Islamic University of
Technology

Lab-7 Report: NoSQL Scenario

Submitted by:

Name : Mashrur Ahsan

ID : 200042115

Program : SWE

Department : CSE

Course Code : CSE 4410

The Scenario:

Think of a scenario of social media platform that allows users to create profiles, post, follow other users, and like posts. Generally, the user provides his basic information for example name, email, password (here you don't have to think of hashing password), phone no, date of birth, address, and profile creation date. Other than that users may add their working status (like what is he/she doing currently, and how long they are doing this), bios, hobbies, and so on. Any user can post content and anybody can like and comment on that post. Additionally, a user can follow others.

Tasks:

1. Create necessary collections with respective attributes.

```
/*  1  */

/*  user collection  */
{
  _id (ObjectId)
  name (string)
  email (string)
  password (string)
  phone_number (string)
  date_of_birth (date)
  address (string)
  profile_creation_date (date)
  working_status (string)
  bios (string)
  hobbies (array)
  following (array)
  followers (array)
}

/*  post collection  */
{
  _id (ObjectId)
  user_id (ObjectId)
  content (string)
  likes (array)
  comments (array)
  created_at (date)
}
```

2. Insertion

(a) An entry of a user with only email, name and password.

```
/*  2(a)  */
db.user.insertOne({
  "name": "DAM",
  "email": "DAM@example.com",
  "password": "12345678",
})
```

(b) Multiple entries of users with basic info and hobbies.

(c) An entry of a user with basic info and working status.

```
/*  2(b)  &  2(c)  */
db.user.insertOne({
  "name": "MAK Shelby",
  "email": "mak_shelby@example.com",
  "password": "12345678",
  "phone_number": "01838292577",
  "date_of_birth": new Date("2002-04-29"),
  "address": "Mirpur, Dhaka, Bangladesh",
  "profile_creation_date": new Date(),
  "working_status": "Software Developer",
  "bios": "Gooner",
  "hobbies": ["Music", "Movies", "Gaming", "Sleeping"],
  "following": [],
  "followers": []
})
```

(d) Add multiple followers for multiple users.

```
/*  2(d)  */
db.user.updateOne(
  { "_id": ObjectId("64100b58d87392106fab76f5") },  // provide the actual ObjectIds
  { $set: { "following":  ObjectId("64100a89d87392106fab76f4") } } // provide the actual ObjectIds of the followers
)
```

```
db.user.updateOne(
  { "_id": ObjectId("64157ee38af25edbd00307b0") },  // provide the actual ObjectIds
  { $set: { "following":  [ ObjectId("64100a89d87392106fab76f4"), ObjectId("64157d5d8af25edbd00307af") ]  } }
  // provide the actual ObjectIds of the followers
)
```

(e) Multiple posts with content, creation time, like and the user id who likes.

```
/*  2(e)  */
db.post.insertOne({
  "user_id": ObjectId("64100a89d87392106fab76f4"), // provide the actual ObjectId of the user who created the post
  "content": "This has been MAK!",
  "likes": 134510,
  "comments": ["Noice", "Luv it!"],
  "created_at": new Date()
})

db.post.insertMany([

  {
    "user_id": ObjectId("64100a89d87392106fab76f4"),  // provide the actual ObjectId of the user who created the post
    "content": "This has been MAK!",
    "likes": ["64157d5d8af25edbd00307af", "64157ee38af25edbd00307b0"],
    "comments": ["Noice", "Luv it!"],
    "created_at": new Date()
  },
  {
    "user_id": ObjectId("64157d5d8af25edbd00307af"),  // provide the actual ObjectId of the user who created the post
    "content": "Yo!",
    "likes": ["64100a89d87392106fab76f4"],
    "comments": ["Noice", "Luv it!"],
    "created_at": new Date()
  }

]);
```

(f) Add multiple comments for multiple posts.

```
/*  2(f)  */
db.post.updateMany(                          //  deleting all of the elements of an array
  {  },                                     //   here, we are modifying each element
  { $set: { comments: [] } }
)


db.post.updateMany(         // match documents with user_ids in the specified list (it could've had more user_ids)
  { user_id: { $in: [ObjectId("64100a89d87392106fab76f4")] } },
  { $push: { comments: { $each: [
      { text: "Noice!", user_id: ObjectId("64157d5d8af25edbd00307af") },
      { text: "Luv it!", user_id: ObjectId("64157ee38af25edbd00307b0") }
    ] } }
  }
)
```

3. Data Retrieving

(a) Display the most recent to oldest posts along with their poster.

```
/*  3(a)  */

db.post.aggregate([{$sort:{createdAt: -1}}])     //  Descending order: "-1"

db.post.find().sort({created_at: -1})
```

Snippet of the result:

```
test> use lab7
switched to db lab7
lab7> db.post.aggregate([{$sort:{created_at: -1}}])
[
  {
    _id: ObjectId("641583a18af25edbd00307b1"),
    user_id: ObjectId("64100a89d87392106fab76f4"),
    content: 'This has been MAK!',
    likes: [ '64157d5d8af25edbd00307af', '64157ee38af25edbd00307b0' ],
    comments: [
      { text: 'Noice!', user_id: ObjectId("64157d5d8af25edbd00307af") },
      {
        text: 'Luv it!',
        user_id: ObjectId("64157ee38af25edbd00307b0")
      }
    ],
    created_at: ISODate("2023-03-18T09:25:53.317Z")
  },
  {
    _id: ObjectId("641583a18af25edbd00307b2"),
    user_id: ObjectId("64157d5d8af25edbd00307af"),
    content: 'Yo!',
    likes: [ '64100a89d87392106fab76f4' ],
    comments: [],
    created_at: ISODate("2023-03-18T09:25:53.317Z")
  }
]
lab7>
```

(b) Show all the posts that were created in the last 24 hours.

```
/*  3(b)  */

db.post.aggregate([
  {
    $match: {
      created_at: {                                  // $gt = greater than
        $gt: new Date(Date.now() - 24*60*60*1000)       //  in milliseconds,
      }
    }
  },
  {
    $sort: {
      created_at: -1     // Descending order
    }
  }
])

db.post.find({created_at: {$gt: new Date(Date.now() - 24*60*60*1000)}})     // in milliseconds

// new Date(Date.now()-24*60*60*1000)}}) is used to calculate the timestamp of 24 hours ago from the current date,
// which is then used as the lower bound for the query to find all posts created within the last 24 hours.
```

```
lab7> db.post.find({created_at: {$gt: new Date(Date.now() - 24*60*60*1000)}})
[
  {
    _id: ObjectId("641583a18af25edbd00307b1"),
    user_id: ObjectId("64100a89d87392106fab76f4"),
    content: 'This has been MAK!',
    likes: [ '64157d5d8af25edbd00307af', '64157ee38af25edbd00307b0' ],
    comments: [
      { text: 'Noice!', user_id: ObjectId("64157d5d8af25edbd00307af") },
      {
        text: 'Luv it!',
        user_id: ObjectId("64157ee38af25edbd00307b0")
      }
    ],
    created_at: ISODate("2023-03-18T09:25:53.317Z")
  },
  {
    _id: ObjectId("641583a18af25edbd00307b2"),
    user_id: ObjectId("64157d5d8af25edbd00307af"),
    content: 'Yo!',
    likes: [ '64100a89d87392106fab76f4' ],
    comments: [],
    created_at: ISODate("2023-03-18T09:25:53.317Z")
  }
]
lab7>
```

(c) Show all the users who have more than 3 followers.

There wasn't any user who had more than 3 followers. But, to show that the code works I'm showing users that have more than 1 followers. (In the actual solution; the number "3" would be there instead of "1")

```
/*  3(c)  */

db.user.find({ $expr: { $gt: [ {$size: "$followers"}, 1 ] } })
// "$expr" is used to compare the values of the size and the provided value

db.user.aggregate([
  {
    $project: {                          // defining what to project
      _id: 1,
      name: 1,
      followersCount: { $size: "$followers" }   // getting the size
    }
  },
  {
    $match: {
      followersCount: { $gt: 1 }      // more than one followers
    }
  },
  {
    $project: {            // "1" means true
      _id: 1,
      name: 1,
      followersCount: 1
    }
  }
])
```

```
lab7> db.user.find({ $expr: { $gt: [ {$size: "$followers"}, 1 ] } })
[
  {
    _id: ObjectId("64157d5d8af25edbd00307af"),
    name: 'DAM',
    email: 'DAM@example.com',
    password: '12345678',
    following: [],
    followers: [ '64157ee38af25edbd00307b0', '64100a89d87392106fab76f4' ]
  }
]
lab7>
```

(d) Show all the users who are following more than 3 users.

```
/*  3(d)  */

db.user.find({ $expr: { $gt: [ {$size: "$following"}, 1 ] } })
// "$expr" is used to compare the values of the size and the provided value

db.user.aggregate([
  {
    $project: {                          // defining what to project
      _id: 1,
      name: 1,
      followingCount: { $size: "$following" }   // getting the size
    }
  },
  {
    $match: {
      followingCount: { $gt: 1 }      // more than one followings
    }
  },
  {
    $project: {            // "1" means true
      _id: 1,
      name: 1,
      followingCount: 1
    }
  }
])
```

```
lab7> db.user.find({ $expr: { $gt: [ {$size: "$following"}, 1 ] } })
[
  {
    _id: ObjectId("64157ee38af25edbd00307b0"),
    name: 'MAK Shelby',
    email: 'mak_shelby@example.com',
    password: '12345678',
    phone_number: '01838292577',
    date_of_birth: ISODate("2002-04-29T00:00:00.000Z"),
    address: 'Mirpur, Dhaka, Bangladesh',
    profile_creation_date: ISODate("2023-03-18T09:05:39.592Z"),
    working_status: 'Software Developer',
    bios: 'Gooner',
    hobbies: [ 'Music', 'Movies', 'Gaming', 'Sleeping' ],
    following: [
      ObjectId("64100a89d87392106fab76f4"),
      ObjectId("64157d5d8af25edbd00307af")
    ],
    followers: []
  }
]
lab7>
```