THE UNIVERSITY OF SYDNEY

School of Information Technologies
Dr. Ying Zhou

COMP5338: Advanced Data Models **2.Sem./2017**

Assignment: Polyglot Persistence with NoSQL Systems

Group Work: 20% **12.09.2017**

Introduction

In this assignment, you will show that you can work with different NoSQL systems for data persistence and understand the strength and weakness of each system with respect to certain query workload features. You are asked to work with a *Question and Answer* data set and a list of target queries. The target queries include very basic OLTP type queries and analytic queries. You will design data schema based on the data set feature and the given query workload for MongoDB and Neo4j respectively. You will show that your design can support the target queries by loading the data in each system following your schema and by running queries against the data.

Data set

The data that you will use is the latest dump (publication date: 2017-08-31) of the `Open Data Stack Exchange` question and answer site (`https://opendata.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 csv files. The data files and the description (`readme.txt`) can be downloaded from ELearning.

The assignment data set contains the following files:

- `Posts.csv` stores information about post, each row represents a post, which could be a question or an answer
- `Users.csv` stores user's profile, each row represents a user
- `Votes.csv` 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.csv` contains summary of tag usage in this site.

Target Queries

Simple queries

- Find all users involved in a given question (identified by id) and their respective profiles including the creationDate, DisplayName, upVote and DownVote.

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

Analytic queries

- We want to discover the questions that are easy to answer. We measure easiness with the time it took to receive an accepted answer. The shorter the time, the easier the question. Given a list of topics, you are asked to list the question easiest to answer in each topic.

- We are interested in discovering trending topics in a certain period. An indicator of trending could be the number of users who have participated in the discussion, by posting either questions or answers, during the period. You are asked to list five such hottest topics in a given period as well as the number of users in each.

- We are also interested in discovering champion user in a certain topic. A champion user has the most answers accepted in that topic. Given a topic, you are asked to list the champion user and all questions the user has answers accepted in that topic.

- Some questions may not have accepted answers. They are considered as unanswered questions. We would like to recommend those questions to a group of potential answerer. For any active user, if the user has n answers accepted in a certain topic and n is greater than or equal to a threshold value α, we consider the user a potential answerer of unanswered questions in that topic. You are asked to recommend up to 5 most recent unanswered questions to a given user.

- We want to know if the acceptance of an answer would affect its upVote as well as other answers' upVote. We denote the day an answer is accepted as "decision day". You are asked to find out among all questions whoes total number of upVote is greater than or equal to a certain threshold value α :

  - The question whose accepted answer has the highest percentage of upVote received after the "decision day"
  - The question whose other answers as a whole has the highest percentage of upVote received after the "decision day".

[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

- Consider all users involved in a question as co-authors, for a given user, find out this user's top 5 co-authors based on the number of times they appear in the same question either as originator or answerer.

Tasks

Your tasks include:

- Schema Design for each system

  You should provide **two** schema design versions to support both the simple queries and the analytic queries. For each schema version, make sure you utilize features provided by 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 mark if you present a schema that is an exact copy of the relational structure in the original data set.

  You can discard data that are not useful. You can duplicate data if you find that helps with certain queries. You may preprocessing the data by adding fields that are not in the data set. Such preprocessing should not be used to create and save any query result directly in the database. 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.

- Query Design and Implementation

  Load the complete data on both system and set up proper indexes that will be used by the target queries. Design and implement all target queries. You may implement a query using the provided query command (e.g. MongoDB shell or Cypher query) alone, or a combination of JavaScript and shell command in the case of MongoDB or as Python/Java program. In case that a programming language is used, make sure that you do most processing at the server side. You may get significant mark deduction if most of the analysis are implemented with programming language.

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 four sections. The first section is a brief introduction of

the project. Section two and three should cover a system each. Section four should provide a comparison and summary.

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

Section two and three should contain the following two sub sections

- **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, a sample row in each HBase table. 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. 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.

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. The actual marking will be done after the demo using submitted code.

- **Source Code/Script and soft copy of report**
  Submit a zip file through the eLearning site, before 5pm on Tuesday ^{10th} of October, 2017 (week 10). The zip file should contain the following:

  - a soft copy of the report

- query script or program code for each system
- data loading script or program for each system
- a `Readme` document for how to prepare the data and run the queries. The instruction should be detailed enough for the markers to quickly prepare the data and to run the queries. For instance, it is preferable that you prepare a single script for data loading. You should also indicate where and how run-time argument are supplied. If you used special features only available in a particular version or environment, indicate that as well. You may not get mark if the marker is not able to prepare the database and run your code.