1. [10 minutes]

What 3 problems does a relational database have and that is a valid reason to consider a NoSQL database because most of them do not have any of these 3 problems.

1. Relational databases don't scale well because of strict consistency
2. Relational databases are hard to change because of the fixed schema
3. Relational databases don't handle unstructured or semi structured data very well.

2. [10 minutes]

Select all statements that are correct

□ A. JMS calls are idempotent

□ B. DTO classes are immutable

□ C. Value objects are immutable

□ D. Entity classes are immutable

□ E. An entity object in domain driven design is stateless

□ F. A value object in domain driven design is stateless

□ G. A domain service object in domain driven design is stateless

□ H. The domain service object in domain driven design resides in the service layer

□ I. A custom router in spring integration returns a boolean

□ J. A custom router in spring integration returns a String

3. [5 minutes]

Give the 4 main features offered by the integration broker in a hub and spoke architecture.

transport
transformation
routing
orchestration

4. [10 minutes]

a. Explain the difference between horizontal and vertical scaling

horizontal scaling: add more servers

vertical scaling: add more resources like CPU, RAM, bigger disk

b. Explain clearly why in master-slave replication all writes have to go the same database master.

So that we can have strict consistency. The Cap theorem tells us that we cannot have partition tolerance, availability and strict consistency all at the same time.

5. [10 minutes]

a. Explain the difference between sharding and replication

Sharding: divide the rows over multiple database instances

Replication: copy the data to multiple replicas

b. Can sharding and replication be used together? If no, why not? If yes, why would we use them both?

Yes, sharding to support scalability and replication to support failover

6. [10 minutes]

Suppose we have the following relational database tables:

Student

studentid	firstName	lastName
5566	Frank	Brown
5532	John	Doe

Course

courseNumber	courseName
CS590	Software Architecture
CS525	Advanced Software Development

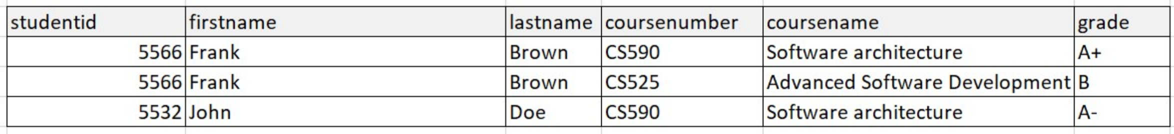
Student_Course

studentid	courseNumber	grade
5566	CS590	A+
5566	CS525	B
5532	CS590	A-

Because the data does not fit on 1 node anymore we decided to replace our relational database with a Cassandra database. It is an important requirement that the application that uses this database should give the same fast performance that we had when we used the relational database.
The application has 2 main pages. The first page allows us to see all grades for a certain student. The second page allows to see all grades from a certain course.
Show the structure of the table (or tables) that we need in Cassandra including the data in the table(s). Specify what column is the partition key.

Grades-per-student table

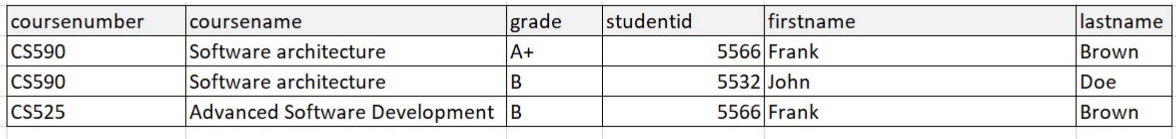
Studentid in partition key



studentid	firstname	lastname	coursenumber	coursename	grade
5566	Frank	Brown	CS590	Software architecture	A+
5566	Frank	Brown	CS525	Advanced Software Development	B
5532	John	Doe	CS590	Software architecture	A-

Grades-per-course table

Course number is partition key

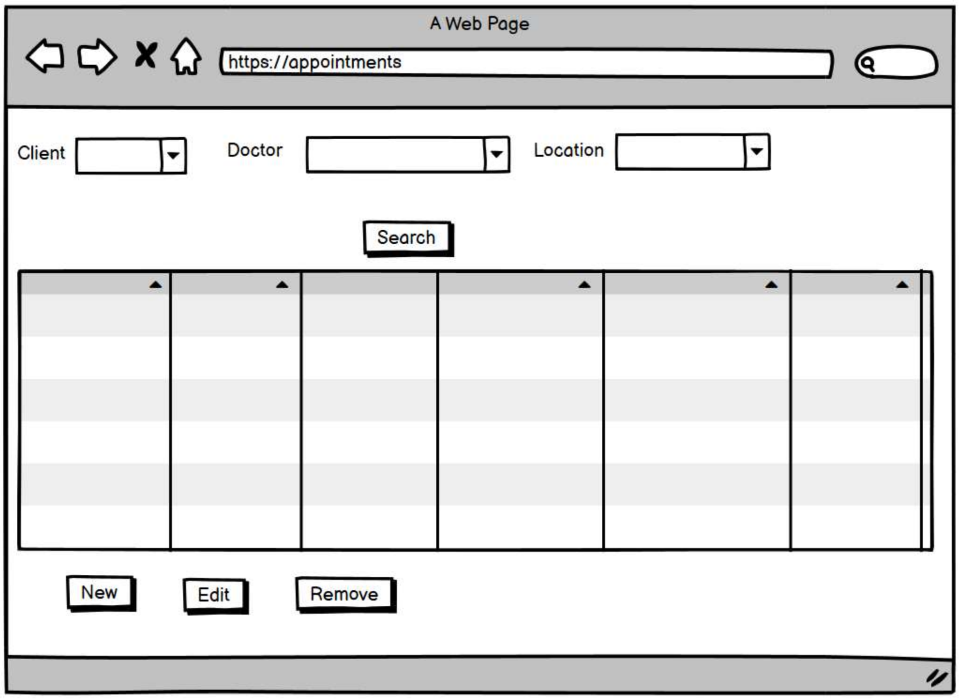


coursenumber	coursename	grade	studentid	firstname	lastname
CS590	Software architecture	A+	5566	Frank	Brown
CS590	Software architecture	B	5532	John	Doe
CS525	Advanced Software Development	B	5566	Frank	Brown

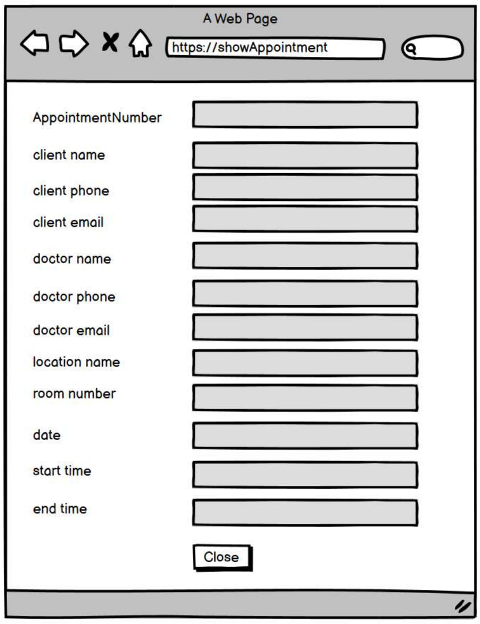
7. [55 minutes]

Doctors appointment application

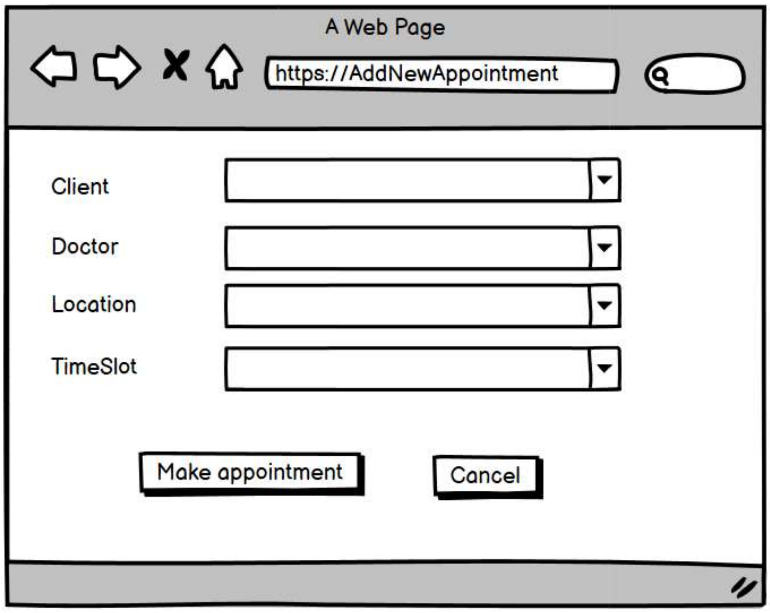
Suppose we need to design a simple doctor's appointment application using DDD. The user interface looks as follows:



The user can see the appointments for a selected client, doctor or location. The user can also add a new appointment, edit an existing appointment or remove an appointment. In the control that shows all appointments, you can also click on an appointment which will show the following appointment details window:

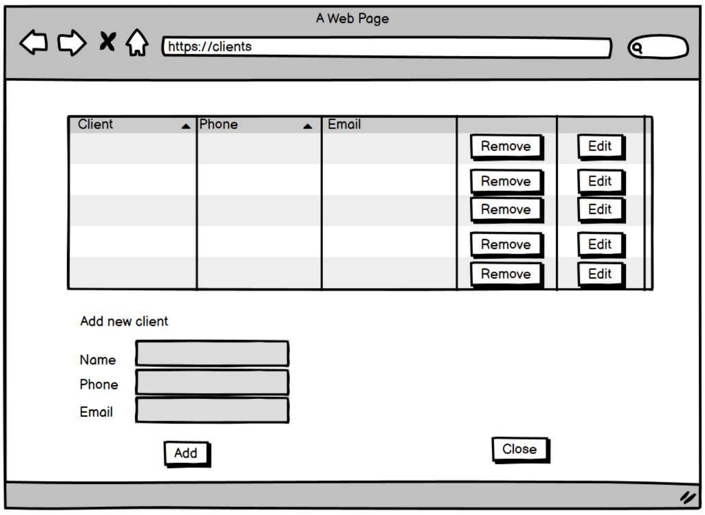


When you click the **Add** button you can add a new appointment using the following window:



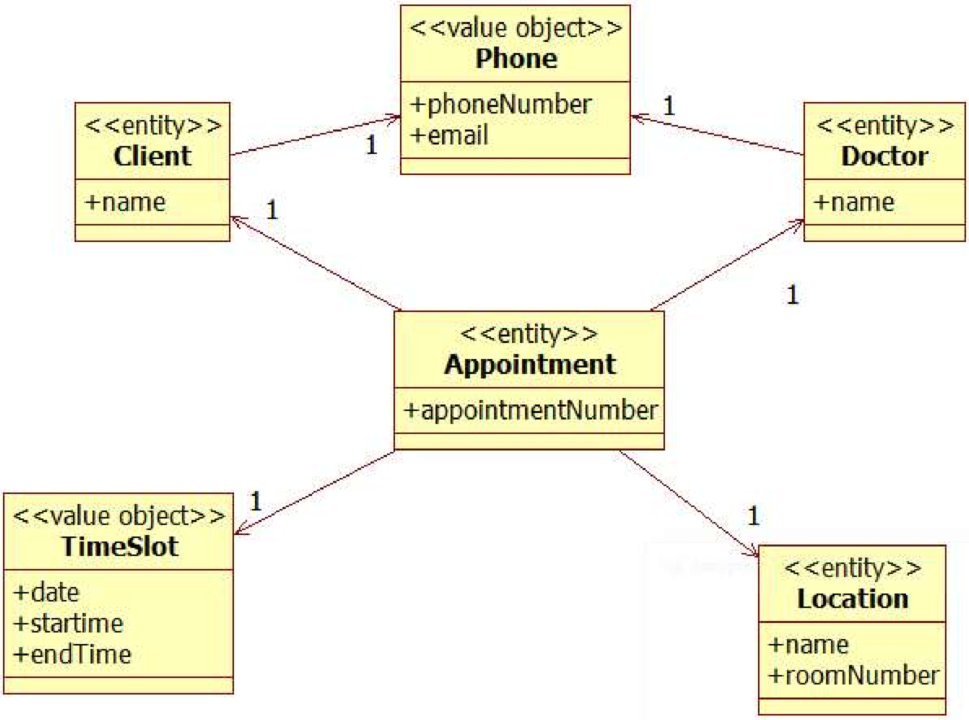
The time is divided into fixed timeslots.
A client can make only 3 appointments. For every appointment the system needs to check if the location and the doctor is available at that timeslot.

The application also allows you to add, edit, remove or view clients, doctors and locations:

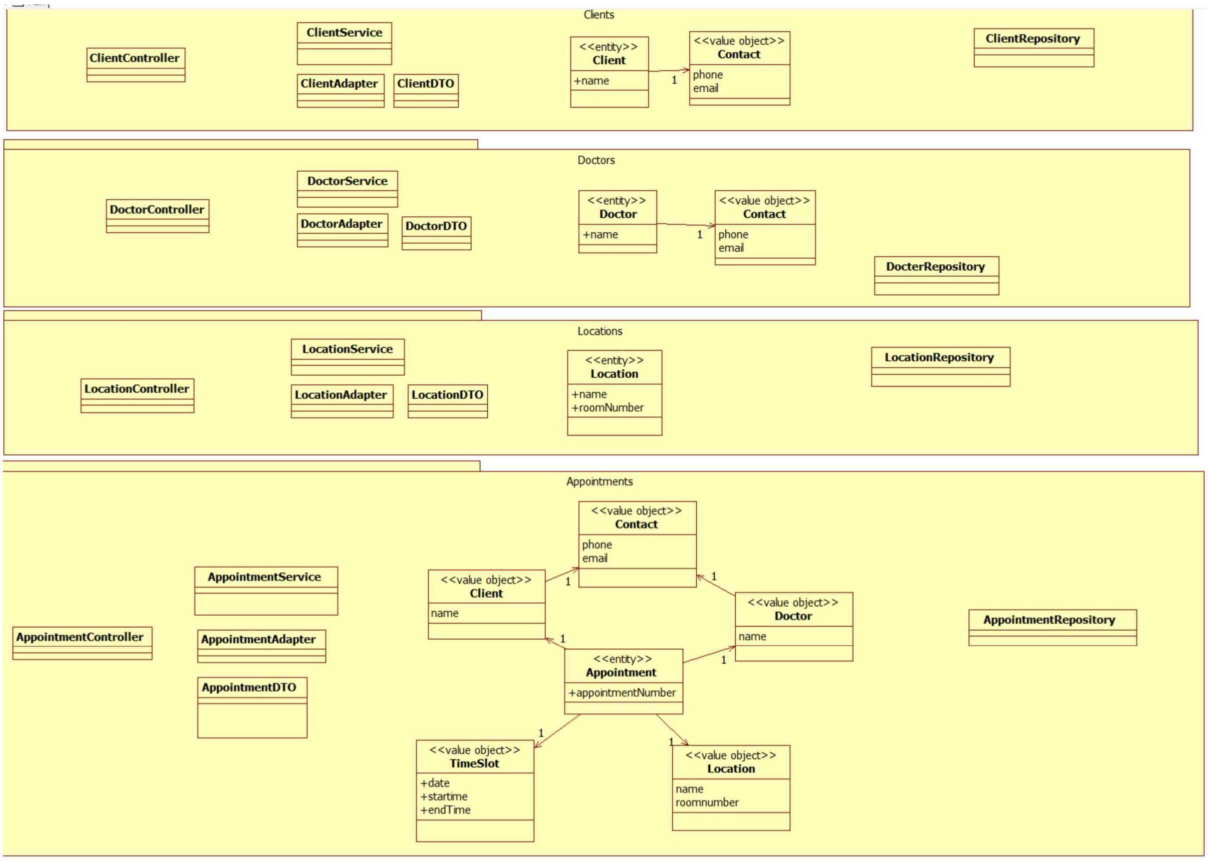


a. Draw a class diagram of the domain classes (classes, attributes and relationships). For every class indicate the type of the class (entity, etc).
b. Suppose we want to divide this application into components. Draw in one class diagram all components including **ALL** classes inside the components. For every domain class indicate the type of the class (entity, etc). For every class show all attributes.

A


<<value object>>
Phone
+phoneNumber
+email
<<entity>>
Client
+name
<<entity>>
Doctor
+name
<<entity>>
Appointment
+appointmentNumber
<<value object>>
TimeSlot
+date
+startime
+endTime
<<entity>>
Location
+name
+roomNumber
1
1
1
1
1
1

B


Clients
ClientController
ClientService
ClientAdapter
ClientDTO
<<entity>>
Client
+name
<<value object>>
Contact
phone
email
1
ClientRepository
Doctors
DoctorController
DoctorService
DoctorAdapter
DoctorDTO
<<entity>>
Doctor
+name
<<value object>>
Contact
phone
email
1
DocterRepository
Locations
LocationController
LocationService
LocationAdapter
LocationDTO
<<entity>>
Location
+name
+roomNumber
LocationRepository
Appointments
AppointmentController
AppointmentService
AppointmentAdapter
AppointmentDTO
<<value object>>
Contact
phone
email
<<value object>>
Client
name
<<value object>>
Doctor
name
<<entity>>
Appointment
+appointmentNumber
<<value object>>
TimeSlot
+date
+startime
+endTime
<<value object>>
Location
name
roomnumber
1
1
1
1
1
1
AppointmentRepository