Sunny Chiu

CP344

Richard Koo

10/15/17

Q and A (MongoDB VS MySQL)

The Q and A application used in this class required a database to store information for four distinct objects: user, questions, answers, and votes. Both MongoDB and MySQL offered database implementations of Q and A that fulfills all of its requirements. Though MySQL provided safer and sometimes easier queries with its database and syntax, MongoDB delivered a more simplistic utilization of the four distinct objects' relationship than MySQL.

The four objects required for this application is referred to as entities/relationships in relational database management system (RDMS) and were created as tables in MySQL. Each of the entities/relationships' attributes were the columns for their corresponding tables. The user table had the columns: user id, email, and password. The question table had the columns: question id, user id, and texts. The answer table had the columns: answer id, question id, user id, and text. The vote table had the columns: user id, answer id, and tally for each individual vote. MySQL provided integrity constraints with foreign keys in each table. For example, every time a question was deleted, all of its answers were deleted or an answer cannot be created without a valid user id. Foreign keys maintained the data consistency without any effort on the application constricting all dependent entities/relationships to follow their parents' cascade and restrict instructions.

Furthermore, MySQL offered a wide variety of queries that were incredibly intuitive, flexible, and powerful. The queries provided the programmer easy readability and construction of queries with its perceptive key words like SELECT, FROM, WHERE, etc. Most importantly, MySQL queries gave many functionalities that enable the programmer to access and manipulate the information needed for the Q and A application. The programmer can use the JOIN clause to concatenate the answer and vote table together to retrieve information about how many answer have votes and have no votes. Simple but powerful clauses like JOIN is very useful when the customer is asking new features. For example, when asked to add a new feature called rank after creating a simple implementation of Q and A, it takes one query statement to find the rankings of all answers with the JOIN clause by combining information from the answer and vote tables. Any new requested feature that uses the information in the four tables can easily be created with the JOIN and VIEW clause.

On the other hand, MongoDB was very different from MySQL and lacked some features of MySQL. MongoDB is a schema-less, document oriented database meaning all data is store in collections made out of individual documents that can have no relationship with other collections or documents. It does not support foreign keys and therefore all integrity constraints for the Q and A application was relied on the application itself. Every time a question was deleted, the application must make sure the answers for that question were deleted too. Creating these constraints may not perform better than the ones created by MySQL.

In addition, the most noticeable different between MongoDB and MySQL is the query capabilities of aggregate functions, JOIN and VIEW. Almost all aggregate functions were done within MongoDB's pipeline which is a list of aggregate functions that manipulated data in a collection and returned the data as an output. Some aggregate functions were more restrictive than others and required help from other aggregate functions to be more flexible. For example, when counting all the votes for answers, the aggregate function, {$sum : 1}, was used to count how many times the same answer appeared even answers with no votes. Because $sum does not allow a filters, $project was used with conditional statements to filter out all answers with no votes. Compared to MySQL where LEFT JOIN already sets all non-voted answers as none, selecting data in MongoDB involved more abstract uses of its functions. Even $lookup, MongoDB’s closest aggregate function to JOIN, does not combine two collections together, but embeds the foreign collection’s similar field document as a nested document to the local collection’s similar field document. Creating the ranking of users with $lookup will require more functions and lines of code to create a relationship between users, questions, and answers than in MySQL.

The advantage of MongoDB was the utilization of relationship between question, answer, and vote. None of these three objects were sub-types of each other. They were nested within their parent data because they had a HAS-a relationship with their parent data. All votes must be within an answer and all answer must be within a question. When you deleted a question, all data about the answer and votes were automatically deleted. The functionality of delete cascade was fulfilled with this nested documents, but integrity constrains such as checking if a question or user exists before creating an answer must be done manually in the Q and A application.

The most effective use of this nested relationship was avoiding joins to other collection and decreasing run-time since all the information required was already embedded in to itself. This can be seen in the ranking of questions and answers. The number of answers for each question can be counted in the nested answer document in each question document. Similarly, the number of votes for each answer can be counted in the nest vote document in each answer document. Without nesting question, answer, and vote together, each ranking implementation would require a $lookup with another collection. Performing a $lookup is much more experience than not performing one. The only $lookup that should be used is for the ranking of user, the only collection outside of the question collection, which less than the two or three joins required in the MySQL Q and A application.

For this Q and A application, MongoDB's integrity constraints and consistency were worse than MySQL's. MongoDB's queries maybe more complex than MySQL's with its limited flexibility in query options. However, the intergraded relationship of the nested documents of question, answer and vote in MongoDB allowed better performance by using less $lookup between collections. This Q and A application can be achieved through the simple relations in MongoDB without any too complex aggregate function and query data.