Comp 5338 – Assignment: Polyglot Persistence with NoSQL Systems

# Section 1 – Introduction

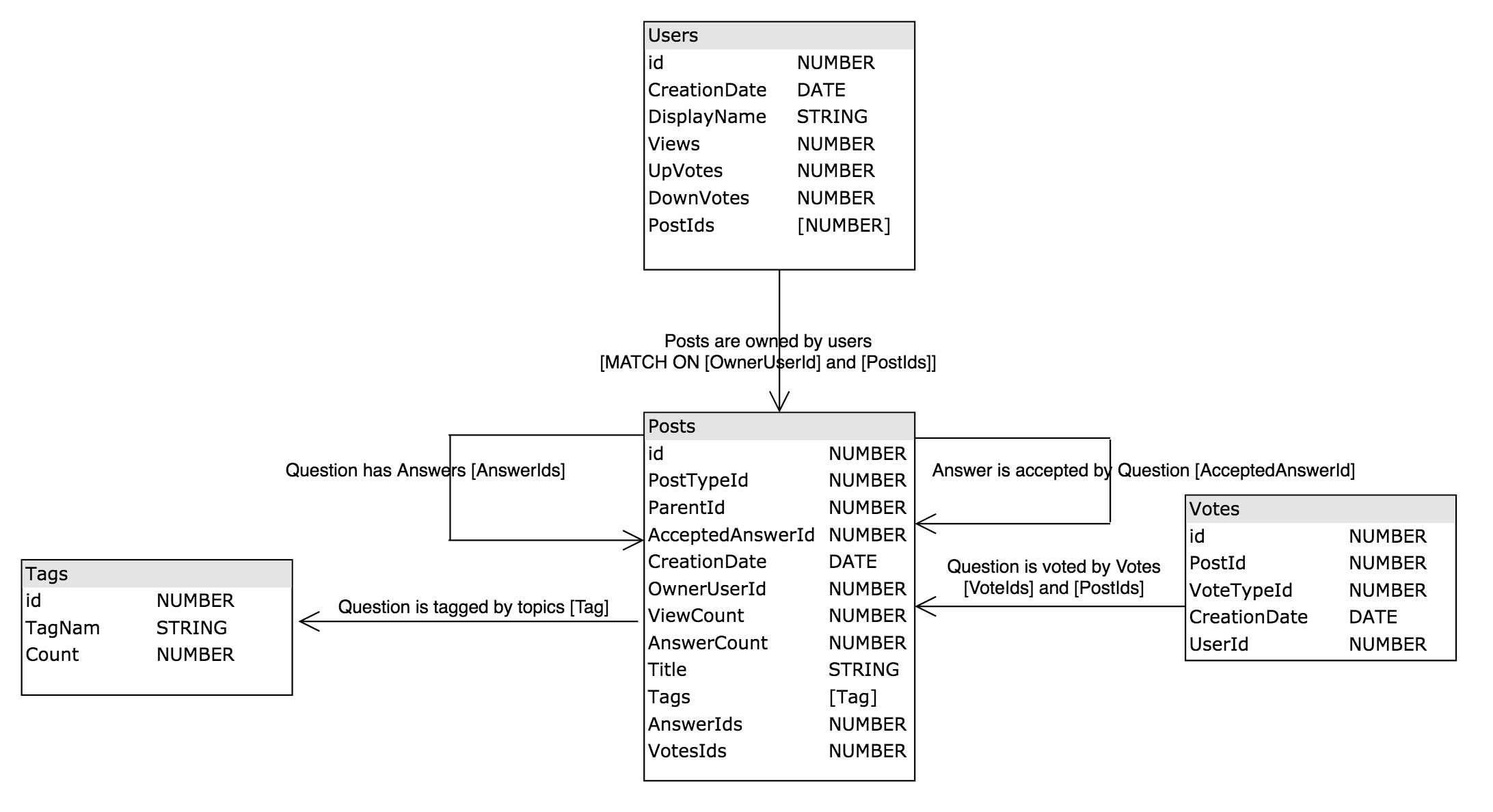
This report will briefly introduce the schema designs for both MongoDB and neo4j storage systems for this assignment, then followed by the query design and execution using command line tool to fetch the correct result given simple queries and analytics queries, at last, this report will compare these two storage systems with their pros and cons regarding the query performances, data schema design etc. You can also find the source code in the following Github link:

<https://github.com/MingxuanLi/comp5338-polygot-persistence-systems>

# Section 2 – Schema Design

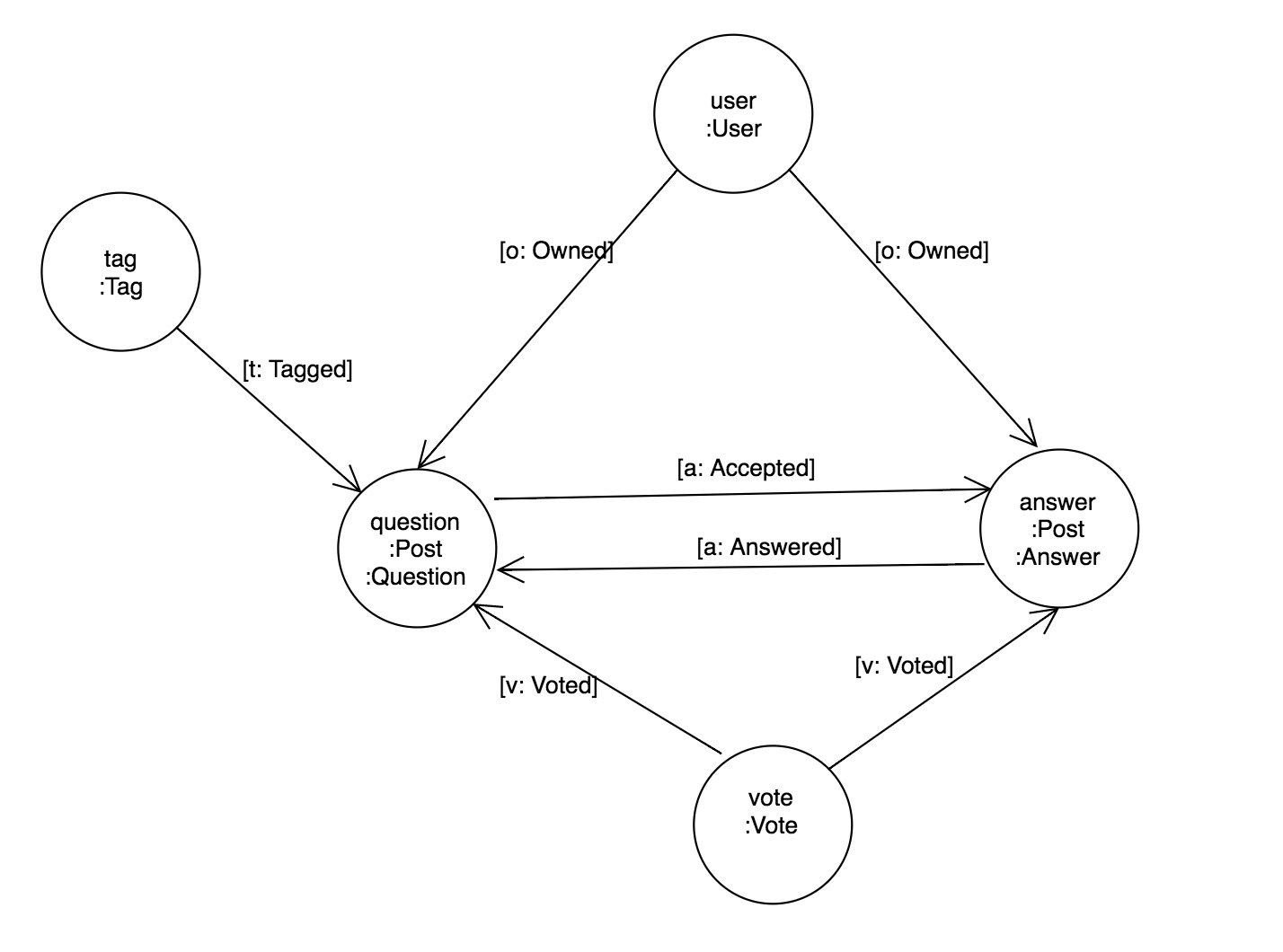
The following diagrams briefly describe the data model schema and their relationships in both MongoDB and neo4j.

## Section 2.1 – MongoDB Schema Design



For MongoDB part, we use mongoose to explicitly design the schema in the javascript files, the schemas are under ***src/mongo-schemas***.

## Section 2.2 – neo4j Schema Design



For neo4j part, all the schemas for loading the data are in ***src/neo4j-helper.js*** file. We also convert the iso date into unix timestamp as neo4j does not support date by default.

# Section 3 – Query Design and Execution

## Section 3.1 – Query Design

## Section 3.2 – Query Execution

There are two major execution steps:

* Loading data
* Executing Queries

First, you need to use Git to check this repo using the following command:

***git clone*** [***https://github.com/MingxuanLi/comp5338-polygot-persistence-systems.git***](https://github.com/MingxuanLi/comp5338-polygot-persistence-systems.git)

After that you need to install the library dependencies we use to import and query data from MongoDB and neo4j, first you need to install **node.js 7.8.x** above and **npm 5.0** above here:

<https://nodejs.org/en/download/current/>

<https://docs.npmjs.com/getting-started/installing-node>

Then run the following:

***npm install***

After the installing finish, you can try loading the data into MongoDB by running following:

***npm run load -- --db=mongodb***

And neo4j loading command is this:

***npm run generate*** – (this does some conversion for the unix date format for neo4j)

***npm run load -- --db=neo4j***

After the loading is finished, then you can use the following query to fetch result.

For MongoDB: ***npm run query -- --db=mongodb --query={$name}***

For neo4j: ***npm run query -- --db=neo4j --query={$name}***

The variable $name can be **‘sq1’, ‘sq2’, ‘aq1’, ‘aq2’, ‘aq3’, ‘aq4’, ‘aq5\_1’, ‘aq5\_2’, ‘aq6’**.

sq – simple query, aq – analytic query.

The result of the query should be printed in the terminal console.

Inside the query files in ***src/mongo-queries***, you can change the input params in each file to query different result sets, neo4j queries are similar.

# Section 4 – Comparison and Summary

Overall speaking, the neo4j query is short and compact than the mongo query since it’s relational database so it’s quite easy to query data with relationships, also we observed that the performance of querying in neo4j is better than mongodb. Now we take the third analytic query for example.

Query Compare:

The aggregation query on mongodb for querying the champion user questions (analytic query 3) is pretty long, we use a couple of $project with $filters, $unwind and $group to find the correct result set, those operations are pretty performance costly especially doing the $unwind.

[  
 {  
 $match: {  
 PostTypeId: 1,  
 AcceptedAnswerId: {"$exists": **true**, "$ne": **null**}  
 }  
 },  
 {  
 $project: {  
 questionId: "$Id",  
 questionTitle: "$Title",  
 acceptedAnswerId: "$AcceptedAnswerId",  
 tags: {  
 $filter: {  
 input: "$Tags",  
 as: "item",  
 cond: {  
 "$setIsSubset": [["$$item.Id"], [4]]  
 }  
 }  
 }  
 }  
 },  
 {  
 $match: {  
 "tags.0": {"$exists": **true**}  
 }  
 },  
 {  
 $unwind: "$tags"  
 },  
 {  
 $lookup: {  
 from: "posts",  
 localField: "acceptedAnswerId",  
 foreignField: "Id",  
 as: "acceptedAnswer"  
 }  
 },  
 {  
 $unwind: "$acceptedAnswer"  
 },  
 {  
 $lookup: {  
 from: "users",  
 localField: "acceptedAnswer.OwnerUserId",  
 foreignField: "Id",  
 as: "acceptedUser"  
 }  
 },  
 {  
 $unwind: "$acceptedUser"  
 },  
 {  
 $sort: {  
 "acceptedUser.Id": 1  
 }  
 },  
 {  
 $group: {  
 \_id: {  
 userId: "$acceptedUser.Id",  
 displayName: "$acceptedUser.DisplayName"  
 },  
 acceptedQuestions: {  
 $addToSet: {  
 questionId: "$questionId",  
 questionTitle: "$questionTitle",  
 tags: "$tags"  
 }  
 }  
 }  
 },  
 {  
 $project: {  
 \_id: "$\_id",  
 acceptedQuestions: "$acceptedQuestions",  
 numOfAcceptedQuestions: {  
 $size: "$acceptedQuestions"  
 }  
 }  
 },  
 {  
 $sort: {"numOfAcceptedQuestions": -1}  
 },  
 {  
 $limit: 1  
 }  
]

While the neo4j query for this question set is pretty simple and clean:

// Input Param, Please change it for different queries  
**const** tagId = 4;  
  
**const** query = `  
 MATCH (user:User)-[:Owned]->(answer:Answer)<-[:Accepted]-(question:Question)-[:Contains]->(tag:Tag{Id:${tagId}})  
 WITH user, tag, count(\*) as answersCount  
 ORDER by answersCount DESC  
 LIMIT 1  
 MATCH (user)-[:Owned]->(answer:Answer)<-[:Accepted]-(question:Question)-[:Contains]->(tag)  
 RETURN user.Id as userId, user.DisplayName as userDisplayName, question.Id as questionId, question.Title as questionTitle  
`;

Performance Compare:

Then we run both query in mongodb and neo4j by using our command line tool, see below, we can see the mongodb query cost ***405*** milliseconds to complete this query and return result set, but if we run the neo4j for same question set it only takes ***59*** milliseconds to fetch the champion user with their question list, we can see that neo4j is much faster than mongodb in handling relational graph database.

Running mongodb query result below:

mingxuanli@192-168-1-4 comp5338-polygot-persistence-systems (master)\*$ npm run query -- --db=mongodb --query=aq3  
  
> comp5338-polygot-persistence-systems@1.0.0 query /Users/mingxuanli/repo/usyd/comp5338-polygot-persistence-systems  
> node ./src/command-line.js --action=query "--db=mongodb" "--query=aq3"  
  
Used Time:405  
Query Result:  
[  
 {  
 "\_id": {  
 "userId": 33,  
 "displayName": "fgregg"  
 },  
 "acceptedQuestions": [  
 {  
 "questionId": 1090,  
 "questionTitle": "How do you respond when government cites time concerns for not releasing data?",  
 "tags": {  
 "Id": 4,  
 "TagName": "government",  
 "Count": 142,  
 "\_id": "59dbf47d99d6d00819376194"  
 }  
 },  
 {  
 "questionId": 7840,  
 "questionTitle": "Is there an open standard for the CAFR (Comprehensive Annual Finance Report)?",  
 "tags": {  
 "Id": 4,  
 "TagName": "government",  
 "Count": 142,  
 "\_id": "59dbf48999d6d00819378a18"  
 }  
 },  
 {  
 "questionId": 961,  
 "questionTitle": "What cities provide open data on rental building bylaw infractions?",  
 "tags": {  
 "Id": 4,  
 "TagName": "government",  
 "Count": 142,  
 "\_id": "59dbf47d99d6d008193760aa"  
 }  
 },  
 {  
 "questionId": 1091,  
 "questionTitle": "How do you respond when government says it should be selling its data, not opening it?",  
 "tags": {  
 "Id": 4,  
 "TagName": "government",  
 "Count": 142,  
 "\_id": "59dbf47d99d6d00819376198"  
 }  
 },  
 {  
 "questionId": 1092,  
 "questionTitle": "How do you respond when government says it needs more proven results to release data?",  
 "tags": {  
 "Id": 4,  
 "TagName": "government",  
 "Count": 142,  
 "\_id": "59dbf47d99d6d0081937619c"  
 }  
 },  
 {  
 "questionId": 1089,  
 "questionTitle": "How do you respond when government cites costs for not releasing data?",  
 "tags": {  
 "Id": 4,  
 "TagName": "government",  
 "Count": 142,  
 "\_id": "59dbf47d99d6d00819376190"  
 }  
 }  
 ],  
 "numOfAcceptedQuestions": 6  
 }  
]

For neo4j, if we run the query for the third analytic query, the result is below:

mingxuanli@192-168-1-4 comp5338-polygot-persistence-systems (master)\*$ npm run query -- --db=neo4j --query=aq3  
  
> comp5338-polygot-persistence-systems@1.0.0 query /Users/mingxuanli/repo/usyd/comp5338-polygot-persistence-systems  
> node ./src/command-line.js --action=query "--db=neo4j" "--query=aq3"  
  
Used Time:59  
Query Result:  
[  
 {  
 "userId": 33,  
 "userDisplayName": "fgregg",  
 "questionId": 7840,  
 "questionTitle": "Is there an open standard for the CAFR (Comprehensive Annual Finance Report)?"  
 },  
 {  
 "userId": 33,  
 "userDisplayName": "fgregg",  
 "questionId": 961,  
 "questionTitle": "What cities provide open data on rental building bylaw infractions?"  
 },  
 {  
 "userId": 33,  
 "userDisplayName": "fgregg",  
 "questionId": 1092,  
 "questionTitle": "How do you respond when government says it needs more proven results to release data?"  
 },  
 {  
 "userId": 33,  
 "userDisplayName": "fgregg",  
 "questionId": 1089,  
 "questionTitle": "How do you respond when government cites costs for not releasing data?"  
 },  
 {  
 "userId": 33,  
 "userDisplayName": "fgregg",  
 "questionId": 1090,  
 "questionTitle": "How do you respond when government cites time concerns for not releasing data?"  
 },  
 {  
 "userId": 33,  
 "userDisplayName": "fgregg",  
 "questionId": 1091,  
 "questionTitle": "How do you respond when government says it should be selling its data, not opening it?"  
 }  
]

# Section 5 – References

<https://docs.mongodb.com/manual/applications/data-models/>

<https://www.mongodb.com/blog/post/6-rules-of-thumb-for-mongodb-schema-design-part-1>

<https://www.mongodb.com/blog/post/6-rules-of-thumb-for-mongodb-schema-design-part-2>

<https://www.mongodb.com/blog/post/6-rules-of-thumb-for-mongodb-schema-design-part-3>

<https://neo4j.com/developer/javascript/>

<https://neo4j.com/developer/guide-data-modeling/>