

# Database Access and Patterns in Erlang/OTP

**Laura M. Castro**

MADS Research Group

Department of Computer Science

University of A Coruña (Spain)

**WSEAS AIC'08**

# Outline

- 1 Motivation
- 2 External data sources
- 3 Database access patterns
- 4 Implementation in Erlang/OTP
- 5 Conclusions

# Motivation

- Databases are an essential component in software systems
- There are multiple well-known (relational) DBMSs
  - ▶ Oracle, DB2, MS-SQL Server, PostgreSQL, MySQL,...
- Access to DBMS is well known from the most popular environments
  - ▶ C, C++, Java, .NET, Python, PHP...

# Motivation

But technology is always evolving, what about...?

- Erlang/OTP
  - ▶ a functional language and a set of libraries
  - ▶ created by Ericsson Telecommunications in 1990
  - ▶ a perfectly valid environment to develop almost any kind of application
  - ▶ strengths: concurrency, distribution, reliability
  - ▶ “the new Java”!

# External data sources

How do we access a DBMS?

**Standard API** Standardisation, independence, clean and reliable interaction

**Native API** Sacrifice independence for the sake of efficiency

**Sockets** The most efficient option in return for taking care of all  
low-level details

## External data sources

Access via a standard API: **ODBC**

- ODBC-compatible DBMS provide an implementation of this API
- Translates ODBC calls into the DB native dialect
- An ODBC driver is also needed for the client application to link to
- Client only performing standard function calls, makes it independent from the driver as well

# Database access patterns

Apart from the physical access, how do we interact with a DB?

4 steps:

- ▶ Identification of the persistent business objects
- ▶ Identification of the persistence operations
- ▶ Control flow definition
- ▶ Functionality or service API implementation

# Database access patterns

## A software pattern is

“a repeatable software solution to a particular kind of problem, which has proved to be both efficient and simple” (*Gang of Four*)

DB-related software patterns:

- Value Object pattern
- Data Access Object pattern (adaptor pattern + factory pattern)
- Facade pattern

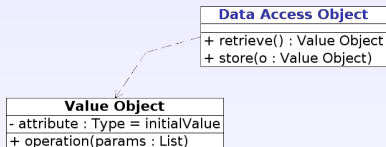
# Database access patterns

- Value Object

<b>Value Object</b>
- attribute : Type = initialValue
+ operation(params : List)

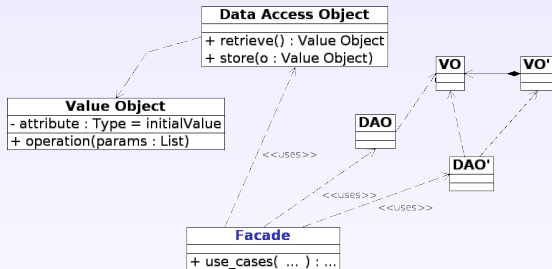
# Database access patterns

- Value Object
- Data Access Object



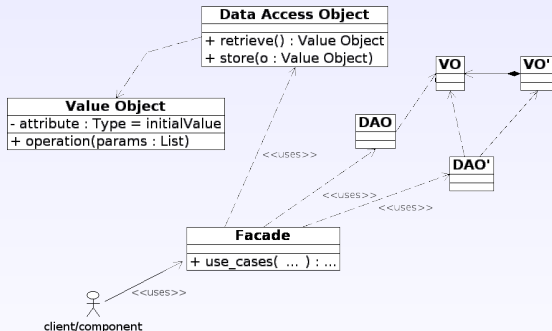
# Database access patterns

- Value Object
- Data Access Object
- Facade



# Database access patterns

- Value Object
- Data Access Object
- Facade



# Implementation in Erlang/OTP

## Why Erlang/OTP?

- Technology now is parallel and distributed
- Erlang/OTP is a framework develop robust, highly available, distributed systems
  - ▶ Distribution and supervision capabilities
  - ▶ Concurrency based on asynchronous message-passing

# Implementation in Erlang/OTP

Challenges:

- Based on the functional programming paradigm
- Getting on well with traditional DBMS
- Exporting software patterns to a functional environment

# Implementation in Erlang/OTP

How well does Erlang/OTP get on with traditional DBMS?

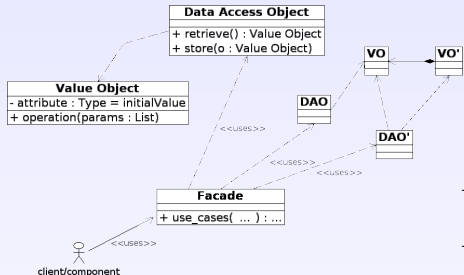
- Provides its own (distributed) DBMS: Mnesia
- Provides support for ODBC
- Some native drivers available (PostgreSQL, MySQL)

# Implementation in Erlang/OTP

How well do this DB patterns export to a functional environment?

- *Modules* are the implementation unit elements
  - ▶ correspondence *object* ↔ *module*
  - ▶ correspondence *object internal state* ↔ *record*
  - ▶ correspondence *object operations* ↔ *exported functions*
- Structural patterns: no inheritance, no polymorphism

# Implementation in Erlang/OTP



```
-module(value_object).  
-export([operation / 0, ...]).  
-record(object ,  
        {attribute , ...}).
```

operation(ValueObject) →

```
#object{ attribute = ValueObject#object.attribute },
```

...

# Conclusions

- Databases are almost omnipresent in software systems
  - ▶ Either created from scratch or preexistent/shared with other systems
- As technology changes, new tools appear and old good practices need to be updated/adapted
  - ▶ Take the most out of our resources
  - ▶ Do it in a reliable and maintainable way
- Erlang/OTP as a great example with a lot to say in the (near) future

Questions?

**Thanks!**