

PERSISTENCE DESIGN

Chapter 6

Software Engineering
Computer Science School
DSIC – UPV




Goals

- Understand the need of maintaining the persistence in the development of software
- Know the Data Access Pattern to be used in implementation to achieve a layer abstraction
- Understand the DB object model vs. the relational model and know its advantages
- *Note:* The relational logical design of the DB from an OO model (class diagram) will be covered in another course (Databases).

Contents

1. Introduction
2. DAO data access pattern
3. Persistence in ORDB and OODB
4. Conclusions

References

-  Presman, R.S., Ingeniería del Software: un enfoque práctico (6ª ed.), McGraw-Hill, 2005.
-  Sommerville, I. Ingeniería del Software. (8ª ed.). Addison-Wesley, 2008.
-  Weitzenfeld, A., Ingeniería del Software Orientada a Objetos con UML, Java e Internet. Thomson, 2005.

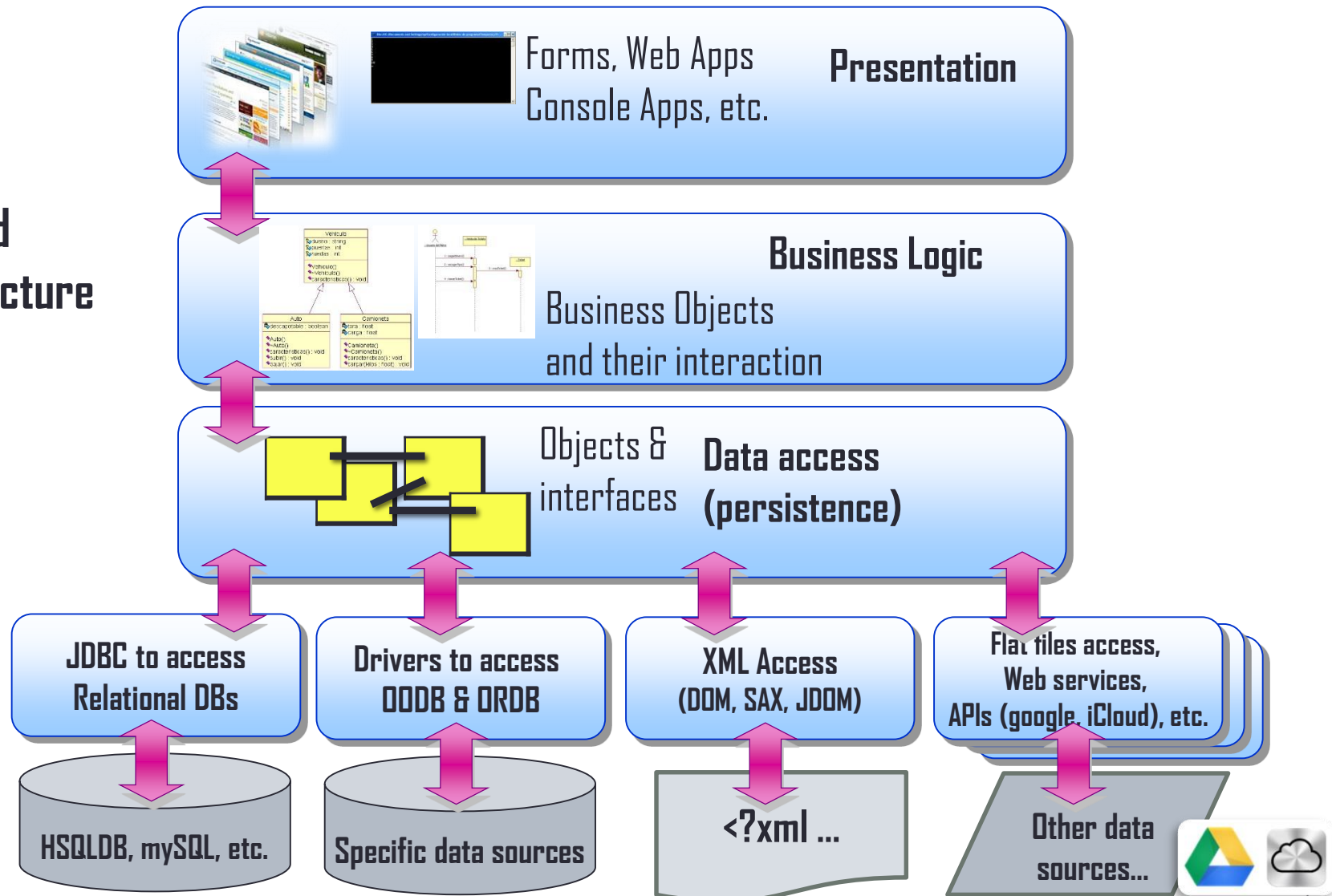
INTRODUCTION

Introduction

- In most applications the storage of non volatile information is essential
 - A specific format may be used for each application (limited compatibility)
 - A structured or relational format based on DB may be used (greater compatibility – based on standards such as SQL)
- The use of DBs results in using libraries to manage the access to data (JDBC, ADO, ODBC, etc.)

Introduction

Layered Architecture



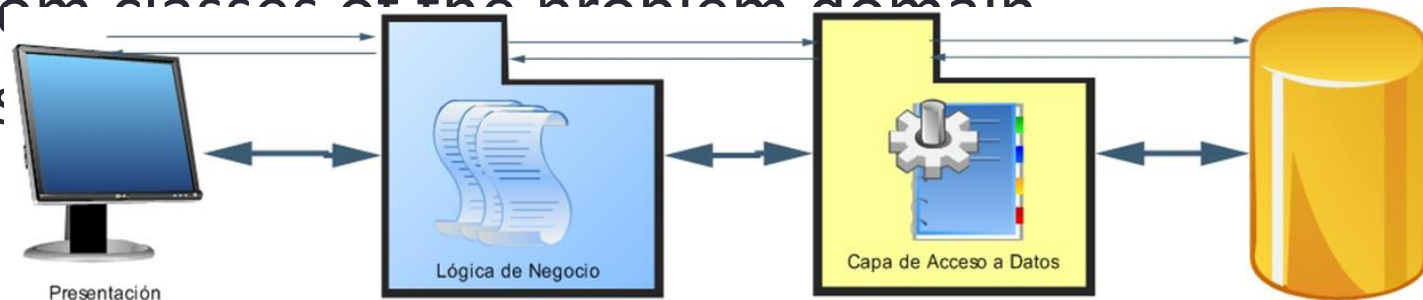
DAO DATA ACCESS PATTERN

- ✓ Structure
- ✓ Pros and Cons
- ✓ Implementation

Data access classes in the implementation

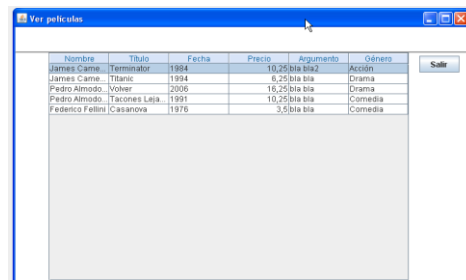
They are implementation bridges between:

- Data stored in objects
- Data stored in a relational DB
- Having methods to add, update, search and remove records
- Encapsulating the necessary logic to copy data values from classes of the problem domain (business



DAO Pattern Structure

Graphical structure of the **DAO pattern (Data Access Objects)**



Nombre	Título	Fecha	Precio	Argumento	Género
James Camé	Terminator	1984	10,25 bilión	Acción	
James Camé	Titanic	1994	6,25 bilión	Drama	
Pedro Almodó	Volver	2006	16,25 bilión	Drama	
Pedro Almodó	Tacones Lejía	1991	10,25 bilión	Comedia	
Federico Fellini	Casanova	1976	3,5 bilión	Comedia	

Presentation
Layer

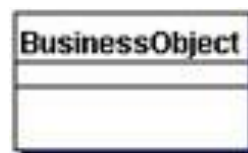
Domain Classes

Business
Layer

DAO Objects

Data Access
Layer

Database

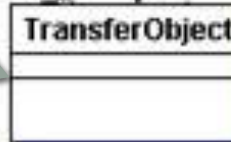


uses

encapsulates

obtains/modifies

creates/uses



Data access classes in the implementation.

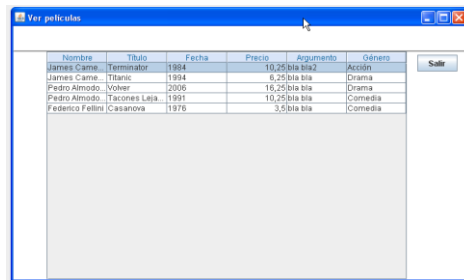
DAO

Structure of the DAO pattern. **Elements**

- **BusinessObject**: object of the business layer that needs access to the data storage to read or write information
- **DataAccessObject**: abstraction of the implementation of the data access layer. BusinessObject delegates the DAO all read/write operations
- **DataTransferObject (DTO)**: represents an object holding data. DAO may return data to BusinessObject by means of a DTO. The DAO may receive data in a DTO to update the DB
- **DataSource**: implementation of the data source (RDBMS, OODBMS, XML repository, raw files, etc.)

DAO Pattern Structure

Graphical structure of the **DAO pattern (Data Access Objects)**



Nombre	Título	Fecha	Precio	Argumento	Género
James Camé	Terminator	1984	10,25 bilia	Acción	
James Camé	Titanic	1994	6,25 bilia	Drama	
Pedro Almodó	Volver	2006	16,25 bilia	Drama	
Pedro Almodó	Tacones Leija	1991	10,25 bilia	Comedia	
Federico Fellini	Casanova	1976	3,5 bilia	Comedia	

Presentation
Layer

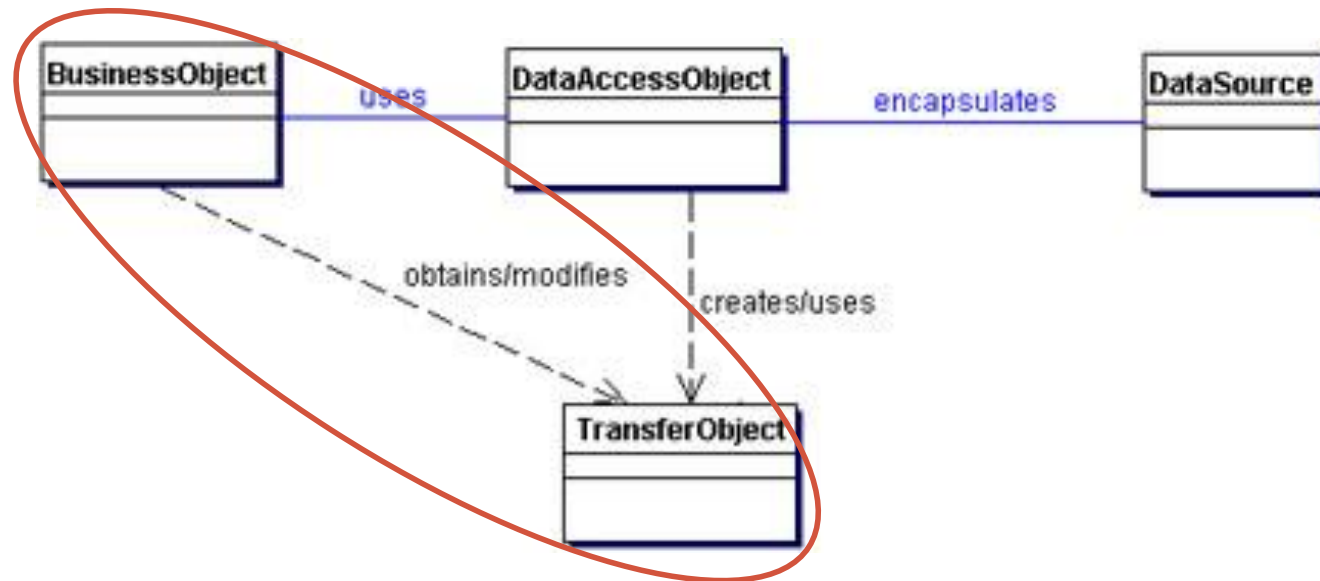
Domain Classes

Business
Layer

DAO Objects

Data Access
Layer

Database



For simplicity, these two classes may become just one which will be the **BusinessObject** (the **TransferObject** class is removed)

Data access classes in the implementation.

DAO

DAO pattern. **Advantages:**

- **Encapsulation.** Objects of the business layer do not know specific details of the implementation of the data access (hidden in the DAO).
- **Easier migration:** migrating to a different DBMS just involves changing the DAO layer.
- **Less complexity** in the business layer because the access to data is isolated.
- Data access **centralized** in a layer.

Data access classes in the implementation.

DAO

DAO pattern. **Disadvantages:**

- Sw architecture slightly more complex
- Additional code for the layer must be developed
- From an efficiency perspective the process may be slower

Implementation

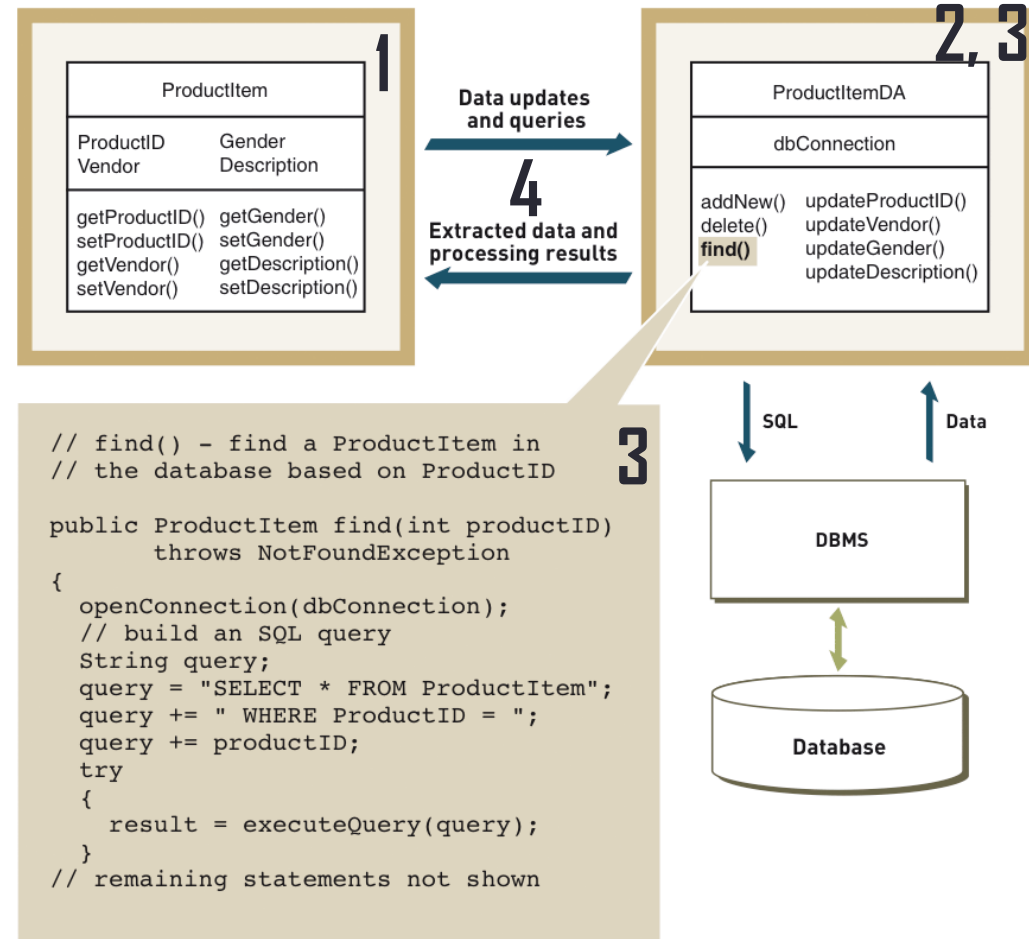
Step by step DAO pattern implementation

1. Take as starting point the Business Logic classes
2. Define an interface for each DAO,
 - A DAO interface **for each domain class** with CRUD operations (Create, Read, Update, Delete) and any other needed operations
3. Define a class for each interface implementing its functionality
 - This class will know the details about how to access the data (e.g. SQL statements)

Implementation

Step by step DAO pattern implementation

4. The Business Logic layer
Communicates with
DAO created at step



Example

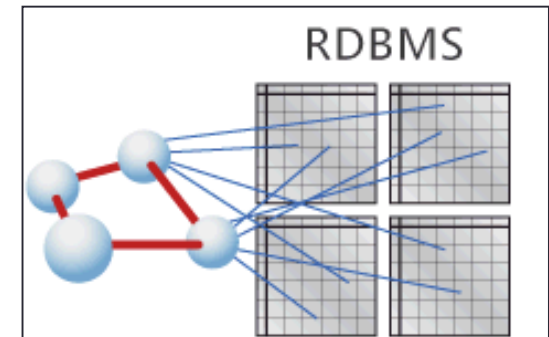
[Systems Analysis and Design in a
Changing World, 4th Edition]

OBJECT DESIGN

- ✓ Goals
- ✓ An example. Caché Intersystems

Object Design

- In complex system it is tedious to convert data between OO and Relational models
 - Mapping features from programming language to SQL and viceversa
- There are several tools that perform automatically this mapping for several languages (Java, C#, VB...)
 - Ex. Hibernate, **Entity Framework**



Object Design. Goals

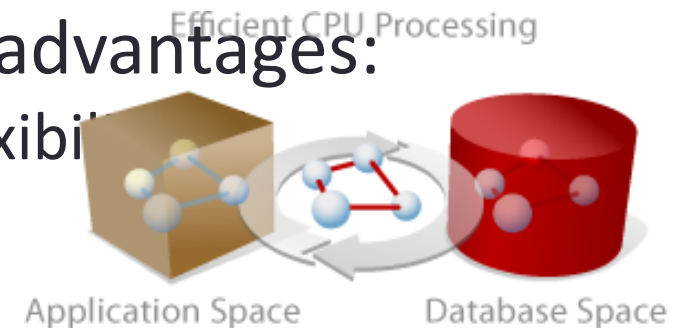
Instead of implementing a relational DB an OO DB is provided

- The internal storage represents objects as such (no dispersion in tables)
- No object-relational or object-SQL middleware
- Most operations are implemented in a more efficient way, no need to manage data in different relational tables
 - Ex., when a relationship is accesed there is no foreign key to retrieve the record of another table but we have the referenced object itself

Object Design. Advantages

There are important associated advantages:

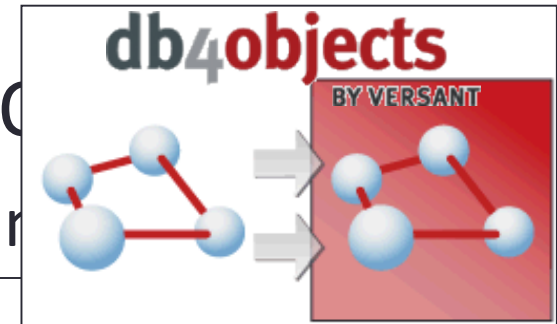
- The power of objects with the flexibility of query languages all together
- The development of layered applications is simplified
 - No need for a persistence layer based on SQL and a business logic based on objects (no need for constructing the model twice)
 - The logic layer communicates with the persistence layer by means of objects without conversion/mapping
 - Costant access in terms of “object.member”



Object Design. Additional systems

Open Source Object Database (OODB)

- db4objects (www.db4o.com) with r



```

1 public void store(Car car){
2     ObjectContainer db = Db4o.openFile("car.yap");
3     car.engine(new TurboEngine());
4     db.set(car);
5     db.commit();
6     db.close();
7 }

```

- ObjectStore (www.oracle.com/technology/products/objectstore/)

- Objectivity

- Orient (www.orientdb.com)

```

// OPEN THE DATABASE
d_Database db;
db.open( "business" );

d_Transaction tx;
tx.begin();

d_Ref obj;
obj = new( &db, "Customer" ) Customer();

obj->name = "Luke";
obj->surname = "Skywalker";

// INSERT THE OBJECT AS "MYFRIEND"
db.set_object_name( obj, "MyFriend" );

tx.commit();

```

```

// OPEN THE DATABASE
d_Database db;
db.open( "business" );

d_Transaction tx;
tx.begin();

d_Ref obj;

// RETRIEVE THE ENTRY CALLED "MYFRIEND"
obj = db.lookup_object( "MyFriend" );

// DISPLAY THE CUSTOMER NAME
cout << "MyFriend is: " << obj->name;

tx.commit();

```



En C++

Object Design. Additional free OODBMS

- **Goods**, Generic OO Database System

www.garret.ru/goods.html



- **JDOInstruments**, embedded OO database for Java Data Objects

sourceforge.net/projects/jdoinstr



- **Ozone**, Java based OO database management system

java-source.net/

source/database-engines/ozone



CONCLUSIONS

To sum up

- It is possible to apply a simple mapping to derive the relational model from a class diagram
 - In some cases several models are possible
- OODBs simplify the development of applications because:
 - The data model is built once and may be projected from/to the application without mappings
 - Operations and data management is simple, just operations on objects and their relationships
 - In general they are simple and efficient