THE UNIVERSITY OF SYDNEY

School of Information Technologies
Dr. Ying Zhou

**COMP5338: Advanced Data Models** **2.Sem./2018**

# Project: NoSQL Schema Design and Query Workload Implementation

**Group Work: 20%** **11.09.2018**

## Introduction

In this assignment, you will demonstrate that you are able to work with both MongoDB and Neo4j in terms of designing suitable schema and writing practical queries. You will also demonstrate that you understand the strength and weakness of each system with respect to certain query workload features. You will be given a real world data set in *Question and Answer* area and a set of target queries. The target queries include very basic OLTP type queries and analytic queries.

You are asked to design two storage options: one with MongoDB as the storage system and the other with Neo4j as the storage system. For each option, you need to store a full copy of the data in the system and implement a subset of the target queries.

## Data set

The data that you will use is the latest dump (publication date: 2018-06-05) of the `Artificial Intelligence Stack Exchange` question and answer site (`https://ai.stackexchange.com/`). The dump is released and maintained by stackexchange: `https://archive.org/details/stackexchange`. The original dump contains many files in XML format. The assignment uses a subset of the data stored in four tsv files. The data files and the description (`readme.txt`) can be downloaded from Canvas.

The assignment data set contains the following files:

- `Posts.tsv` stores information about post, each row represents a post, which could be a question or an answer
- `Users.tsv` stores user's profile, each row represents a user
- `Votes.tsv` stores detailed vote information about post, each row represents a vote, including the vote type, the date this vote is made and a few other information
- `Tags.tsv` contains summary of tag usage in this site.

# Target Queries

**Simple queries**

- `[SQ1]` Find all users involved in a given question (identified by id) and their respective profiles including the creationDate, DisplayName, upVote and DownVote. Note, we are only interested in existing users who either posted or answered the question. You may ignore users that do not have an id.

- `[SQ2]` Assuming each tag represents a topic, find the most viewed question in a given topic.[1]

**Analytic queries**

- `[AQ1]` Given a list of topics (tags), find the question easiest to answer in each topic.

  We measure easiness using the time it took to receive an accepted answer. The shorter the time, the easier the question. For instance, the question easiest to answer in topic 'neural-networks' is question with id 1: `What is ‘‘backprop’’?`. The question was posted on `2016-08-02T15:39:14` and received an accepted answer on `2016-08-02T15:40:24`. It took only a little over 1 minute to receive an accepted answer.

- `[AQ2]` Given a time period as indicated by starting and ending date, find the top 5 topics in that period.

  We rank a topic by the number of users that has either posted a question in that topic or has answered a question in that topic. This would help us to understand the trending topics in different periods of time. For instance, the trending topics and respective user numbers in August, 2018 as indicated by start date `2018-08-01T00:00:00` and end date `2018-08-31T00:00:00` are:

| Topic | User Number |
|---|---|
| neural-networks | 65 |
| machine-learning | 44 |
| deep-learning | 39 |
| reinforcement-learning | 28 |
| convolutional-neural-networks | 24 |

- `[AQ3]` Given a topic, find the champion user and all questions the user has answers accepted in that topic.

  We define a champion user as the one who has the most answers accepted in a given topic. For instance, the champion user of topic 'deep-learning' is 4398 and 1847. Both users have 9 answers being accepted in this topic. Your result may show either of the two users. Below is an example of the questions user 4398 has answers accepted:

[1] For any query, if there are more results than the specified limit, e.g. two questions have the highest view count but we only want one, you can return any of the valid results within the limit

| Id | Title |
|---|---|
| 3402 | Is there ever a need to combine deep learning ... |
| 3453 | What are the pros and cons of using a spatial ... |
| 4080 | Is it necessary to clear the replay memory reg... |
| 4085 | Policy gradients for multiple continuous actions |
| 4167 | Reinforcement learning for robotic motion plan... |
| 4346 | Is the new Alpha Go implementation using Gener... |
| 4425 | Deep Learning - Classification or Regression A... |
| 4740 | I don't understand the "Target Network" in Dee... |
| 5185 | What is the purpose of the GAN |

- `[AQ4]` Recommend unanswered questions to a given user.

  Some question may have been posted for a period of time but may not have an accepted answers yet. We refer to such question as unanswered question. We would like to recommend unanswered questions to potential answerers. For any user with $n$ answers accepted in a certain topic with $n$ greater than or equal to a threshold value $\alpha$, we consider the user a potential answerer of unanswered questions in that topic.

  For instance, given a user with id 4398, an $\alpha$ value 4 and a cutoff date of question creation as `2018-08-30T00:00:00`. We will find user 4398 is a potential answerer in the following topics: reinforcement-learning, deep-learning, machine-learning and ai-design. The user has 10 answers accepted in reinforcement-learning area, 9 in deep-learning area, 5 in machine-learning area and 4 in ai-design area. The most 5 recent unanswered question in those topics that are posted before `2018-08-30T00:00:00` are:

| Id | Title | CreationDate |
|---|---|---|
| 7755 | How to implement a constrained action space in... | 2018-08-29 16:04:16.113 |
| 7737 | creating application to transform human comput... | 2018-08-28 00:17:55.907 |
| 7736 | In imitation learning, do you simply inject op... | 2018-08-27 18:41:56.223 |
| 7734 | AI composing music | 2018-08-27 16:13:16.433 |
| 7727 | How is it possible to teach a neural network t... | 2018-08-27 10:18:17.893 |

  They should be the questions recommended to user 4398.

- `[AQ5]` Discover questions with arguable accepted answer.

  Users can give upVote to both question and answer. Usually the accepted answer of a question receives the highest number of upVote among all answers of this question. In rare case, another answer(s) may receive higher upVote count than the upVote count of the accepted answer. In this query, you are asked to discover such questions whose accepted answer has less upVote count than the upVote count of one of its other answers. Note We are only interested in questions with upVote count greater than a given threshold value $\alpha$. With high $\alpha$ value, you are likely to get an empty set as the result. A reasonable $\alpha$ value would be between 5-15. Your result should show the

question id, the upVote count of its accepted answer, and the highest upVote count received by other answers.

- `[AQ6]` Discover the top five coauthors of a given user.

  Consider all users involved in a question as co-authors, for a given user, we rank the coauthors by the number of questions this user and the coauthor appear together either as originator or answerer. For instance, user 4398 has the following top co-authors: 1671(5), 11571(4), 9161(4), 4302(3) and 6019(3). User 4398 and user 1671 appeared together in five questions; user 4398 and user 1571 appeared together in 4 questions. Your result should include both the user id and the number of questions the pair appeared together (co-authored).

## Tasks

Your tasks include:

- Decide on the query workload to implement in each option

  Among all eight target queries, you are required to implement five for each storage system option. These include the two simple queries: [SQ1] and [SQ2] and three analytic queries. Note that for any analytic query, you need to implement it either in MongoDB or Neo4j. Below is an example of valid query workload combination:

  - MongoDB query workload: {`SQ1, SQ2, AQ1, AQ2, AQ3`}
  - Neo4j query workload: {`SQ1, SQ2, AQ4, AQ5, AQ6`}

  Below is an example of not valid combination, because `AQ5` is not implemented in any system:

  - MongoDB query set: {`SQ1, SQ2, AQ1, AQ2, AQ3`}
  - Neo4j query set: {`SQ1, SQ2, AQ3, AQ4, AQ6`}

- Schema Design for MongoDB and Neo4j

  For each storage option design a proper schema that would best support the workload and data set feature.

  For each schema version, make sure you utilize features of the storage system such as indexing, aggregation, ordering, filtering and so on. Please note that your schema may deviate a lot from the relational structure of the original data set. You will not get point if you present a schema that is an exact copy of the relational structure in the original data set.

  You may discard data that are not useful or not involved in the query. You may duplicate data if you find that helps with certain queries. You may run preprocessing

to reorganize the data to allow easy importing. You should avoid running preprocessing to generate partial or final results for any target query. You need to justify any preprocessing you have employed. The justification should include benefits, potential performance cost as well as how often the preprocessing should be carried out. Keep in mind that the storage system is also used to support online live transaction. Any activity such as posting or voting will send one or more write queries to the system. You should not use preprocessing that would cause significant delay to regular queries. The data set is stored with utf-8 encoding. Please ensure that your preprocessing script would save data back in the same encoding. This is important for the 'posts.tsv' and 'users.tsv' files, which contain non-ascii characters.

- Query Design and Implementation

  Load the full data set (after some necessary preporocessing) into both systems and set up proper indexes that will be used by the target queries. Design and implement the chosen query workload for each system. You may implement a query using the shell command (e.g. MongoDB shell or Cypher query) alone, or a combination of JavaScript and shell commands in the case of MongoDB or as Python/Java program. In case that a programming language is used, make sure that you do majority of the processing on the database side. The client side processing should be restricted to activities like collecting output from previous database query and send the output as is to the subsequent one. In particular, you should avoid sorting, filtering and grouping query output on the client side.

## Deliverable and Submission Guideline

This is a group project, each group can have up to 2 students. Each group needs to produce the following:

- **A Written Report** .
  The report should contain five sections. The first section is a brief introduction of the project. Section two and three should cover a storage option each. Section four should provide a comparison and summary. Section five should be an appendix for sample results.

  There is no point allocated on section one. It is included to make the report complete. So please keep it short.

  Section two and three should contain the following three sub sections

  - **Query Workload**
    In this section, briefly describe the query workload you have chosen to work on the system. You may add short explanation on your selection criteria.
  - **Schema Design**
    In this section, describe the schema with respect to the particular system. Your

description should include information at "table" and "column" level as well as possible primary keys/row keys and secondary indexes. You should show sample data based on schema. For instance, you may show sample documents of each MongoDB collection, a sample property graph involving all node types and relationship types for Neo4j. Highlight the data that are different to the given one and justify your decision. These would include data you decide not to put in the database, data that are duplicated, or data that are results of preprocessing.

- **Query Design and Execution**
  In this section, describe implementation of each target query in the workload. You may include the entire shell query command, or show pseudo code. You should also run sample queries and provide a brief performance analysis of each query by citing actual query execution statistics if such statistics are available or by indicating that certain indexing would be used.

In section four, compare the two systems with respect to ease of use, query performance and schema differences. You can also describe problems encountered in schema design or query design.

In section five, document the sample query results as well as the respective argument(s) you use for target queries.

Submit a hard copy of your report together with a signed group assignment sheet in Week 10 lab.

- **System Demo**
  Each group will demo in week 10 lab. You can run demo on your own machine, on lab machine or on some cloud servers. Please make sure you prepare the data before the demo. The marker does not need to see your data loading steps. The marker will ask you to run a few randomly selected queries to get an overview of the data model and query design. All members of the group are required attend the demo. The marker will ask each member a few questions to establish their respective contribution to the project. Members in the same group may get different marks depending on their individual contributions.

- **Source Code/Script and soft copy of report**
  Submit a zip file through the eLearning site, before 5pm on Tuesday $^{9th}$ of October, 2018 (week 10). The zip file should contain the following:

  - a soft copy of the report
  - query script or program code for each option
  - any data preprocessing and loading script.
  - a `Readme` document for how to run the preprocessing script and the target queries. The instruction should be detailed enough for the markers to quickly prepare the data and to run the queries. For instance, you should indicate where and how

run-time argument are supplied. If you use special features only available in a particular version or environment, indicate that as well.