Question-Answering System

# Database Design Document

Version 0.1

02/28/2019

Table of Contents

[1. Overview iii](#_Toc3817067)

[2. Assumptions/Constraints/Risks iv](#_Toc3817068)

[3. Design Decisions v](#_Toc3817069)

[4. Detailed Database Design vii](#_Toc3817070)

[1. x](#_Toc3817071)

[4.1 Roles and Responsibilities x](#_Toc3817072)

[4.2 Performance Monitoring and Database Efficiency x](#_Toc3817073)

[4.2.1 Operational Implications x](#_Toc3817074)

[4.2.2 Data Transfer Requirements x](#_Toc3817075)

[4.2.3 Data Formats x](#_Toc3817076)

[Appendix A: Acronyms xi](#_Toc3817077)

No table of figures entries found.

List of Tables

[Table 1 - Acronyms 9](#Table!1|sequence)

## Overview

In a Question-Answering problem, a question needs to be stored with its respective answers. When a new question arrives, ideally, we want to check if a similar question was already asked before. If so, the question will not be stored as a new one. Instead, the answers for the similar question will be output to the user. This is an NLP (Natural Language Processing) solution for avoiding storing duplicate questions in question-and-answer sites such Quora and Stack Overflow. Nevertheless, even using this NLP solution, the set of questions/answers might become very big overtime. Therefore, a Big Data architecture would be the most suitable way of dealing with the increasing amount of data to be stored and accessed in this type of system.

From a question-and-answer website, when a user asks a specific question, this question is not immediately answered. In Quora’s system for example, each question is reviewed and assigned to another user that can answer it according with his/her interests/knowledge. The assigned user can also re-assign the question to someone else that would be better suited to answer that question.

The goal of this project is to model the question-and-answer pairs in a dedicated database considering that it will be interacting with a second database that models the users (who asked/answered questions), as well as their preferences, actions and relationships; like a social media system.

## Assumptions/Constraints/Risks

In this project, we assume that target customer/company has (or has the means of obtaining) all the hardware and software resources necessary to the development of a free consumer internet product in which 10s of millions of users can connect.

Regarding the question-and-answer data to be stored, which is the main scope of this project, we need to foresee a scalable solution that expects to handle more than 10,000 questions per day. This scalable storage solution is defined in the next section.

## Design Decisions

Instructions: Describe how performance and availability requirements will be met. Examples include:

* Describe design decisions on database distribution (such as client/server), master database file updates and maintenance, including maintaining consistency, establishing/ reestablishing and maintaining synchronization, enforcing integrity and business rules.
* Describe design decisions to address concurrence issues (e.g., how the data are partitioned or distributed to support multiple applications or competing update functions, if applicable).
* Describe design decisions to support Service Level Agreements (SLAs) for key functions supported by the database.
* Describe design decisions on backup and restoration including data and process distribution strategies, permissible actions during backup and restoration, and special considerations for new or non-standard technologies such as video and sound. Describe the impact this maintenance will have on availability.
* Describe design decisions on data reorganization (i.e., repacking, sorting, table and index maintenance), synchronization, and consistency, including automated disk management and space reclamation considerations, optimizing strategies and considerations, storage and size considerations (e.g., future expansion), and population of the database and capture of legacy data. Describe the impact this maintenance will have on availability.
* Describe design decisions to support purging and/or archiving of data to ensure performance and storage objectives are met. Describe the impact this maintenance will have on availability. Describe any needs to recall archived data back into the database.

Regarding the database choice, different options have been considered for this project.

A traditional Relational DB is an interesting option that allows us to model the entities-relationships of our system in the most natural way: questions *versus* answers (see the Conceptual and Logical models in the next section). However, we can run into scalability issues that are better dealt with some noSQL solutions. The later can expand easily to take advantage of a distributed system (“scale out”).

Among the noSQL DBs, Cassandra is a good option that has been considered (see <https://github.com/luana-be/CEB1250_repo/blob/master/QUESTION-ANSWERING.cql>). For the NLP subproblem of returning questions that are close to the asked question, the column “question” would be the only one employed for querying. Cassandra is well suited for this type of problem, since all questions would be stored in the same place and no other columns would be employed in a query – of course, the corresponding “answer” would be output in the end. However, we need to be sure that the system will always work this way, otherwise we could encounter some performance issues in the future.

Therefore, we opted for a document-database, MongoDB, in which we model each question-and-answer pair as a specific entry in a collection. A known disadvantage of MongoDB is that it does not allow partial matches during a query, an aspect that we cannot accept in our system -- since our final goal is to implement an NLP solution for avoiding question duplicates. A work around for this matching problem, well reported in the literature, is to use ElasticSearch as a query layer.

By using Mongo connector

Syncronizing MongoDB and ElasticSearch

Picture!!

## Detailed Database Design

Instructions: Describe the design of all DBMS structure associated with the system. The headings and sub-headings in this section should be structured according to the information to be presented, and may include discussions about or references to the following:

* Conceptual Data Model (CDM)
* Logical Data Model (LDM) and LDM Entity Relationship Diagram (ERD).
* Physical Data Model (PDM) with a description of the DBMS schemas, sub-schemas, records, sets, tables.
* A comprehensive Data Dictionary showing data stores, data element name, type, length, source, constraints, validation rules, maintenance (create, read, update, delete (CRUD) capability), audit and data masking requirements, expected data volumes, life expectancy of the data, information life-cycle management strategy or at least an archiving strategy, outputs, aliases, and description.
* Planned implementation factors (e.g., distribution and synchronization) that impact the design.
* Estimate of the DBMS file size or volume of data per entity.
* Definition of the update frequency of the database tables, views, files, areas, records, sets, and data pages. Also provide an estimate of the number of transactions, if the database is an online transaction-based system.

The detailed database design information can be included as an appendix, such as DDLs, which would be referenced here.

Regarding the database definition, different options have been considered. For instance, a Key-value-based NoSQL would be a good choice if the system allowed only one answer per question. However, a same question can have more than one answer. In a document-based NoSQL (or in a simple file-based system), a question with its respective answers could be grouped in a same data structure (e.g., a json file), under a same key. However, adding new answers to a question (as well as retrieving answers from a question) wouldn't be trivial, since the answers need to be ranked by using a voting scheme. Therefore, a traditional Relational DB (MySQL) has been chosen for our proposed Question-Answering problem.

As for the data properties, despite the choice of using a Relational Data Base, this system can behave well with a BASE Consistency Model. In the following, we define our conceptual (Figure 1) and logical models (Figure 2).

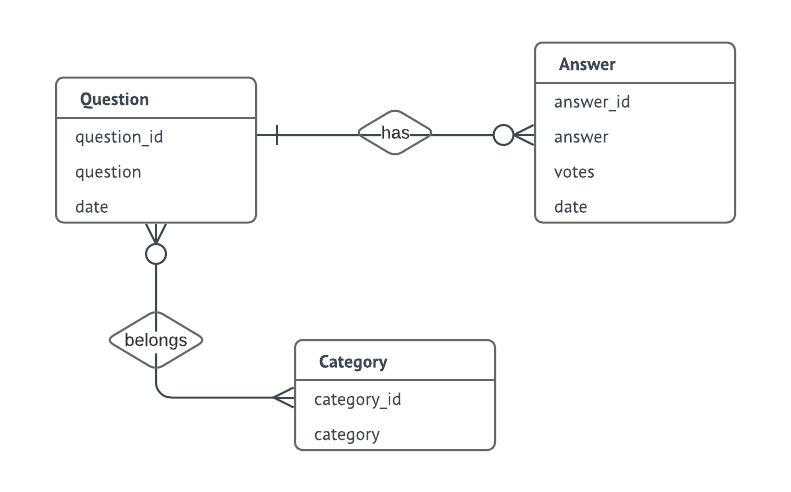


Figure 1 – Conceptual Model

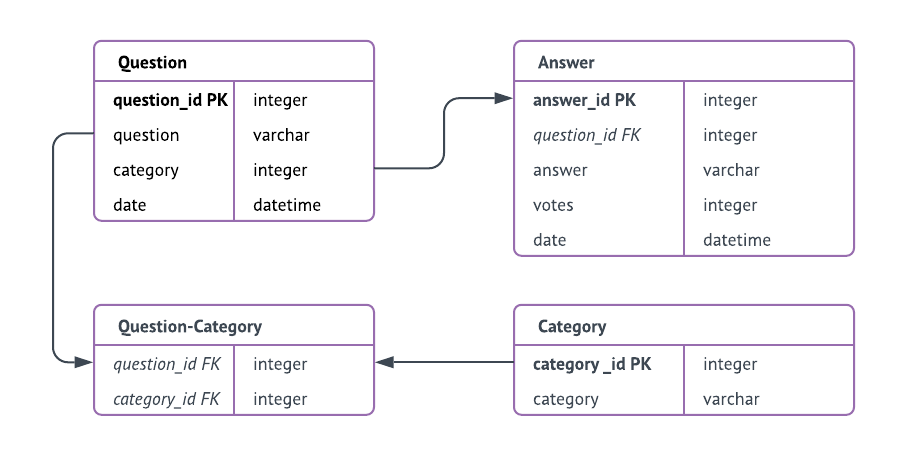


Figure 2 – Logical Model, where all attributes are mandatory (not null)

As for the DDLs, please refer to the SQL script:

<https://github.com/luana-be/CEB1250_repo/blob/master/QUESTION-ANSWERING.sql>

## 

### Roles and Responsibilities

Instructions: Identify the organizations and personnel responsible for the following database administrative functions: database administrator, system administrator, and security administrator. Describe specific administration skill requirements applicable to the database.

### Performance Monitoring and Database Efficiency

#### Operational Implications

Instructions: Describe operational implications of data transfer, refresh and update scenarios and expected windows.

#### Data Transfer Requirements

Instructions: Describe data transfer requirements to and from the software, including data content, format, sequence, volume/frequency and any conversion issues.

#### Data Formats

Instructions: Describe formats of data for both the sending and receiving systems, including the data item names, codes, or abbreviations that are to be interchanged, as well as any units of measure/conversion issues.

Appendix A: Acronyms

Instructions: Provide a list of acronyms and associated literal translations used within the document. List the acronyms in alphabetical order using a tabular format as depicted below.

Table 1 - Acronyms

| Acronym | Literal Translation |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |