**Act5.1 CRUD in MongoDB**

*/\* Introduction*

*In this activity we will design and development the database of a social network without implementing the social network*

*itself. This approach does not try to be complete or totally correct, but it does try to reflect how NoSQL databases are*

*able to solve real problems of current software.*

*The goal of these exercises to test the commands in MongoDB so that you can check if your solution works.*

*\*/*

use social-network

*/\**

*During login users of our social network tell us their name, surname, age and gender. We also insert the registration*

*date at the time of registration. As identifier we want to use the Long type (use NumberLong()), not the ObjectID that*

*Mongo gives us by default. Let's say that two users are registered. Write the statements to insert them into the "user"*

*collection, with identifiers 5 and 6 respectively.*

*- Juan García Castellano, 23 years old*

*- Beatriz Perez Solaz, 27 years old.*

*\*/*

db.createCollection('user')

db.user.insertMany([

{\_id: new NumberLong(5), name: 'Juan', surname: 'García Castellano', age: 23, gender: 'M', registration: new Date},

{\_id: new NumberLong(6), name: 'Beatriz', surname: 'Perez Solaz', age: 27, gender: 'F', registration: new Date},

])

*/\**

*Write the command to retrieve all documents in the "user" collection.*

*\*/*

db.user.find()

*/\**

*Users can belong to as many groups as they want, and we save those groups in an Array field called groups in each user.*

*Insert the new user Jorge Lopez Sevilla, with identifier 7, who does not tell us his age, and who belongs to the groups*

*"basketball", "kitchen" and "historical novel".*

*\*/*

db.user.insertOne({\_id: new NumberLong(7), name: 'Jorge', surname: 'Lopez Sevilla', gender: 'M', registration: new Date, groups: ['basketball', 'kitchen', 'historical novel']})

*/\**

*Juan García, with identifier 5 is unsubscribed. Write the sentence to delete it.*

*\*/*

db.user.deleteOne({\_id: 5})

*/\**

*The user Beatriz, with identifier 6, signs up for two groups: "historical novel" and "dance". Write the statement to*

*update these fields in your document without re-reporting the rest. Remember that groups are saved as arrays.*

*\*/*

db.user.updateOne({\_id: 6}, {$set: {groups: ['historical novel', 'dance']}})

*/\**

*In our social network you can also register companies, which we keep in the collection "company", also with Long*

*identifier, and for which at the moment we only store the name of the company. Write the command to insert the company*

*"Gardening Gardenia" with identifier 10.*

*\*/*

db.createCollection('company')

db.company.insertOne({\_id: new NumberLong(10), name: 'Gardening Gardenia'})

*/\**

*Now you must update the data of the company "Gardening Gardenia" adding the following fields:*

*- Address. It must be an embedded document with street Palmeras, number 8 and town Torrente.*

*- Sector: services.*

*- Web: http: //www.gardeninggardenia.com*

*\*/*

db.company.updateOne(

{name: 'Gardening Gardenia'},

{$set: {address: {street: 'Palmeras', number: 8, town: 'Torrente'}, sector: 'services', web: 'http: //www.gardeninggardenia.com'}}

)

*/\**

*We are going to count the followers of the companies of the social network, using a field "followers" in the collection*

*"company". Five users have marked FOLLOW the company "Gardening Gardenia". Type the command to create the "followers"*

*field being equal to 5.*

*Then two people have followed the company. Type the command to increment it.*

*Finally, one has unfollowed. Also type the command to decrease it.*

*\*/*

db.company.updateOne({name: 'Gardening Gardenia'}, {$set: {followers: 5}})

db.company.updateOne({name: 'Gardening Gardenia'}, {$inc: {followers: 2}})

db.company.updateOne({name: 'Gardening Gardenia'}, {$inc: {followers: -1}})

*/\**

*Update the address of the company "Gardening Gardenia", add the postal code 46009.*

*\*/*

db.company.updateOne({name: 'Gardening Gardenia'}, {$set: {'address.postal': 46009}})

*/\**

*Eliminate the field "sector" of the company "Gardening Gardenia", leaving intact the rest of the fields.*

*\*/*

db.company.updateOne({name: 'Gardening Gardenia'}, {$unset: {sector: ''}})

*/\**

*The user Beatriz, with identifier 6, signs up for the group "theater". Write the command to add that group to its array.*

*Docs: https: //www.mongodb.com/docs/manual/reference/operator/update/addToSet/*

*\*/*

db.user.updateOne({\_id: 6}, {$addToSet: {groups: 'theater'}})

*/\**

*The user with identifier 6 is deleted from the "dance" group. Write the statement to remove it from its group array.*

*\*/*

db.user.updateOne({\_id: 6}, {$pull: {groups: 'dance'}})

*/\**

*As in any social network, users can enter comments. In our case the comments have several fields, which are.*

*- Title*

*- Text*

*- The group to which the comment belongs.*

*- Date*

*Comments will be saved in the "user" collection itself, in a new "comments" field, which will be an Array of objects*

*with the previous properties.*

*In addition, at the user level, we will also keep a counter of the number of total comments made by each user, in a*

*field "total\_comments", which we will increase each time we insert anew comment.*

*Write the command to insert a new comment for the user Jorge Lopez Sevilla in the group "historical novel", while*

*increasing the "total\_comments" by one.*

*Afterwards, write the command to insert a new comment for the user Jorge Lopez Sevilla in the group "basketball", while*

*increasing the "total\_comments" by one.*

*\*/*

db.user.updateOne(

{name: 'Jorge', surname: 'Lopez Sevilla'},

{$set: {comments: [{title: 'New historical novel comment title', text: 'New historical novel comment text', group: 'historical novel', date: new Date}], total\_comments: 1}}

)

db.user.updateOne(

{name: 'Jorge', surname: 'Lopez Sevilla'},

{

$push: {comments: {title: 'New basketball comment title', text: 'New basketball comment text', group: 'basketball', date: new Date}},

$inc: {total\_comments: 1}

}

)

*/\**

*Write the commands to retrieve the following information:*

*- Name, surname, age and \_id of those users that belong to “historical novel” group and are older than 25 years old.*

*- Name, surname and groups (but not the \_id) of those users belonging to at least 2 groups.*

*- Name, surname and groups (but not the \_id) of those users belonging to “historical novel” group and “theater”.*

*- Name, surname of those users that have a field named comments.*

*- Name of the companies that are in Torrente and have zero followers.*

*- Name of the companies that are in Torrente and have more than 5 followers.*

*\*/*

db.user.find(

{groups: 'historical novel', age: {$gt: 25}},

{name: 1, surname: 1, age: 1, \_id: 1}

)

db.user.find(

{$where: "this.groups.length >= 2"},

{\_id: 0, name: 1, surname: 1, groups: 1}

)

db.user.find(

{groups: 'historical novel', groups: 'theater'},

{\_id: 0, name: 1, surname: 1, groups: 1}

)

db.user.find(

{comments: {$exists: true}},

{name: 1, surname: 1}

)

db.company.find(

{'address.town': 'Torrente', followers: 0},

{name: 1}

)

db.company.find(

{'address.town': 'Torrente', followers: {$gt: 5}},

{name: 1}

)