MongoDB

Clone Recitation 4 Repo

https://github.com/61040-fa24/rec-4-exercise

Walk through

Prep solution

Motivation

- How can we create a robust web application?
  - Problems: server shutdowns/crashes, user clearing of browser cache, etc.
- How can we store data persistently?
- Other considerations for data storage:
  - Modifiability: create/modify/delete data concurrently
  - Extensibility: easily redefine data attributes for organization
  - Searchability: quickly find and deliver data in a usable format

Document-Oriented Databases

- Subset of NoSQL databases
- Stores all information for a given object in a single **document** in the database
- What is a document?
  - A data structure composed of field and value pairs
  - MongoDB documents are similar to JSON objects
    - The values of fields may include other documents, arrays, and arrays of documents.

Schemas/Models

authenticating.ts

```ts
export interface UserDoc extends BaseDoc {
  username: string;
  password: string;
}
```

_id

```
/**
 * Add `item` to the collection.
 * @returns the object ID of the inserted document
 */
async createOne(item: Partial<Schema>): Promise<ObjectId> {
  const safe = this.withoutInternal(item);
  safe.dateCreated = new Date();
  safe.dateUpdated = new Date();
  return (await this.collection.insertOne(safe as OptionalUnlessRequiredId<Schema>)).insertedId;
}
```

ObjectId

```
export interface PostDoc extends BaseDoc {
  author: ObjectId;
  content: string;
}
```

Filtering

Running Example

```
const myDB = client.db("myDB");
const myColl = myDB.collection("fruits");
await myColl.insertMany([
{ "_id": 1, "name": "apples", "qty": 5, "rating": 3 },
{ "_id": 2, "name": "bananas", "qty": 7, "rating": 1},
{ "_id": 3, "name": "oranges", "qty": 6, "rating": 2 },
{ "_id": 4, "name": "avocados", "qty": 3, "rating": 5 },
]);
```

Literal Value Queries

```
const query = { "name": "apples" };
const cursor = myColl.find(query);
```

```
{ "_id": 1, "name": "apples", "qty": 5, "rating": 3 }
```

Comparison Operators

```
// $gt means "greater than"
const query = { qty: { $gt : 5 } };
const cursor = myColl.find(query);
```

```
{ "_id": 2, "name": "bananas", "qty": 7, "rating": 1 }
{ "_id": 3, "name": "oranges", "qty": 6, "rating": 2 }
```

https://www.mongodb.com/docs/manual/reference/operator/query-comparison/

Logical Operators

```
const query = { qty: { $not: { $gt: 5 }}};
const cursor = myColl.find(query);
```

```
{ "_id": 4, "name": "avocados", "qty": 3, "rating": 5 }
{ "_id": 1, "name": "apples", "qty": 5, "rating": 3 }
```

https://www.mongodb.com/docs/manual/reference/operator/query-logical/

Logical Operators

```
async removeFriend(user: ObjectId, friend: ObjectId) {
  const friendship = await this.friends.popOne({
    $or: [
      { user1: user, user2: friend },
      { user1: friend, user2: user },
    ],
  });
  if (friendship === null) {
    throw new FriendNotFoundError(user, friend);
  }
  return { msg: "Unfriended!" };
}
```

https://www.mongodb.com/docs/manual/reference/operator/query-logical/

Evaluation Operators

```
// $mod means "modulo" and returns the remainder after division
const query = { qty: { $mod: [ 3, 0 ] } };
const cursor = myColl.find(query);
```

```
{ "_id": 3, "name": "oranges", "qty": 6, "rating": 2 }
{ "_id": 4, "name": "avocados", "qty": 3, "rating": 5 }
```

https://www.mongodb.com/docs/manual/reference/operator/query-evaluation/

Filtering Resources

https://www.mongodb.com/docs/drivers/node/current/fundamentals/crud/query-document/#std-label-node-fundamentals-query-document

Exercises

Data Model

```
concept Label[Creator, Item]

  state
    labels: set Label
    creator: labels -> one Creator
    name: labels -> one String
    items: labels -> set Item

  actions
    createLabel (s: String, u: Creator, out p: Label)
      label.creator := u;
      label.name := s;
      labels += label;

    getByCreator(u: Creator, out ps: set Label)
      label in labels
          if label.creator == u:
              out ps += label

    affix (i: Item, p: Label)     // labeling an item
      p.items += i;
```

Cursor Methods

options? in doc.ts

```ts
/**
 * Read the document that matches `filter`. Returns `null` if no document matches.
 */
async readOne(filter: Filter<Schema>, options?: FindOptions): Promise<Schema | null> {
  this.sanitizeFilter(filter);
  return await this.collection.findOne<Schema>(filter, options);
}
```

sort()

```
async read(query: Filter<PostDoc>) {
  const posts = await this.posts.readMany(query, {
    sort: { dateUpdated: -1 },
  });
  return posts;
}
```

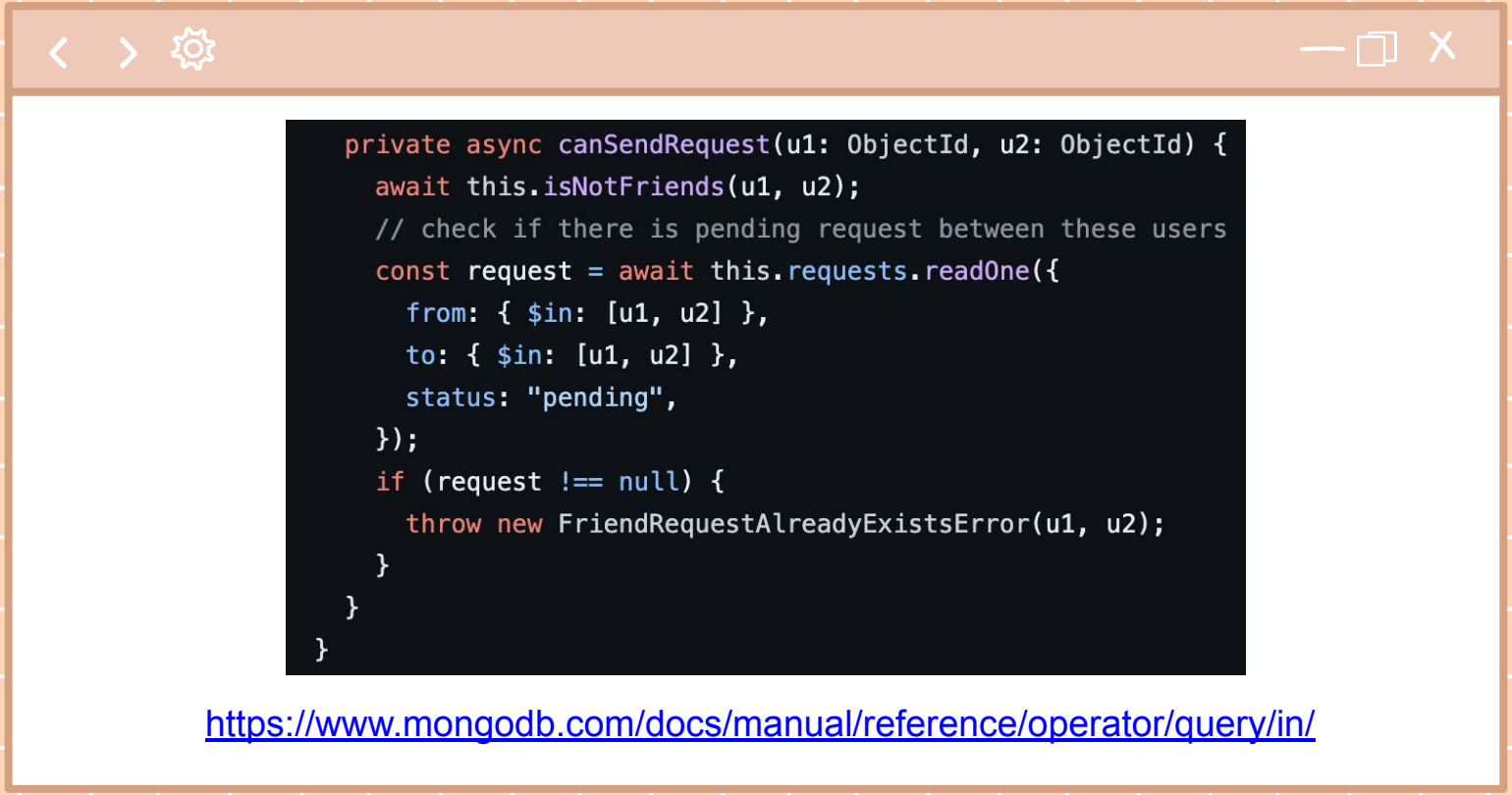
https://www.mongodb.com/docs/manual/reference/method/cursor.sort/

Cursor Methods Exercise

Testing!

Testing Exercise

Exercise Solution



https://www.mongodb.com/docs/manual/reference/operator/query/in/