Logical DB Query and Compound indexes.

Till now we were just working with User model, but now we will introduce us will begin one user will send the connection request to another user.

Complexity of code will increase.

We will start defining the connection request.

There is a reason why we add collections is because it defines something.

User schema ← it identify is data of user.
for request we will create another schema

We will create new model.
→ connectionRequest

To create new model.

import mongoose.
↓

const connectionRequestSchema = new mongoose.Schema({
↓ ↑ name of schema
const ConnectionRequestModel = new mongoose.model(
↓ ↑ (Name of model, schema name)
export connection request model

new enum
{ enum: { values: ["ignored", "interested", "etc"] }
message: '{VALUE} is not a incorrect status type'
}
← Kind of validator.

enum will help to get inserted only specific data to model

we can also do schema validation also.

```
ConnectionRequestSchema.pre("save", function(next) {
const ConnectionRequest = this;

if (connectionRequest.fromUserId.equals(connectionRequest.toUserId)
{
throw new Error("error")
}
next();
})
```

to create user Schema.

```
Const ConnectionRequestSchema = new mongoose.Schema({

toUserId: {
type: mongoose.Schema.type.ObjectId,
required: true,
},
});

there, enum: {
values: ["", "", "", ""],
message: '{VALUES} is incorrect status type',
},
```

Date
Page

Indexing.

why we do indexing in database.

it ~~because~~ because just to increase the ~~time~~ performance of querying in database.

Suppose we have 1m data in Db. and i have search for Name - Praveen. then it will search one by one one by one and search for Praveen and then it will give the result.

↑ it makes slow

to make it faster we use indexing.
if we give index to first name then it will search very quickly via ~~for~~ index

How to use index.

① if we have used (unique:true) in schema, mongodb will automatically create index for this.

or we can do

index: true. ← it will create index at that

(unique index are much faster)

HW : Why should we do not create index unnecessarily.

Date
Page

There is something is known as compound index

ConnectionRequestSchema.index({ fromUserId: 1, toUserId: 1 }) ← asscend

1 = ascending order
-1 = descending order.

compound index mean when ~~else~~ otherwise we will

compound index mean when ever we will run query ~~search~~ for both parameters (both) ← combined then this query will become very fast.

```
const existingConnectionRequest = awaitConnectionRequest
    .findOne({ $or: [{ fromUserId, toUserId }, { fromUserId: toUserId, toUserId: fromUserId }] })
```

{ $or ← it make to find either by first or second basically or operator

Question if index make querying fast then lets/ets us make it on everywhere (Mongodb will optimize it)

Creating index unnecessarily will come with a cost.

what is Advantage and disadvantage of creating index.

Date
Page

Learning.

to compare objects we usually can use .equals() method.

if we want to do any operation before save we can do it in schema level ~~of~~ pre.

schema.pre("save", function(next) {
  const schema = this;

Suppose here we are checking connection request. We got id, from user, to user, status but we want data of that user not user id

To get user details we have two ways

① We can loop over to all the collection request and we can find user data one by one.

② By Building relation between user and request

Basically Building relation between two Schema

We can do it by.

ref: "user" ← give reference to the collection from where you want data

once we created reference now we will just Populate it now

```
const connectionRequest = await connectionRequest.find
({ toUserId: loggedInUserId,
  Status: "interested",
}).populate("fromUserId", ["firstName", "lastName"])
```

filter ① way

② way "firstName lastName"

Always give this way

Always be very specific data that you are fetching.