Large-Scale and Multi-Structured Databases

Ecommerce Application Design Review Part 1

Prof. Pietro Ducange

Eng. José Corcuera

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

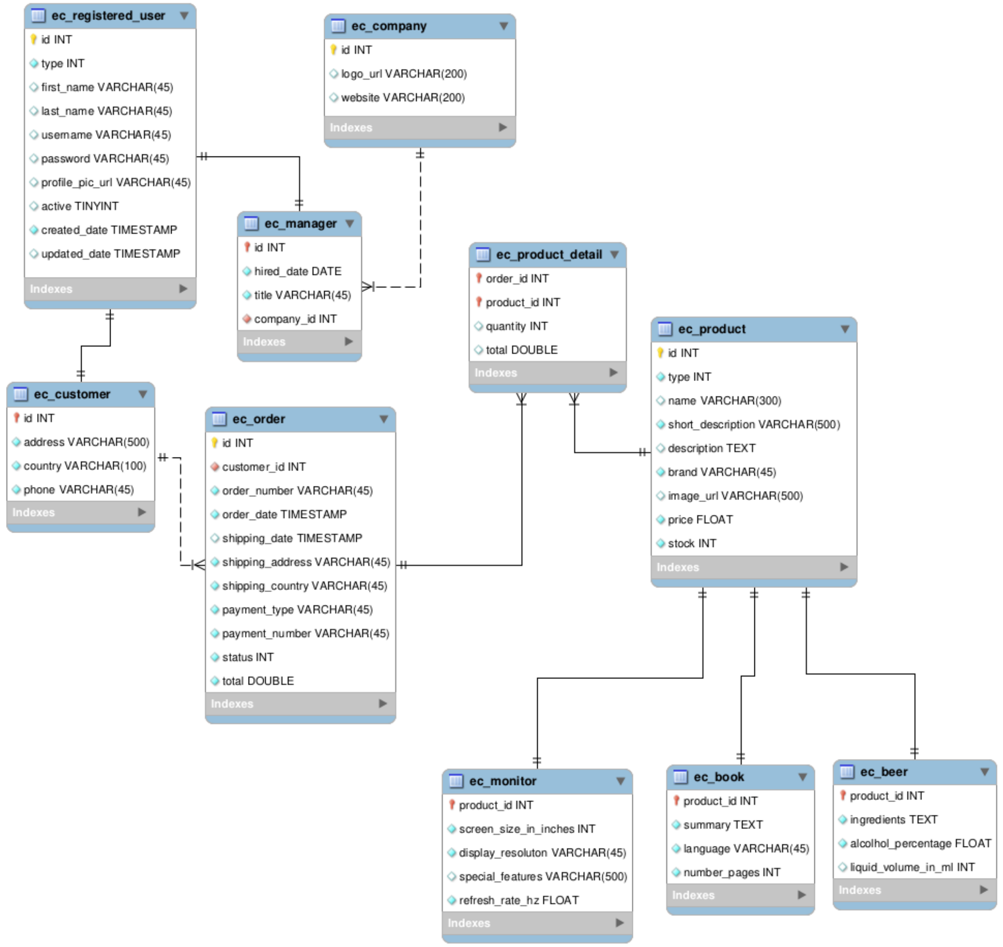
UNIVERSITÀ DI PISA

CROSSLAB Innovation for industry 4.0

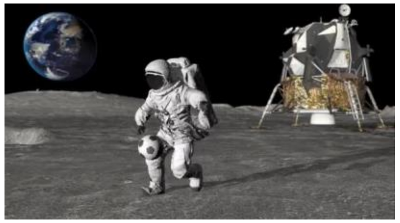
Objective of this Class

- To review the design of the database using a RDBMS.
- To review the **initial version** of the **Software Architecture**.
- To review the **target version** of the **Software Architecture**.
- To review the **Software Layered Architecture**.
- To review **some pieces of code.**
- To review **open questions**.
- Exercises.

Design of the database using a RDBMS



Database schema



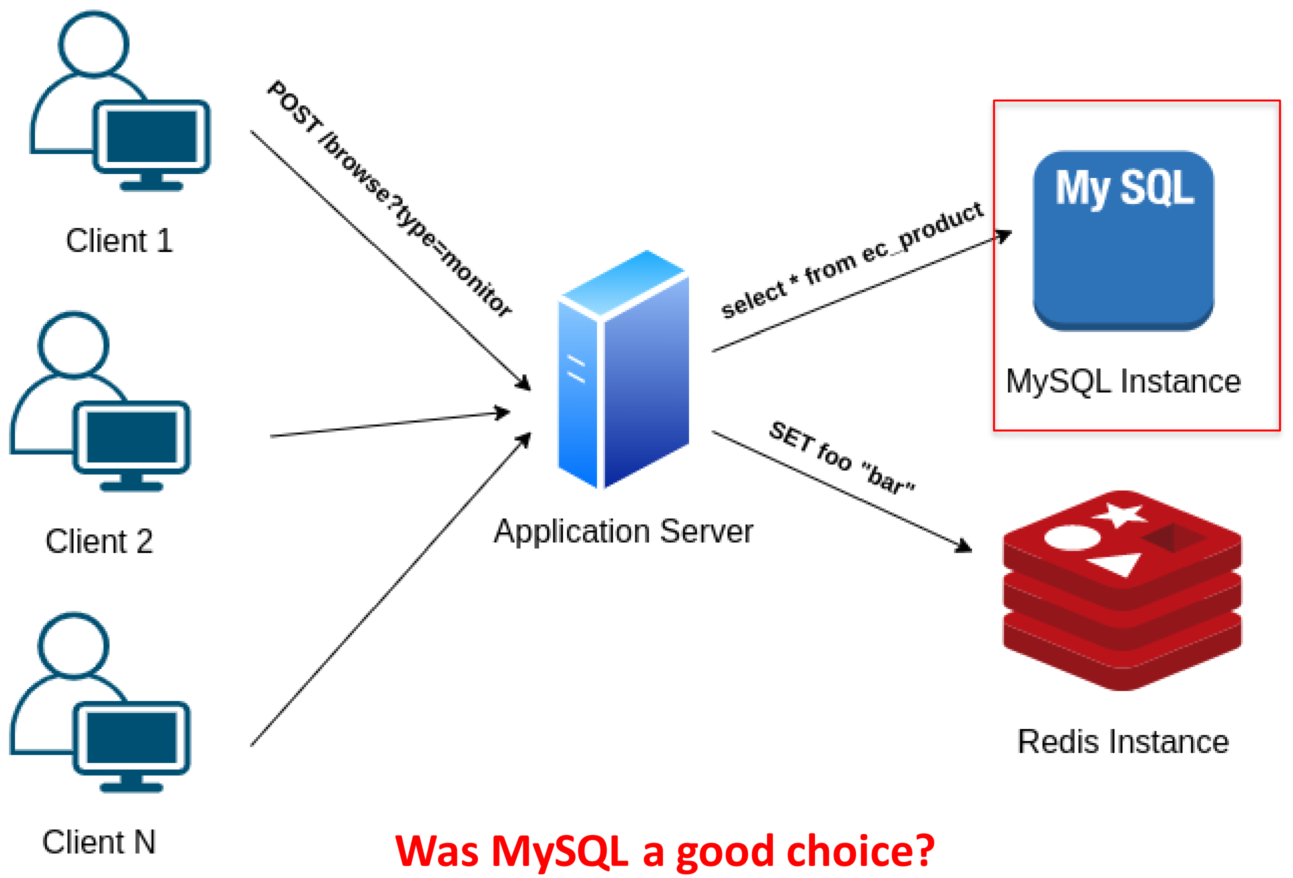
After inserting 10K products, a query on **ec_product** table goes slow, what could the cause be?

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

IN SUPREMÆ DIGNITATIS 1343
UNIVERSITÀ DI PISA

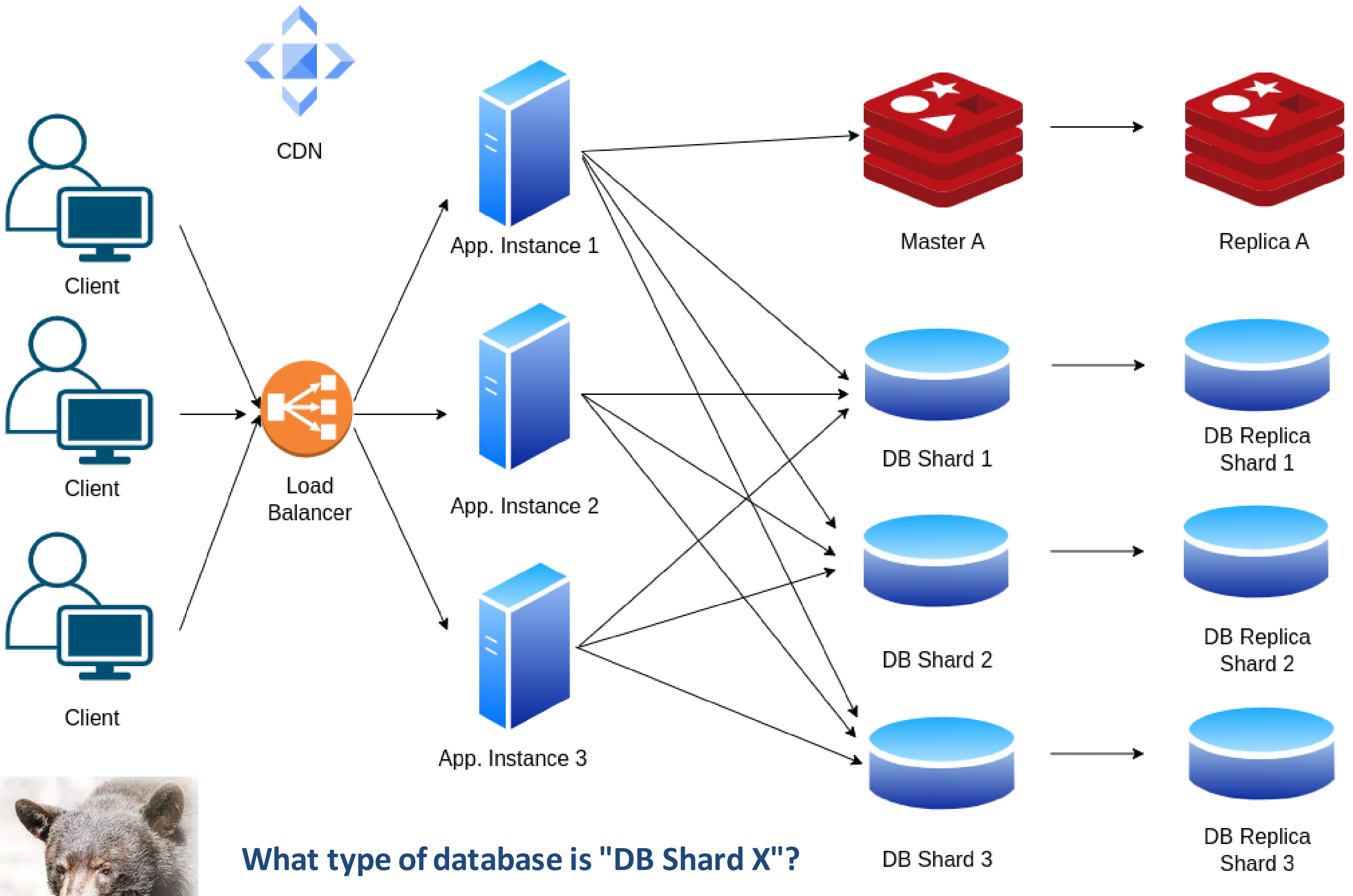
CROSSLAB
Innovation for industry 4.0

Initial version of the Software Architecture



Was MySQL a good choice?

Target version of the Software Architecture



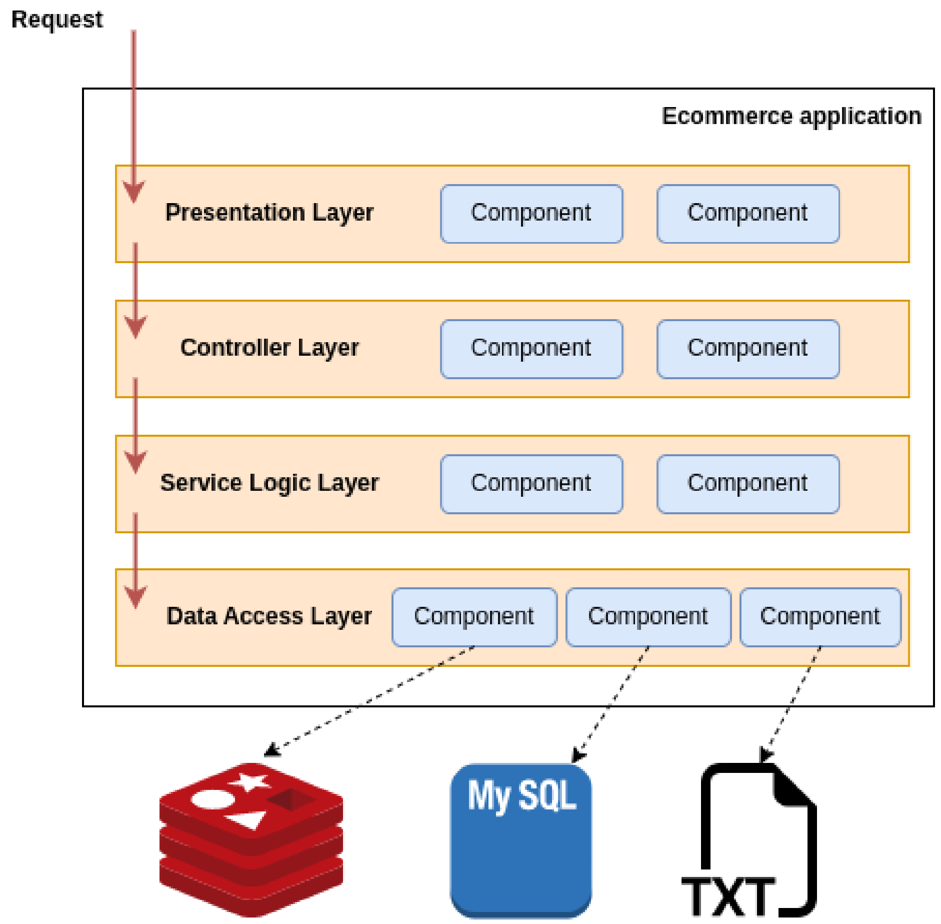
What type of database is "DB Shard X"?

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ DI PISA

CROSSLAB Innovation for industry 4.0

Software Layered Architecture



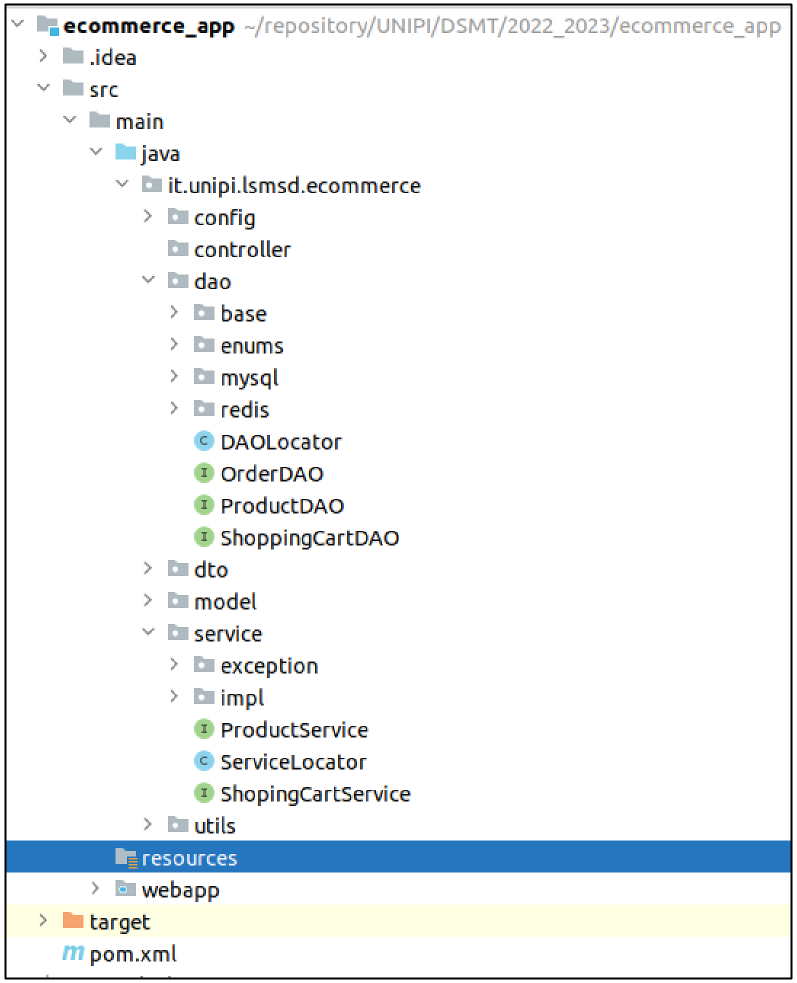
Model–view–controller (MVC) software architectural pattern

Model: Manages the data and the business logic.

View: Frontend, what a user sees.

Controller: It determines what action a user wants to execute.

Some pieces of code (1) - Package organization



```
ecommerce_app ~/repository/UNIPI/DSMT/2022_2023/ecommerce_app
  .idea
  src
    main
      java
        it.unipi.lsmsd.ecommerce
          config
          controller
          dao
            base
            enums
            mysql
            redis
            DAOLocator
            OrderDAO
            ProductDAO
            ShoppingCartDAO
          dto
          model
          service
            exception
            impl
            ProductService
            ServiceLocator
            ShopingCartService
          utils
      resources
      webapp
  target
  pom.xml
```

Classes grouped by layer.

Some pieces of code (2) - Entities

```
public abstract class RegisteredUser {

    public Long id;
    private String firstName;
    private String lastName;
    private String username;
    private String password;
    private String profilePicUrl;
    private Boolean active;
    private Date createdDate;
    private Date updatedDate;
```

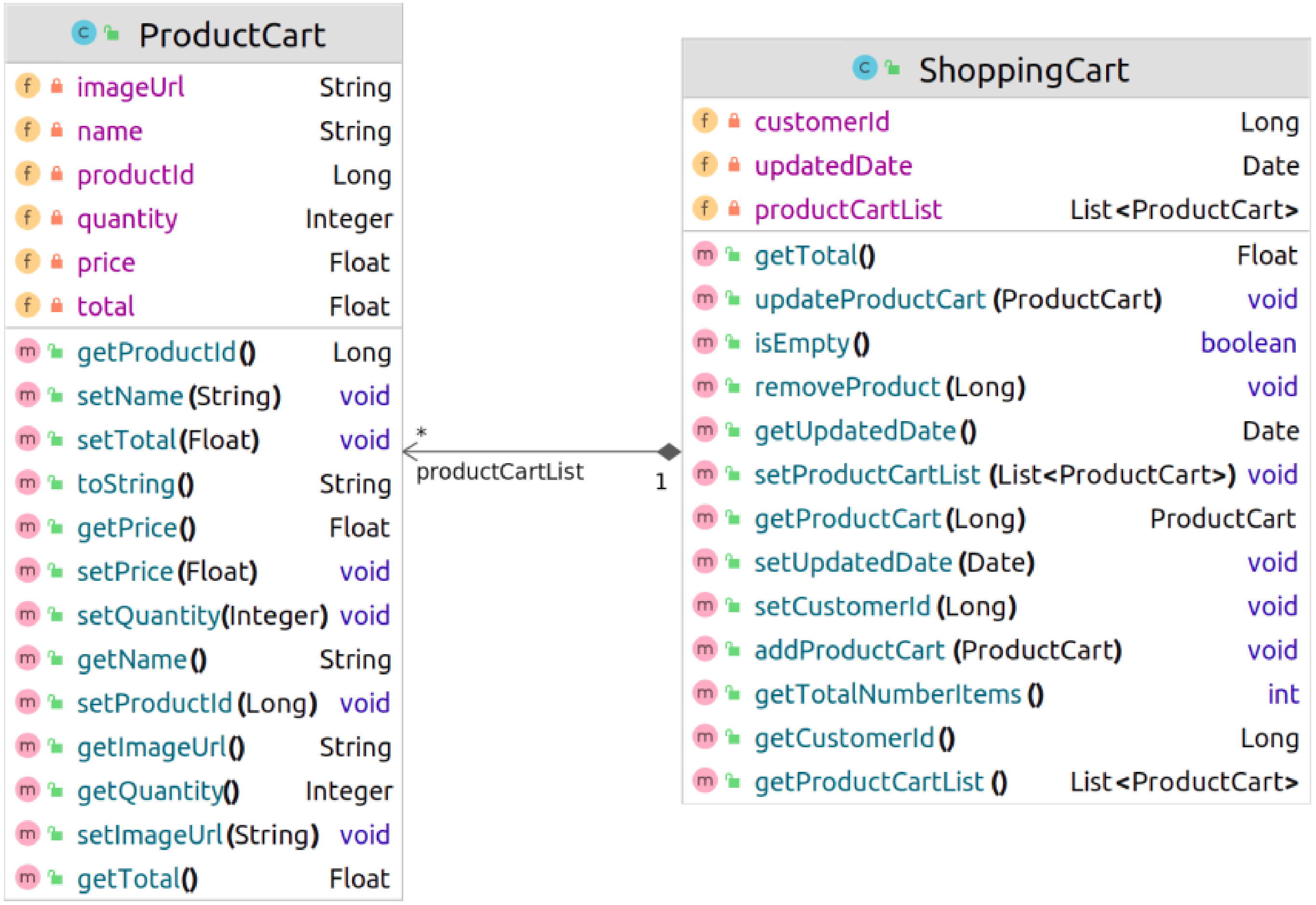
Why?

Inheritance

```
public class Customer extends RegisteredUser {
    private String address;
    private String country;
    private String phone;
```

```
public class Manager extends RegisteredUser {

    private Date hiredDate;
    private String title;
    private Company company;
```

Some pieces of code (3) - Entities



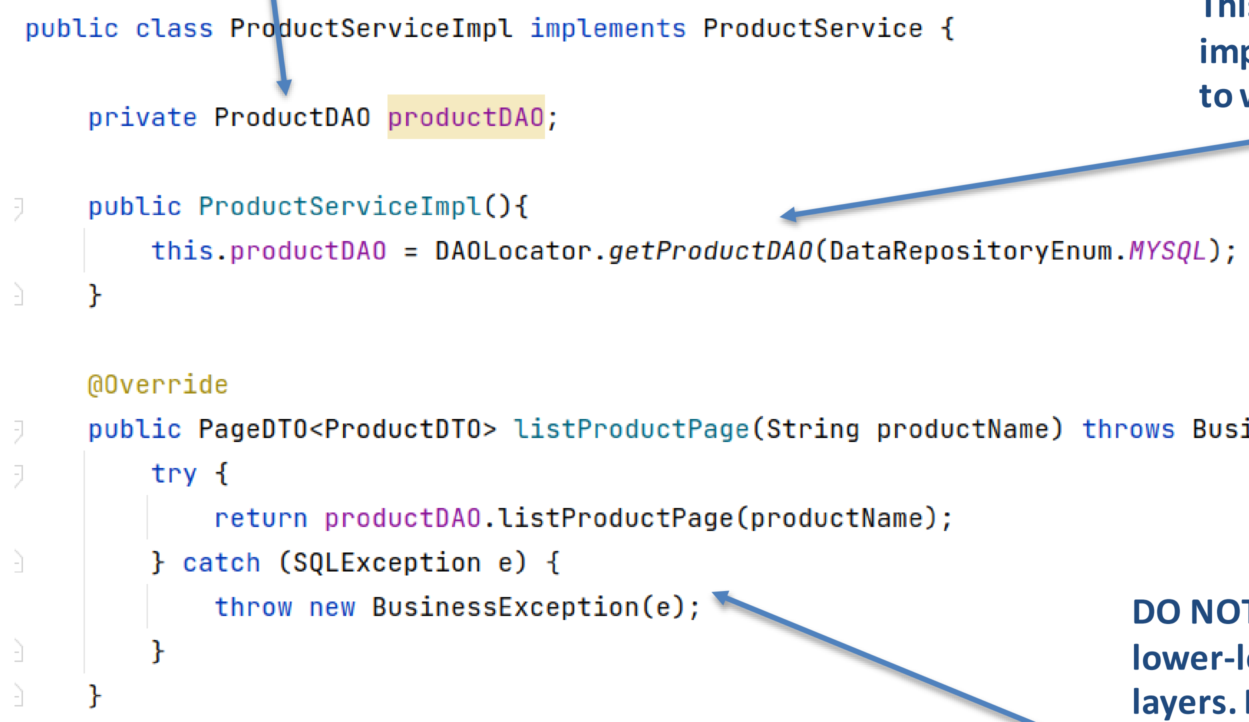
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ DI PISA

CROSSLAB Innovation for industry 4.0

Some pieces of code (4) - Example of a service

Make use of Interfaces or Abstract classes to support different behaviors.

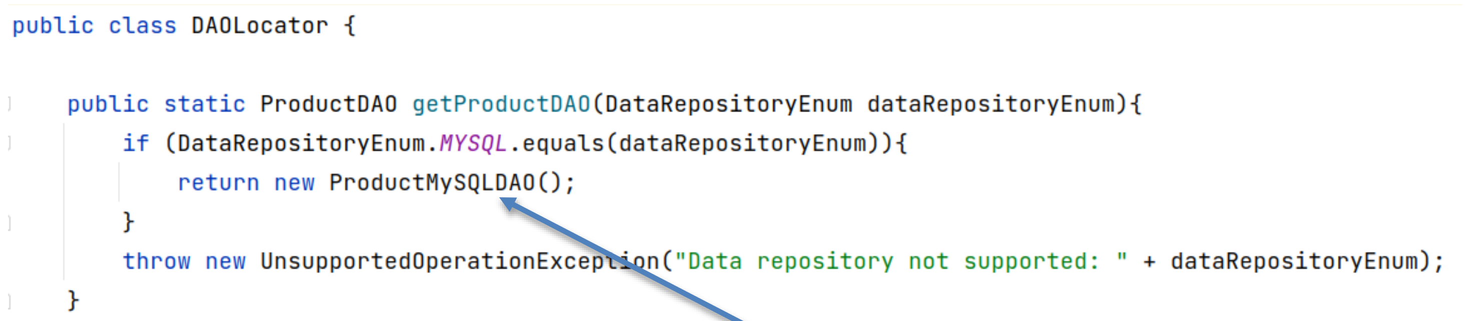


```
public class ProductServiceImpl implements ProductService {

    private ProductDAO productDAO;

    public ProductServiceImpl(){
        this.productDAO = DAOLocator.getProductDAO(DataRepositoryEnum.MYSQL);
    }

    @Override
    public PageDTO<ProductDTO> listProductPage(String productName) throws BusinessException {
        try {
            return productDAO.listProductPage(productName);
        } catch (SQLException e) {
            throw new BusinessException(e);
        }
    }
```

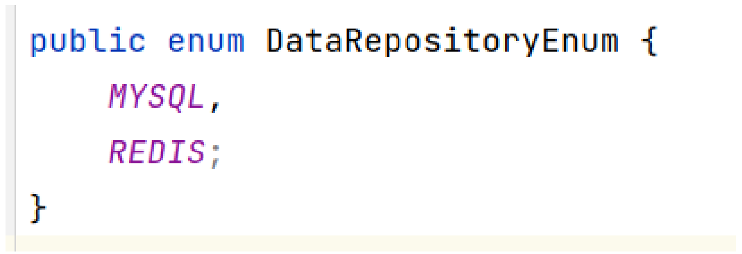
This locator returns a specific DAO implementation. In this case, a DAO to work with MySQL.

DO NOT propagate exceptions from lower-level layers to upper-level layers. Define your own Exception and define a custom message. Also, write into a log file the stack trace of the error.

Some pieces of code (5) - Example of a DAO provider



```
public class DAOLocator {

    public static ProductDAO getProductDAO(DataRepositoryEnum dataRepositoryEnum){
        if (DataRepositoryEnum.MYSQL.equals(dataRepositoryEnum)){
            return new ProductMySQLDAO();
        }
        throw new UnsupportedOperationException("Data repository not supported: " + dataRepositoryEnum);
    }
```



```
public enum DataRepositoryEnum {
    MYSQL,
    REDIS;
}
```

Implementation to work with MySQL. Other implementations could be supported.

You can set which implementation to use by using a configuration file...

Some pieces of code (6)

Let's review the shopping cart checkout implementation.

Open questions

- Some operations need to perform operations on either MySQL or Redis, how do we handle the atomicity of a business logic operation?
  - Example: When you complete the checkout, all products in the shopping cart must be removed. What happened if this last operation fails? Is it required to roll back the checkout?
- So far, we have 3 products and, for each of them we have an entity.
  - In case I want to support a new product type, do I need to create its entity?
- Can a request bypass a layer?

Only when a layer is marked as Open.

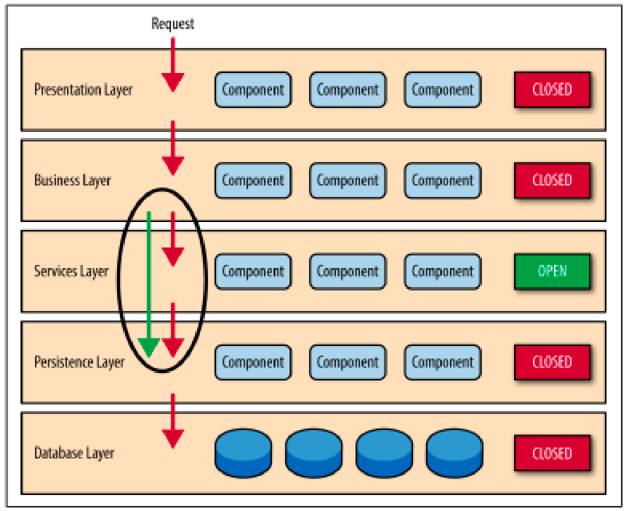
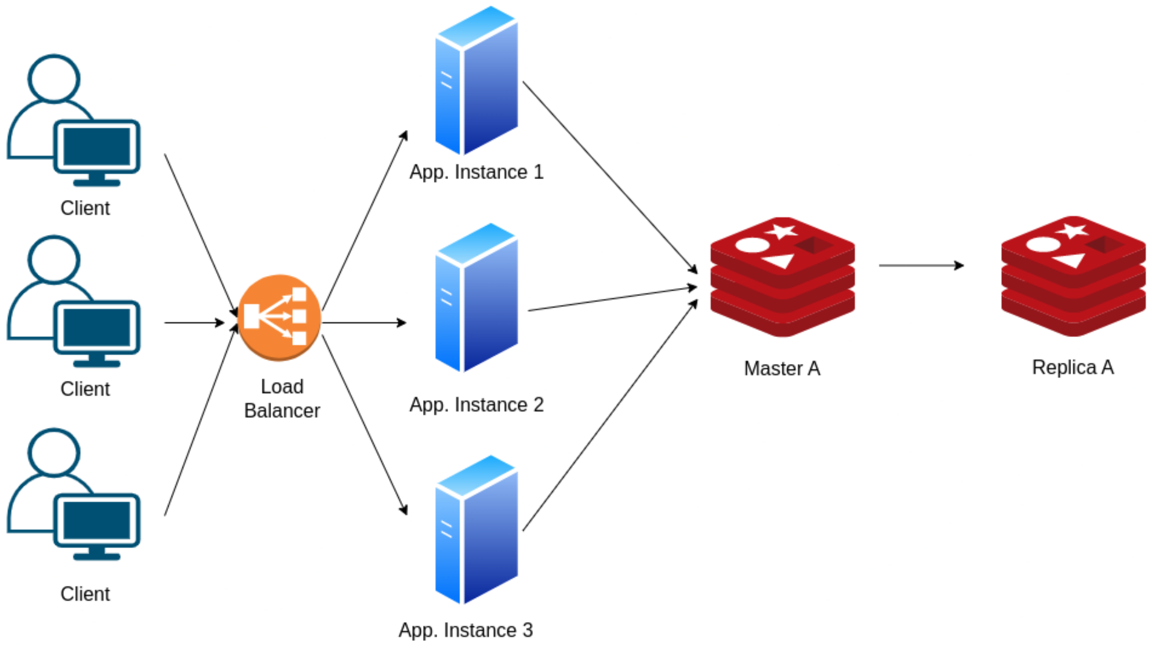


Image from: https://get.oreilly.com/rs/107-FMS-070/images/Software-Architecture-Patterns.pdf, page 5

Exercise 1: HA Application Design choices

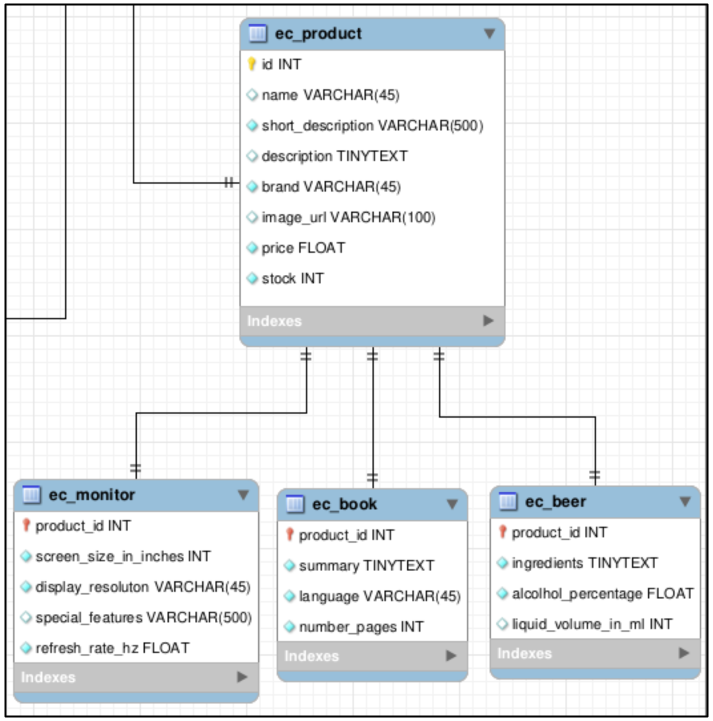
An IT department of a well-known Ecommerce has reported issues on ***PRODUCTION environment*** *due to the increasing number of sales in Black Friday. After reviewing the log files, they realized that* ***CPU utilization in Redis Master node was 100%****. What would you suggest to improve this situation?*



Exercise 2: Ecommerce database

During the last weeks, you have been working in the implementation of an Ecommerce application. You designed a database to support different types of products. Discussing about this point:

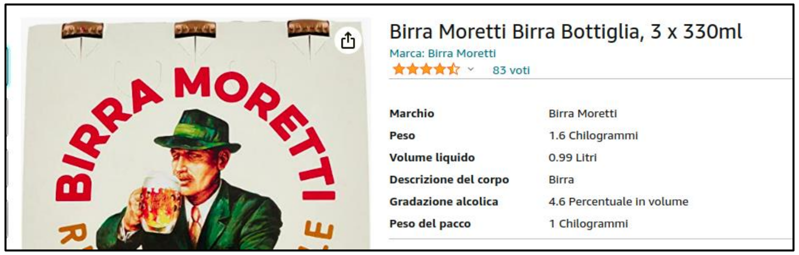
1. *What are the steps to a new product type?*
2. *Which other option do you know to implement this?*
3. *For question number (2), please provide some examples of how the information could be stored for each product type.*
4. *How easy is identifying the product type in this new approach?*



Exercise 3: Ecommerce application

In the previous exercise, we realized that using a RDBMS complicates supporting new product types. With the new database:

1. *What are the changes to be done in the application?*
2. *How do the application know the information to display in a product page? **Remember, each product has different fields**.*
3. *What about if I want to add a new product type?*

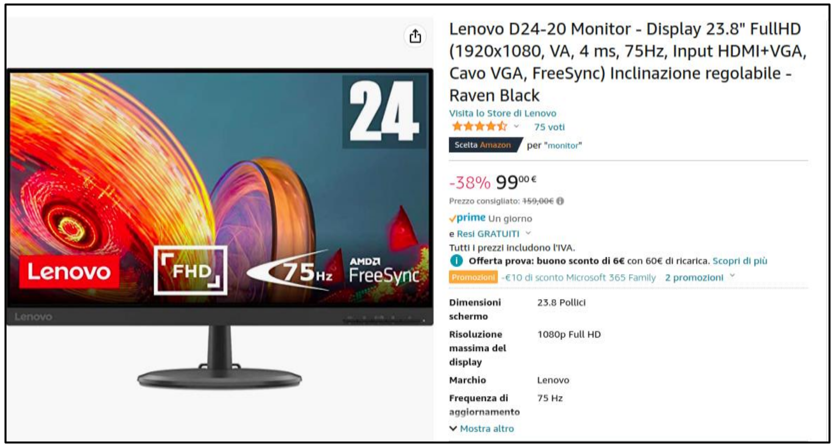


Birra Moretti Birra Bottiglia, 3 x 330ml

Marca: Birra Moretti

★★★★☆ 83 voti

Marchio	Birra Moretti
Peso	1.6 Chilogrammi
Volume liquido	0.99 Litri
Descrizione del corpo	Birra
Gradazione alcolica	4.6 Percentuale in volume
Peso del pacco	1 Chilogrammi



Lenovo D24-20 Monitor - Display 23.8" FullHD (1920x1080, VA, 4 ms, 75Hz, Input HDMI+VGA, Cavo VGA, FreeSync) Inclinazione regolabile - Raven Black

Visita lo Store di Lenovo

★★★★☆ 75 voti

Scelta Amazon per "monitor"

-38% 99,00 €

Prezzo consigliato: ~~159,00€~~

prime Un giorno

e Resi GRATUITI

Tutti i prezzi includono l'IVA.

Offerta prova: buono sconto di 6€ con 60€ di ricarica. Scopri di più

Promozioni -€10 di sconto Microsoft 365 Family 2 promozioni

Dimensioni schermo	23.8 Pollici
Risoluzione massima del display	1080p Full HD
Marchio	Lenovo
Frequenza di aggiornamento	75 Hz

Mostra altro

Exercise 4: Shopping Cart Design choices

You were hired as a Software Engineer in a well-known Company. This company implemented an E-commerce and you have asked to modelling how the information of shopping carts can be stored into a Redis database. Your design must take into consideration the following requirements:

1. *It must be easy to know, for each customer, which products are in their shopping cart.*
2. *It must be cheap the processing in answering what the potential income from sales could be.*
3. *Suppose a product is not more active (or deleted) and many users have that product in their shopping cart, it must be easy to remove it from them.*

You must motivate your solution and mention possible problems/situations that could impact it.

References

- https://get.oreilly.com/rs/107-FMS-070/images/Software-Architecture-Patterns.pdf
- https://www.joelonsoftware.com/2006/11/21/choices-headaches/