EXERCISES

EXERCISE 1 – BOOKS

We are required to design a MongoDB database to store the following data about books.

Each book is identified by the ISBN code and it is characterized by its title, subtitle, the number of pages, the language ('IT', 'EN', 'FR', etc.) and the publication date.

The system is required to store the book prices (e.g., 12.34$), for each country and for each format (e.g., 'ebook', 'paperback'). Please note that book prices are searched based on their currency (e.g., all book prices in dollars $, or pounds £), and separately also on their amount (e.g., higher than 10.0).

Each book belongs to one or more categories (e.g., 'fantasy', 'classical'). Categories are organized in a hierarchical structure (e.g., 'children' category includes 'fiction', which includes 'fantasy', 'classical', etc.).

Indicate for each collection in the database the document structure and the strategies used for modelling.

EXERCISE 1 – BOOKS

book

```
{_id: '1234-5678',  // ISBN
 title: 'Book Title',
 subtitle: 'Subtitle of the book',
 pages: 123,
 lang: 'EN',
 published: 2021-01-31
 prices: [
          {
           format: 'ebook',
           country: 'UK',
           amount: 12.34,
           currency: '£'
          },
          {
           format: 'paperback',
           country: 'US',
           amount: 23.45,
           currency: '$'
          }

         ],
categories: ['fantasy','classical']
ancestors:  ['fiction', 'children']
}
```

Attribute pattern to track the price according to the book format and its country

Tree pattern to track all the ancestors in the category hierarchy

EXERCISE 2 – PROPERTY RENTAL REVIEWS

We are required to design a MongoDB database for the management of a website similar to AirBnb where property rental reviews from users are collected.

Each property is characterized by a name, the price per night, and its location (postal code, city, and country). The price per night is characterized by its amount and its currency.

Each review is characterized by the timestamp, the text of the review, the grade and the author information, and it is related to only one property. When accessing a review, only the name and the email of the author who wrote the review are displayed.

When accessing a property, only the top 10 most recent reviews are displayed, each presenting only its text, grade and timestamp. Each property page reports the number of total reviews received, and the average grade.

- Indicate for each collection in the database the document structure and the patterns used for modelling.

EXERCISE 2 – PROPERTY RENTAL REVIEWS

Properties

```
{_id: <ObjectId>,
 name: <string>,
 price: {  amount: <double>,
           currency: <string>},
 location:{
           postal_code: <int>,
           city: <string>,
           country: <string>,
          },
 reviews:[   // top 10 most recent
           {review_id: <ObjectId>
            text: <string>,
            timestamp: <datetime>,
            grade: <double>}
         ],
 nReviews: <int>,
 avgGrade: <double>
}
```

Reviews

```
{_id: <ObjectId>,
 property_id: <ObjectId>,
 text: <string>,
 timestamp: <datetime>,
 grade: <double>,
 author: { name: <string>,
           email: <string>}
}
```

Subset pattern to track only the most recent reviews of each property.

Extended reference pattern for the properties, to display the relevant information of the reviews without joins.

Computed pattern to store the overall number of reviews and their average grade.

EXERCISE 3 – INSURANCE

We are required to design the database for the management of an insurance application.

Each customer is characterized by the name, surname, birthdate, the contact methods and the home address. The contact methods can be, for example, the telephone number, the email, a Skype username, etc. The address is characterized by the street name, street number, city, province and region.

Each insurance policy is signed by a customer and is characterized by its type, the date of signature, and the list of included items. The included items can be modified by the customer over time, by adding new ones or removing undesired ones.

The application should efficiently retrieve the currently active insurance data. However, previous policy versions must be available on request.

In addition to the insurance details, it is necessary to show only the name, surname and the contact methods of the customer.

Indicate for each collection in the database the document structure and the strategies used for modelling.

EXERCISE 3 – INSURANCE

Insurance policy (latest version only)

```
{_id: <ObjectId>,
 type: <string>,
 date: <date>,
 customer: {user: <ObjectId>,
            name: <string>,
            surname: <string>,
            contacts: {email: <string>,
                       mobile: <string>,
                       skype: <string>}
           },
 items: [<string>],
 version: <number>
}
```

Insurance_rev (previous policy versions)

```
{_id: < ObjectId >,
 type: <string>,
 date: <date>,
 customer: {user: <ObjectId>,
            name: <string>,
            surname: <string>,
            contacts: {email:<string>,
                       mobile: <string>}
           },
 items: [<string>],
 version: <number>
}
```

Customer

```
{_id: <ObjectId>,
 name: <string>,
 surname: <string>,
 contacts: {email: <string>,
            mobile: <string>,
            skype: <string>},
 birthdate: <date>,
 address: {
          street: <string>,
          number: <string>,
          city: <string>,
          province: <string>,
          region: <string>
}
```

Document versioning for the updates of insurance policies

Extended reference for the customer information

Polymorphic pattern to track only the contact methods that are available

ACKNOWLEDGMENT

BIBLIOGRAPHY

For further information on the content of these slides, please refer to the book

“Design with MongoDB”

Best Models for Applications

by Alessandro Fiori

https://flowygo.com/en/projects/design-with-mongodb/