15131310 , 170936...	[{ "Type" : "status" , "Origin" :...

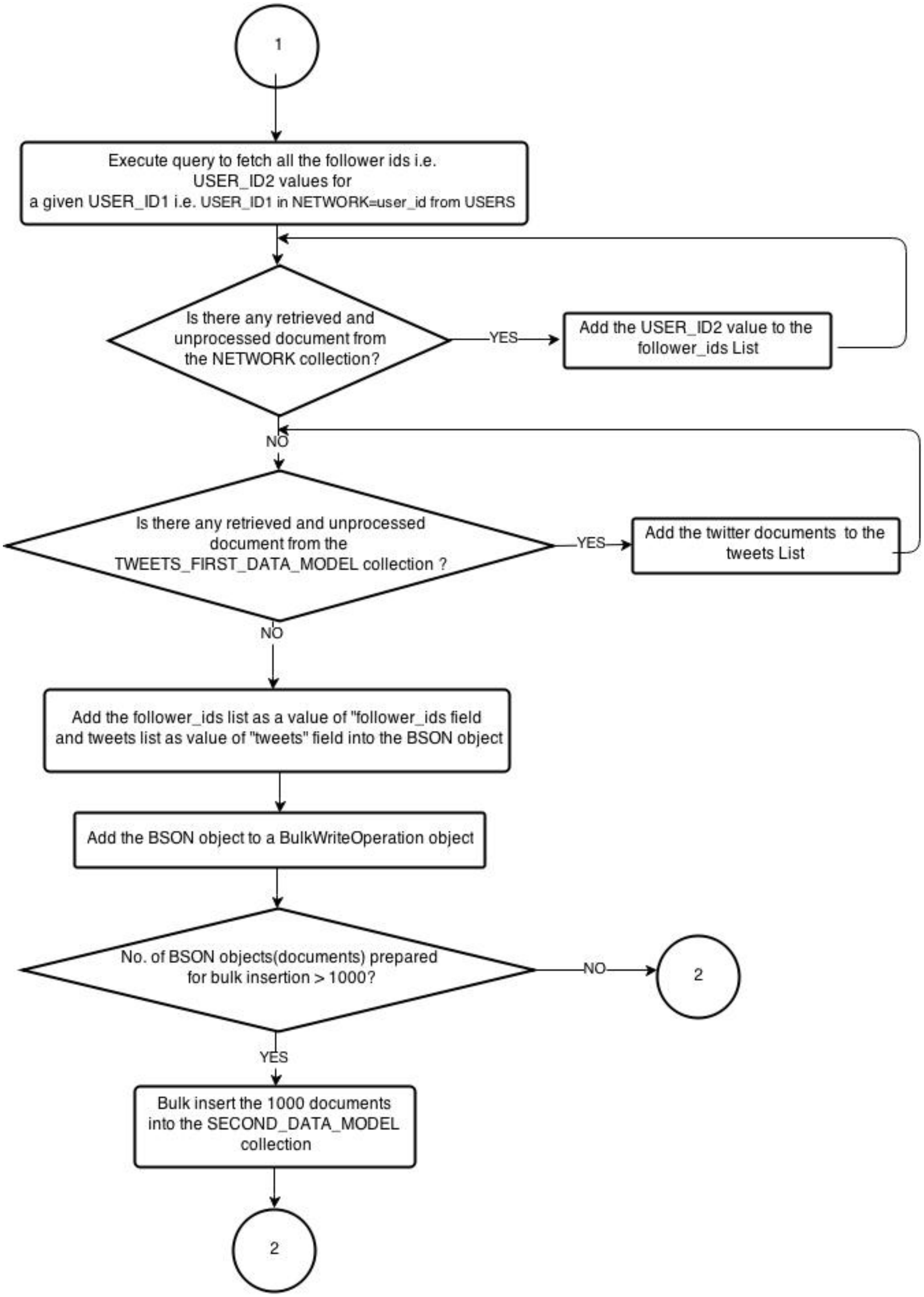
4.3 Data Loading:

NOTE:

- Due to space, time and hardware constraints, the second data model contains partial data unlike the first data model.
- It has total of 94413 out of 3123270 documents from USERS collection, 25736808 out of 284884526 documents from NETWORK collection and 17160294 out of 30881904 documents from TWEETS_FIRST_DATA_MODEL collection.

4.4 Flowchart for Embedded Data Model Data Loading Operation




1
Execute query to fetch all the follower ids i.e. USER_ID2 values for a given USER_ID1 i.e. USER_ID1 in NETWORK=user_id from USERS
Is there any retrieved and unprocessed document from the NETWORK collection?
YES
Add the USER_ID2 value to the follower_ids List
NO
Is there any retrieved and unprocessed document from the TWEETS_FIRST_DATA_MODEL collection ?
YES
Add the twitter documents to the tweets List
NO
Add the follower_ids list as a value of "follower_ids field and tweets list as value of "tweets" field into the BSON object
Add the BSON object to a BulkWriteOperation object
No. of BSON objects(documents) prepared for bulk insertion > 1000?
NO
2
YES
Bulk insert the 1000 documents into the SECOND_DATA_MODEL collection
2

4.5 Challenges faced during data loading operations

- PROBLEM: There were 231 erroneous records in the users.txt file whose user id field was having random string characters instead of a numeric value and blank value for other fields. Moreover, these records were resulting in class cast exception and number format exception while preparing the SECOND_DATA_MODEL collection since the user_id field was of numeric type in the users and network collection.
    - SOLUTION: Remove these records from the USERS collection in data pre-processing step.
- PROBLEM: Since the volume of data was huge and Mongo DB was hosted on an external hard disk, data write operations were very slow.
    - SOLUTION: Used Bulk Write feature of Mongo DB JAVA driver to bulk insert 1000 documents at a time in SECOND_DATA_MODEL collection resulting in approx. 50% reduction in data loading time. Also, loaded those user documents who have any tweets to save the time wasted in processing and loading user documents that have no tweets.
- PROBLEM: Due to the huge volume of fata and Mongo DB been hosted on an external hard drive, data read operations were slow.
    - SOLUTION: Created index on user_id field of the SECOND_DATA_MODEL collection to improve the query performance.

```
  show dbs
BIGDATA  99.905GB
admin     (empty)
local     0.078GB
testDB    0.078GB
> use BIGDATA
switched to db BIGDATA
> show collections
network
second_data_model
system.indexes
tweets_first_data_model
users
> db.system.indexes.find()
{ "v" : 1, "key" : { "_id" : 1 }, "name" : "_id_", "ns" : "BIGDATA.network" }
{ "v" : 1, "key" : { "USER_ID1" : 1, "USER_ID2" : 1 }, "name" : "USER_ID1_1_USER_ID2_1", "ns" : "BIGDATA.network" }
{ "v" : 1, "key" : { "_id" : 1 }, "name" : "_id_", "ns" : "BIGDATA.tweets_first_data_model" }
{ "v" : 1, "key" : { "UserID" : 1 }, "name" : "UserID_1", "ns" : "BIGDATA.tweets_first_data_model" }
{ "v" : 1, "key" : { "_id" : 1 }, "name" : "_id_", "ns" : "BIGDATA.users" }
{ "v" : 1, "key" : { "user_id" : 1 }, "name" : "user_id_1", "ns" : "BIGDATA.users" }
```

4.6 Data Loading Time

- It took approximately **3 days** to load total 94413 out of 3123270 documents from USERS collection, 25736808 out of 284884526 documents from NETWORK collection and 17160294 out of 30881904 documents from TWEETS_FIRST_DATA_MODEL collection embedded together into SECOND_DATA_MODEL collection.

4.7 Query Execution Time

Query	**Execution time**
a) Return all the user IDs from the tweets, which contain keyword KEYWORD in their text fields. Set the KEYWORD to a high-frequency word (e.g., “good”) first, then set it to a low-frequency word (e.g., “qwertyuiopasdfghjkl”) and run the query again. That is, running two queries for each data model.	For high frequency word “good”: **104 milliseconds** For low frequency word “qwertyuiopasdfghjkl”: **142 milliseconds**
b) Return cumulated retweet counts of all tweets, each of which has at least one hashtag.	Result : [{ "cumulatedRetweetCount" : 56936}]: **10 milliseconds**
c) Select a user/users who has/have the largest number of followers, find all the followers in the network dataset, and return all the names of the followers (if these names can be retrieved from the profile dataset)	**1 minute 4 seconds**
d) Add a follower to a user, update all the necessary collections.	**6.972 seconds**

4.8 Reference Data Model VS Embedded Data Model

- Reference data models are a good choice when the document size is huge since in this data model a document is normalized and stored in separate collections linked using references. Since, Mongo DB has an upper limit of 16 MB for a document size, embedded data model wont be a good choice when document size is > 16MB.

- Reference data models provide greater flexibility in querying i.e. supports more complex query operations than embedded data models.

- Reference data model's select query operations require more time than embedded data models since multiple disk seek operations need to be performed - one for each collection involved. Whereas Embedded data model supports single seek query operation since all the relevant data are embedded into a single document and hence are faster. Thus, embedded data models are preferred when read intensive operations are to be frequently performed.

- Mongo DB implements atomic write operations. So in the case of Insert / Update operations, embedded data models perform better since all the relevant data are stored as a single document resulting in less disk seek operations. Also, since embedded data models favor atomic write operations, they are a better choice for data consistency.

  But, if there is a possibility of a document to exceed >16MB size during Update operations, embedded data model will perform slow since it will involve an overhead of allocating a new space in the disk for the updated document and copying the old document to it followed by appending the old document with new data. So in such cases reference data model is a better choice.