Information Management and Systems Engineering
SS2019

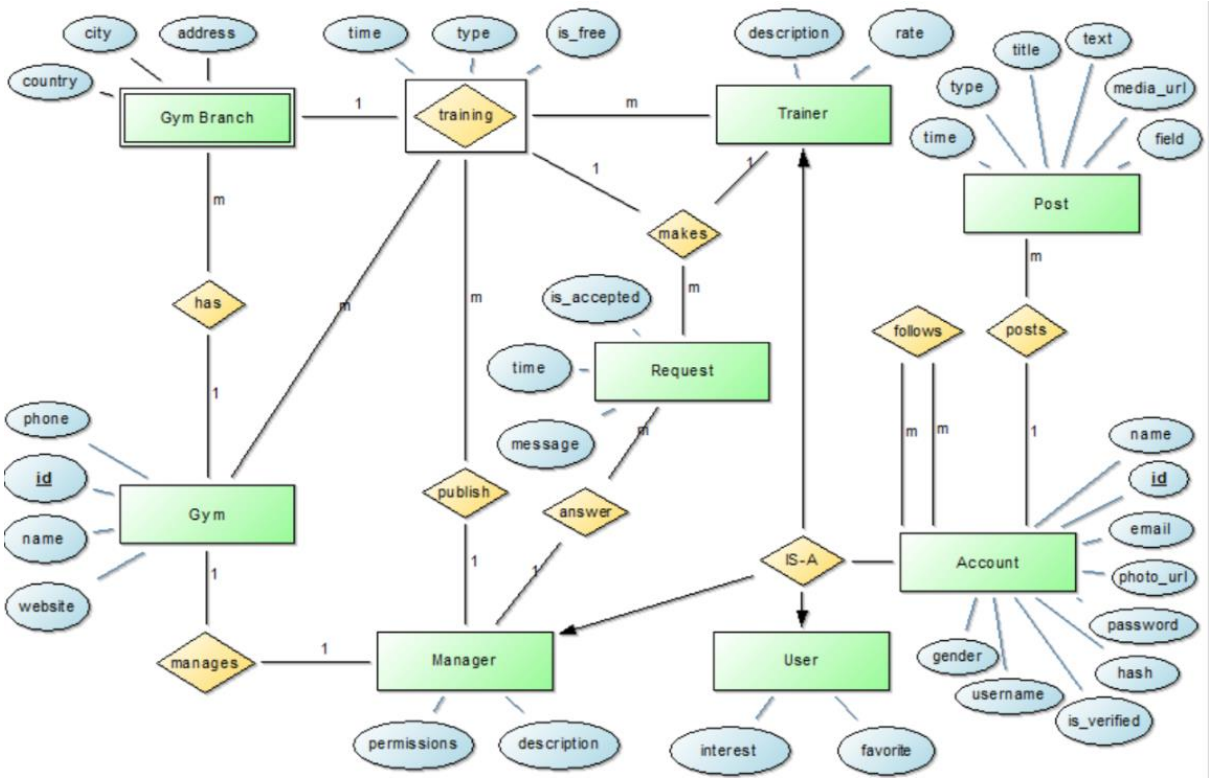
Nikodijevic Mihajlo
01646292

Milestone 3
Specification of IS

Fitman®

3.1. NoSQL Database Design

Structure SQL:



Motivation:

Adapting this schema for MongoDB was not easy task for me, even against all my previous experience with all NoSQL databases like Redis, CouchDB etc.

This course was very nice experience for me specially MongoDB part and Docker containers. I also played with frontend framework Vue.js which is amazing! Finally, something interesting at University!

And to be honest, I am very sorry that I didn't implement all use cases for this time slots, it took too much time anyway to learn Node.js, Vue.js, Docker, Mongo, Mongoose etc. But amazing experience and worth it!

Listing trainings and adding new training are working in MongoDB as in SQL for the Milestone2. Additional backend functionality for other use cases like "Login" is present but not integrated with the frontend yet..

Discussion:

Nesting model approach is good if we don't change data too often. For example, Account is stored in accounts collection, but also in Training, to know which trainer and manager are responsible for that one. Updating some information for trainer, like username, must be done in all trainings too in my example, meaning that this operation will be difficult and very slow. But however, that action happens very rare and it's okay because this nesting structure for Training model for example results in insane fast query for trainings.

That one is the most important use case in our system, which creates Calendar View.

Alternative design would be reference to trainer and manager, what will result in slower queries (performance issue with joining data in application layer) because I would need to fire 2 additional queries, 1 for trainer and 1 for manager.

Nested Models in MongoDB: (mongo folder)

Account Model: Stores other accounts in "following" and "followers" properties, but also in the "GymBranch" place where this manager works.

Gym Model: Stores all branches in "branches" property.

Post Model: Stores account(publisher) in "accounts" property.

Request Model: Stores trainer and manager as nested properties, not foreign keys anymore.

Training Model: Stores gym branch(place), trainer and manager as nested properties, not foreign keys anymore.

Models which doesn't exists standalone anymore:

Follow – In SQL Structure this one held ids from 2 Accounts, representing "follow" relationship between them. Now, in MongoDB they are part of the Account model which stores array of other Accounts as "following" and "followers".

Gym Branch – In SQL Structure this one held id from Gym, and it was representing Training place (Foreign Key in SQL approach), but now in MongoDB, gym branch is only "abstract" part of Gym and Training in "place" property.

Indexing and Sharding:

All “_id” fields in my MongoDB are custom type: Number, because my implementation of primary keys in SQL structure was unsigned integer and that was easier for migration than built-in ObjectId data type.
Also, all _id fields are indexes in my implementation.
I also used _id as shard key in most of my models because it was very hard for me too choose good shard key. It’s also bad that my _id is autoincremented as I had in my MySQL database, I know it is better to have hashed _id as shard key.

3.2. Data Migration

On the right side, in sidebar, there is orange button “Migrate data!”. By clicking on this button, request/migrate endpoint is triggered via POST request and run “migrate” function in request controller.
Data is automatically inserted in our SQL database already from Milestone 2.
This data is queried from MySQL database in this function and inserted accordingly into MongoDB.

Conclusion:

Docker container has the same problem as you mentioned in CeWebs. When running “docker-compose up” it doesn’t wait for MySQL to install completely and starts app before that, so you will need to run it 2 times.
Opening localhost:3000 should serve frontend application.
In case of any questions, it would be nice to have presentation talk at the end to clarify all things and also discuss the problems and optimization possibilities!