Milestone 3

Vitalii Romanov. Matr.nr: 01549702

June 2019

Contents

1 How to run the migration and the app generally

1. To run this app just use the command ”docker-compose -f stack.yml up”.
2. Wait, until the data is inserted.
3. After go to the http://localhost:4001 to use the app.
4. On the first page (Home) click on the button ”Migrate data”.
5. Use Social Networks MongoDB and Developers MongoDB to check 2 main use cases.

2 NoSQL design

In this section we will discuss the NoSQL design while migration from RDBMS to NoSQL DBMS. First of all, on the diagram bellow you can see the Entity-Relationship model of original RDBMS:

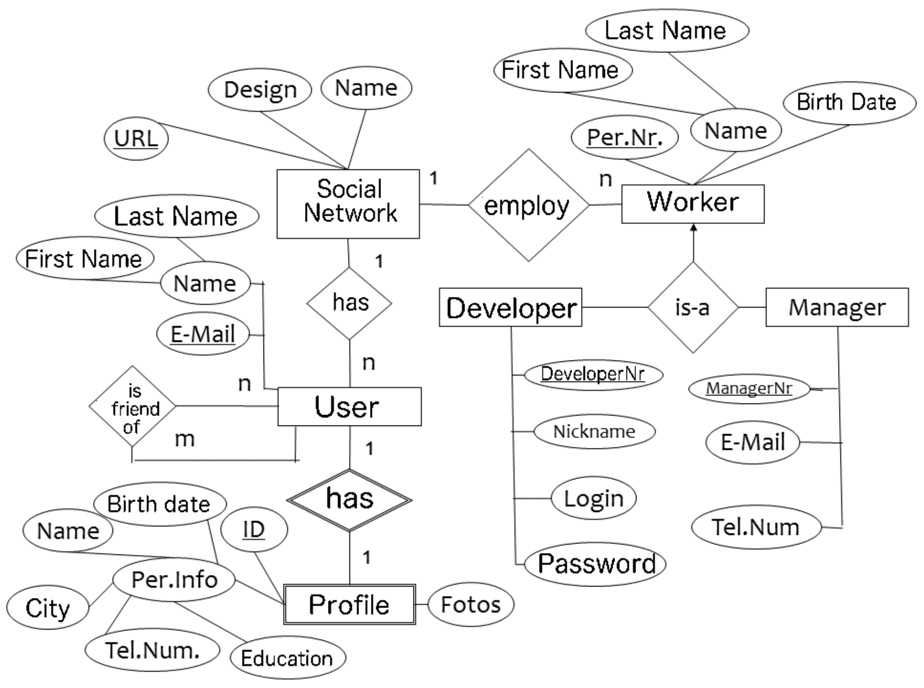
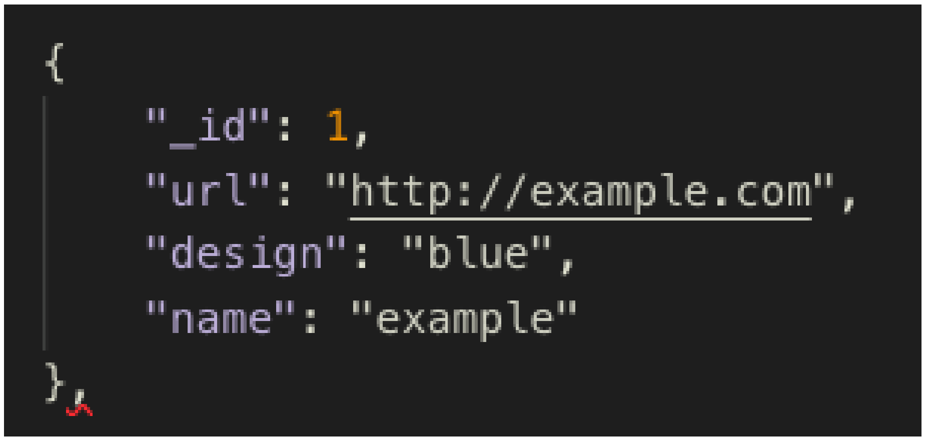


Figure 1: ER diagram

Social network is stand alone collection (see bellow):

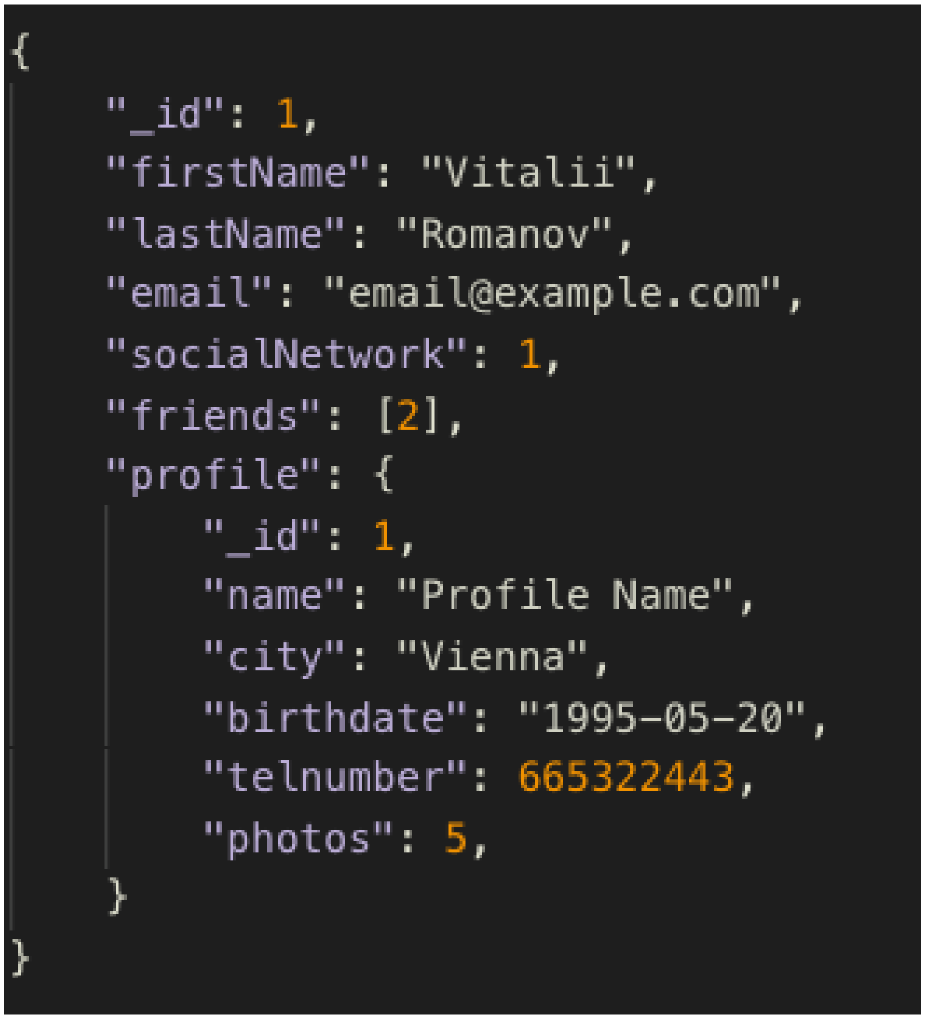


```
{
    "_id": 1,
    "url": "http://example.com",
    "design": "blue",
    "name": "example"
},
```

Figure 2: Social Network Json

Theoretically we could embed users and workers inside of social network. At the same time, it was chosen to be stand alone collection, because the information about social network will be not changed very often, that is why this information should be encapsulated.

Because the ”User” and the ”Profile” has relationships 1 to 1 and ”Profile” is weak entity, so it’s logical to embed ”Profile” in the ”User”, so we have the following design:

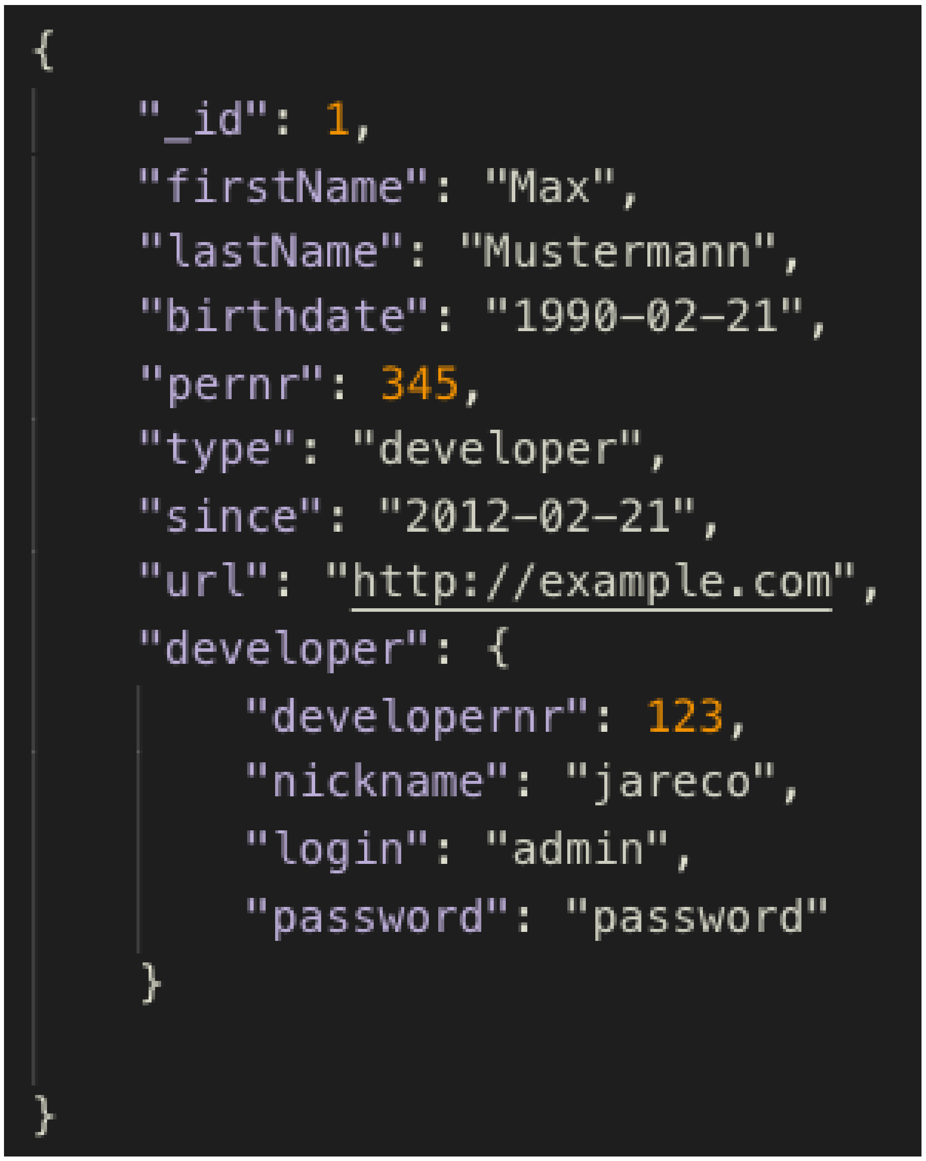


```
{
    "_id": 1,
    "firstName": "Vitalii",
    "lastName": "Romanov",
    "email": "email@example.com",
    "socialNetwork": 1,
    "friends": [2],
    "profile": {
        "_id": 1,
        "name": "Profile Name",
        "city": "Vienna",
        "birthdate": "1995-05-20",
        "telnumber": 665322443,
        "photos": 5,
    }
}
```

Figure 3: User + Profile

Social network is referenced using variable ”socialNetwork”, which contains the id of social network. Because we have n:m relationships between ”Users”, so we have the variable ”friends” which contains id of friends.

Because ”Developer” and ”Worker” has ”is-a” relationship, it’s logical to embed ”Developer” in the ”Worker”.



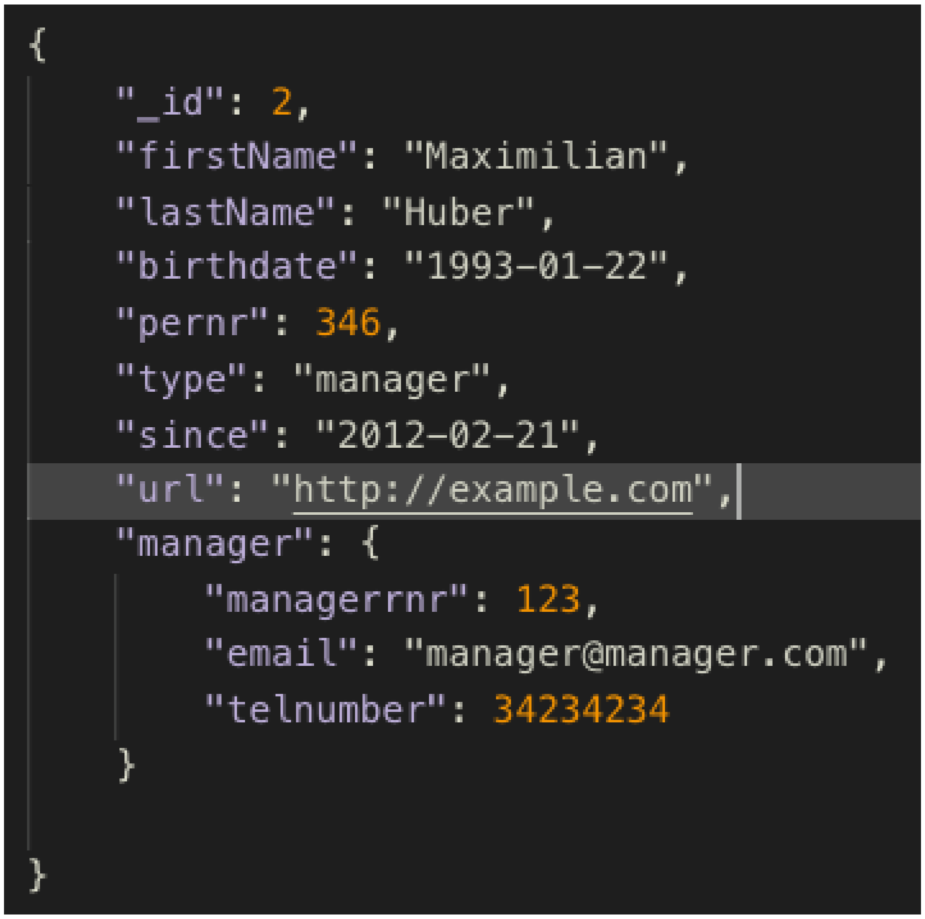
```json
{
    "_id": 1,
    "firstName": "Max",
    "lastName": "Mustermann",
    "birthdate": "1990-02-21",
    "pernr": 345,
    "type": "developer",
    "since": "2012-02-21",
    "url": "http://example.com",
    "developer": {
        "developernr": 123,
        "nickname": "jareco",
        "login": "admin",
        "password": "password"
    }

}
```

Figure 4: Worker + Developer

For better recognition between ”Developer” and ”Manager” is used the variable ”type”, which can have two values: ”developer” and ”manager”.

The ”Worker” can be not only ”Developer”, but also ”Manager”, so we have the similar structure as above:



```
{
    "_id": 2,
    "firstName": "Maximilian",
    "lastName": "Huber",
    "birthdate": "1993-01-22",
    "pernr": 346,
    "type": "manager",
    "since": "2012-02-21",
    "url": "http://example.com",
    "manager": {
        "managerrnr": 123,
        "email": "manager@manager.com",
        "telnumber": 34234234
    }

}
```

Figure 5: Worker + Manager

In both situation (”Developer” and ”Worker”) was used ”url” variable as a reference to ”socialNetwork” collection.

3 Alternative design choices

As it was already mentioned, alternatively we can embed in e.g. users and workers in social network, because it’s typical solution by 1:n relationships.

In all other situations the solution was made according to the typical solutions in the official tutorial of MongoDB developers for migration from RDBMS to NoSQL DBMS.

4 Indexing and discussion on optimized sharding of NoSQL DBMS for the IS

RDBMS use very often joins for getting needed information. If we have a lot of big tables, this operation can be very costly. More flexible NoSQL design helps us to construct the structure of data storing in the way, so we need less complicated operations to get needed data.

Sharding generally speaking means that we store the rows of the table in different places. Using this approach we reduce the time of indexing in each table, because we have less rows. At the same time we have problem with consistency. If one table must be changed, it must be changed in all places, where it's stored. We have also problem with latency, if one more than one shard should be accessed. Also the connection between servers can produce some problems.