UNIVERSIDADE FEDERAL DE ITAJUBÁ

Banco de Dados II

COM 231

Persistência
Mapeamento Objeto-Relacional
Aula 9

Vanessa Cristina Oliveira de Souza

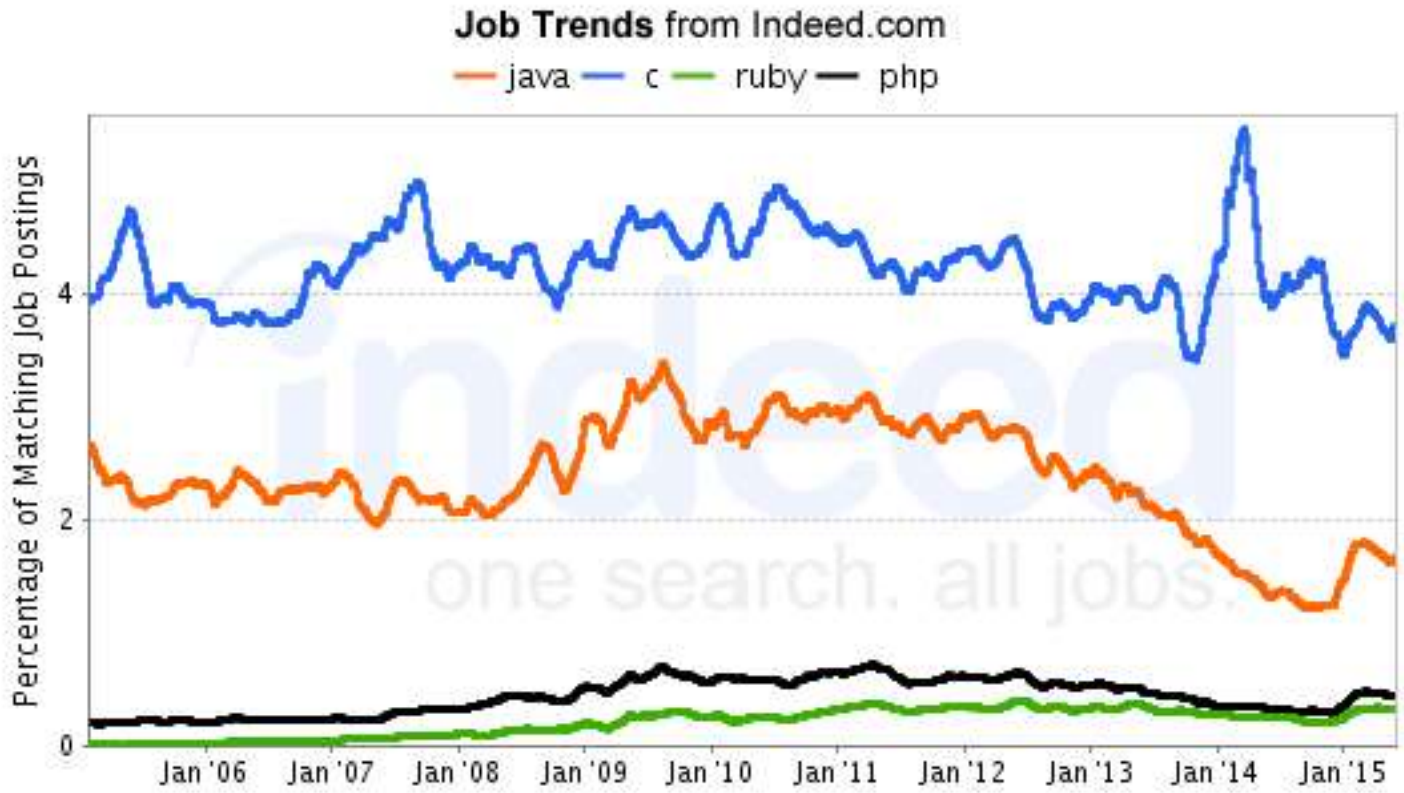
Crescimento das linguagens de programação

Apr 2016	Apr 2015	Change	Programming Language
1	1		Java
2	2		C
3	3		C++
4	5	^	C#
5	8	^	Python
6	7	^	PHP
7	6	v	JavaScript
8	12	^^	Perl
9	18	^^	Ruby
10	10		Visual Basic .NET

Fonte : TIOBE – Abril/2016

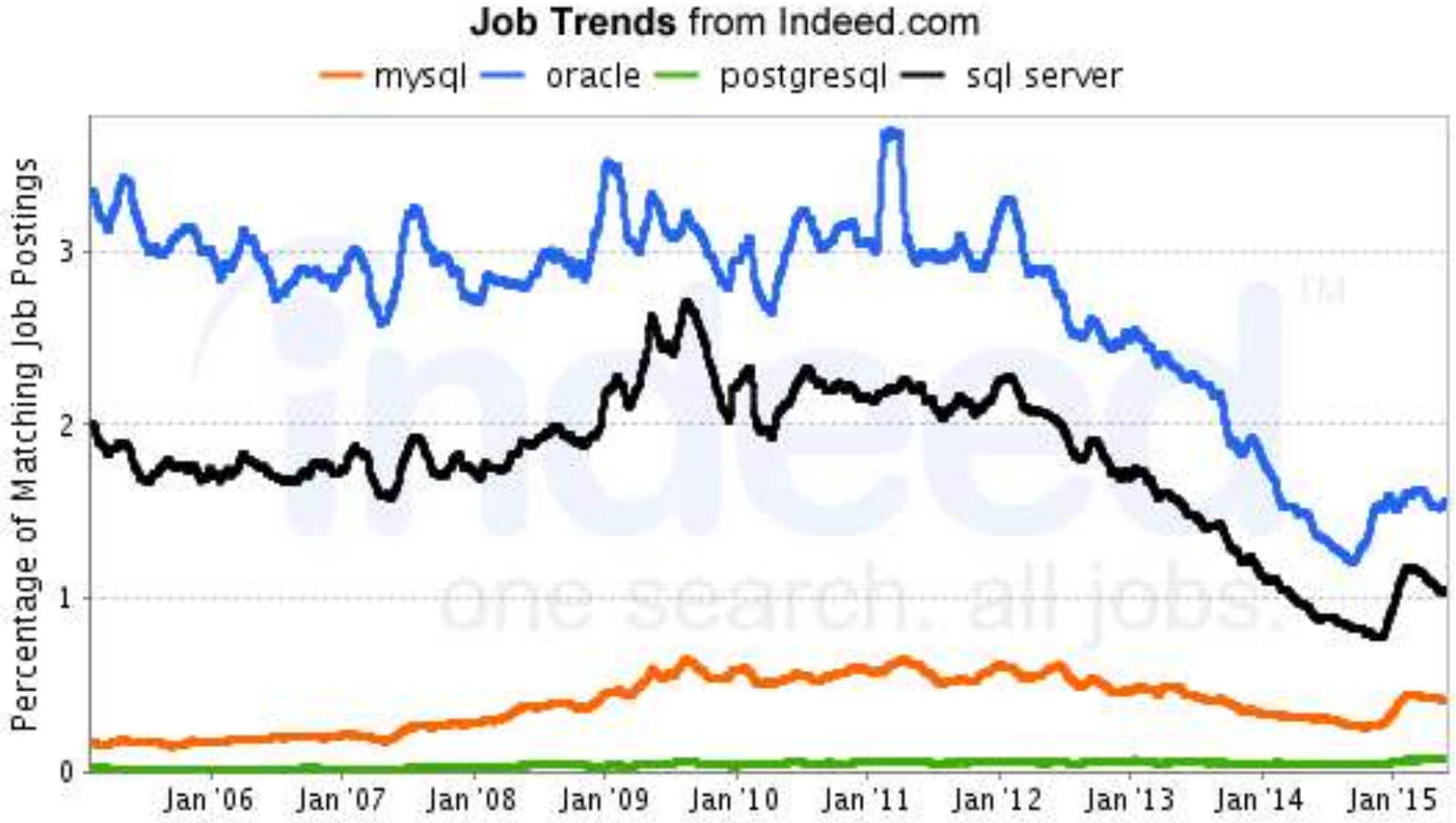
Crescimento das linguagens de programação

Percentual de buscas de postagens de empregos



Fonte : Indeed

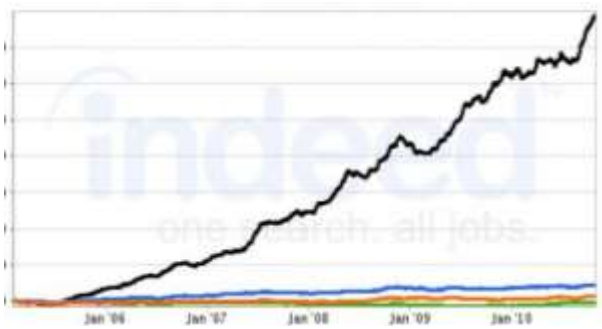
Crescimento das linguagens de programação



Fonte : Indeed

Introdução

Crescimento das linguagens OO



Uso consolidado dos SGBDR

Introdução

COMO PERSISTIR DADOS DE UMA APLICAÇÃO OO EM UM SGBDR??

UNIVERSIDADE FEDERAL DE ITAJUBÁ

PERSISTÊNCIA

Persistência de Dados

- Persistência não é simplesmente armazenar o dado:
  - Armazenamento, organização e recuperação
  - Concorrência e integridade
  - Compartilhamento

- Portanto, a Persistência de Dados consiste no armazenamento confiável e coerente das informações em um sistema de armazenamento de dados.

Persistência

- Em uma aplicação OO, a persistência permite um objeto sobreviver ao processo que o criou.
- O estado do objeto pode ser guardado no disco, e um objeto com um mesmo estado pode ser recriado no futuro.
- Quase todas as aplicações necessitam de dados persistentes.
- A persistência define como guardar dados em um banco de dados relacional usando SQL.

Persistência

- Objeto transiente x Objeto persistente

Persistência de Objetos

- Aplicações OO x SGBDR
  - Impedância entre modelos
  - OO é semanticamente mais rico que o MR
  - Várias formas de representar uma estrutura do OO no MR
  - Mapeamento Objeto-Relacional
    - Técnica desenvolvida para reduzir a impedância

Persistência de Objetos

- **Resenha do item 1.2 do livro**
  - Java Persistence com Hibernate
  - Bauer & King
  - 2007

UNIVERSIDADE FEDERAL DE ITAJUBÁ

ARQUITETURA EM CAMADAS

Arquitetura em Camadas

- O grande desafio das equipes de desenvolvimento de aplicações é cada vez mais produzir aplicativos seguros , eficientes, de fácil manutenção, reutilizáveis e em prazos cada vez menores.

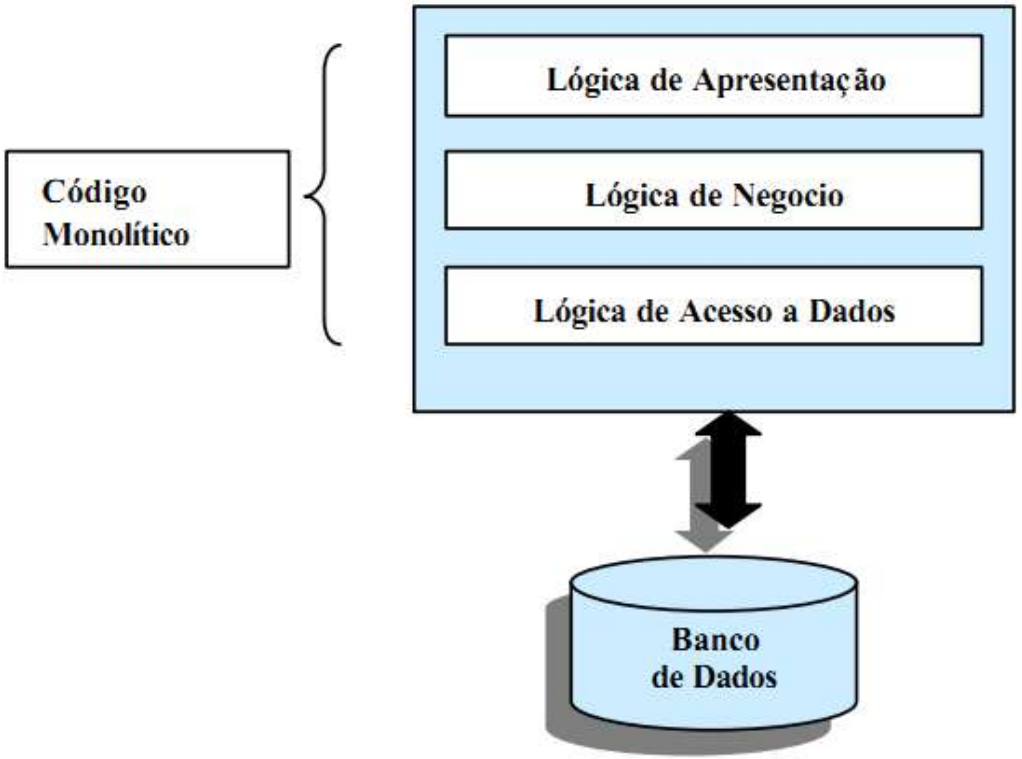
- A organização em camadas é a chave para a independência entre os componentes e esta independência é que vai atingir os objetivos de eficiência, escalabilidade, reutilização e facilidade de manutenção.

Arquitetura em Camadas

- Uma arquitetura em camadas define interfaces entre os códigos que implementam as várias funcionalidades, permitindo que mudanças sejam feitas de forma que uma funcionalidade seja implementada sem afetar significativamente o código de outras camadas.

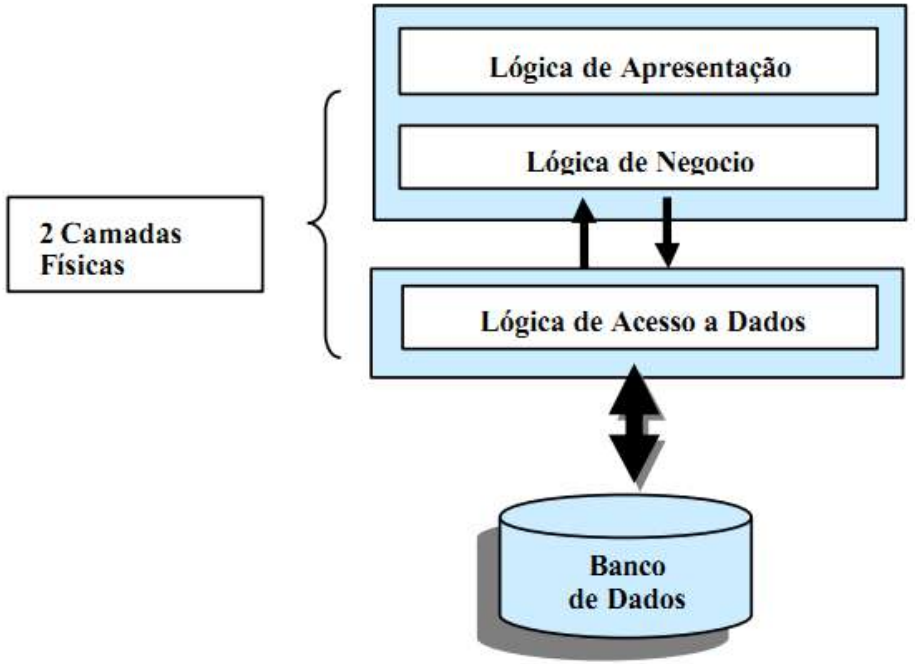
Arquitetura Monolítica

- Login do usuário, verificação, lógica de negócio e acesso a banco de dados em um único módulo.



Arquitetura em duas camadas

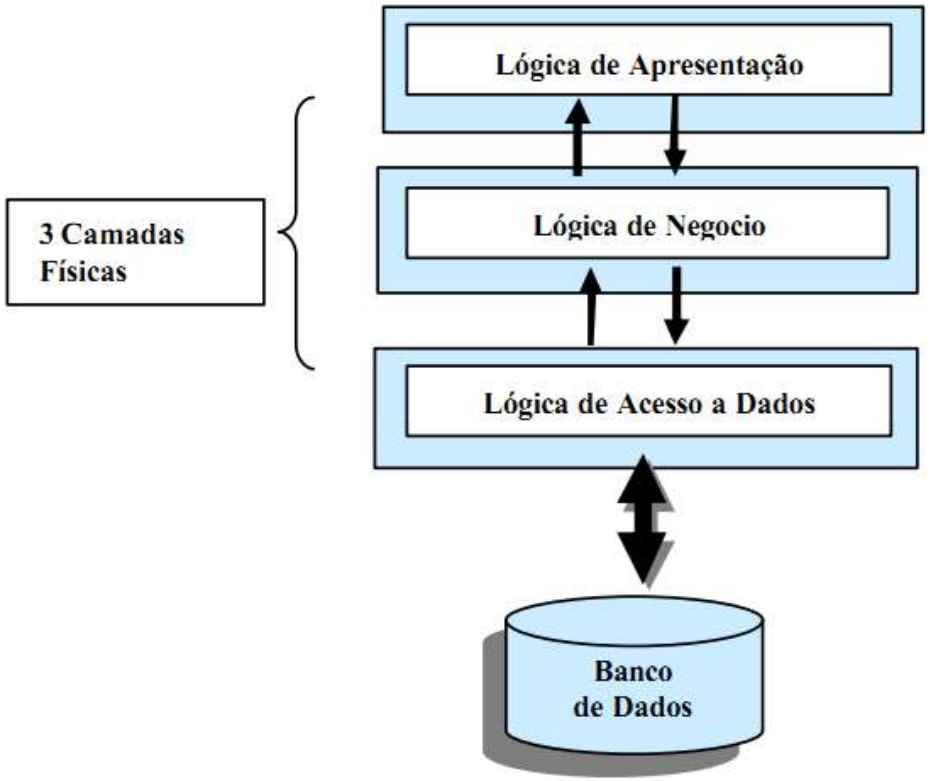
- Necessidade de compartilhar a lógica de acesso a dados entre vários usuários simultâneos fez surgir as aplicações em duas camadas.



Arquitetura em três camadas

- Com o advento da internet houve um movimento para separar a lógica de negócio da interface com o usuário.

- A arquitetura em 3 camadas, envolve a separação das funcionalidades usando camadas, com o objetivo de separar a lógica de apresentação, a lógica de negocio e a conexão com o banco de dados (lógica de acesso a dados).

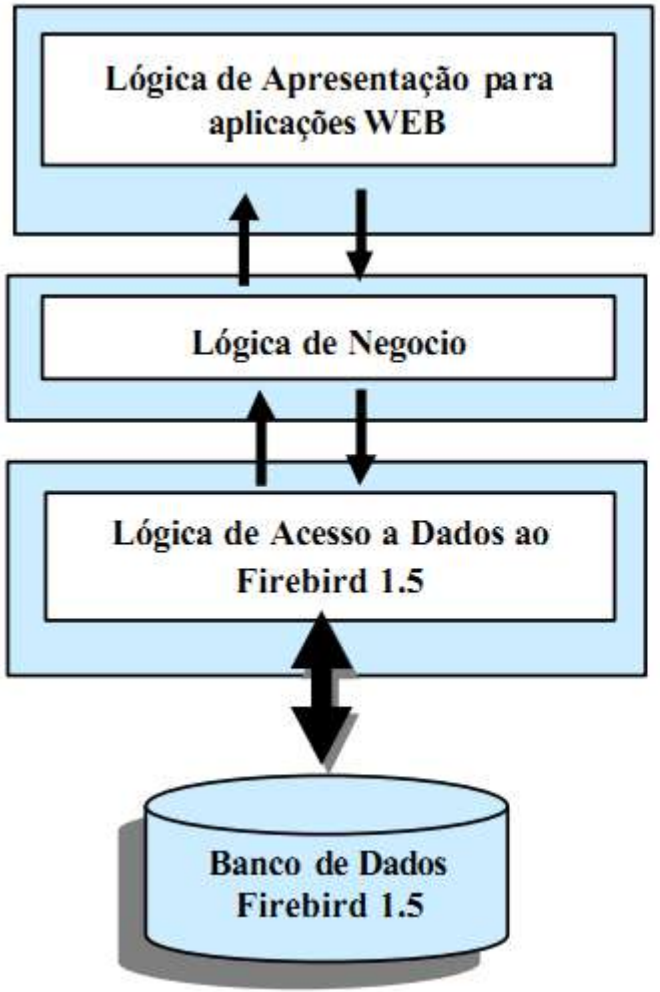


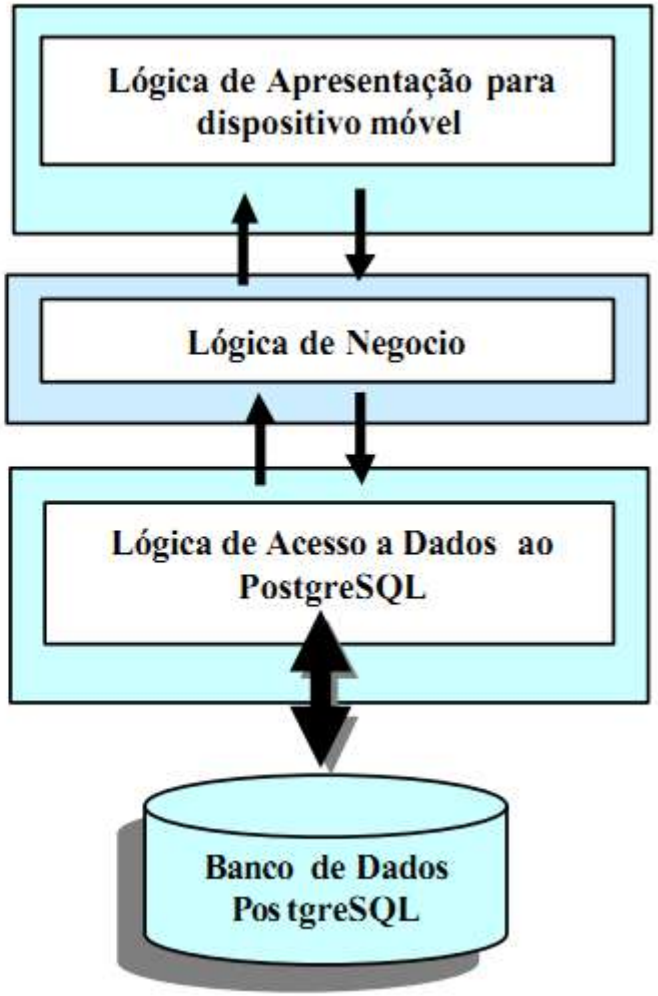
Arquitetura em três camadas

- A separação em três camadas torna o sistema mais flexível, de modo que partes podem ser alteradas independentemente.

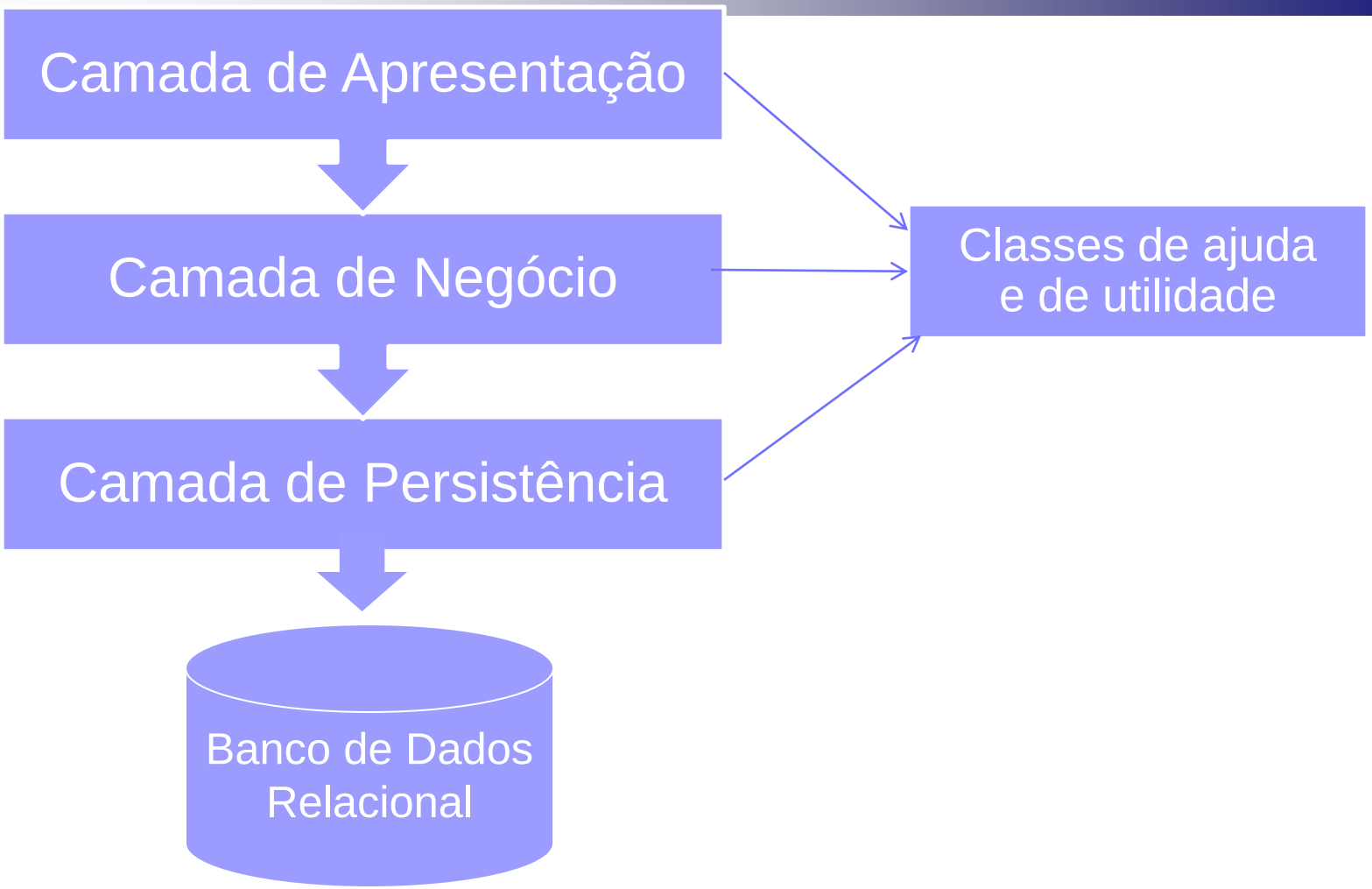
- Com o emprego de arquitetura em três camdas, qualquer alteração em uma determinada camada não influi (ou pouco influi) nas demais, desde que o mecanismo de comunicação entre elas permaneça inalterado.

Arquitetura em três camadas





Arquitetura em três camadas



Como persistir dados em Java?

- Codificar a mão com SQL/JDBC
- Serialização
- SGBDOO
- XML
- Mapeamento Objeto/Relacional

Como persistir dados em Java?

- Codificar a mão com SQL/JDBC
- Serialização
- SGBDOO
- XML
- Mapeamento Objeto/Relacional

UNIVERSIDADE FEDERAL DE ITAJUBÁ

MAPEAMENTO OBJETO/RELACIONAL ORM

ORM

- Mapeamento objeto/relacional é a **persistência automatizada** e transparente dos objetos em uma aplicação Java para as tabelas de um banco de dados relacional, utilizando **metadados** que descrevem o mapeamento entre os objetos e o banco de dados.

ORM

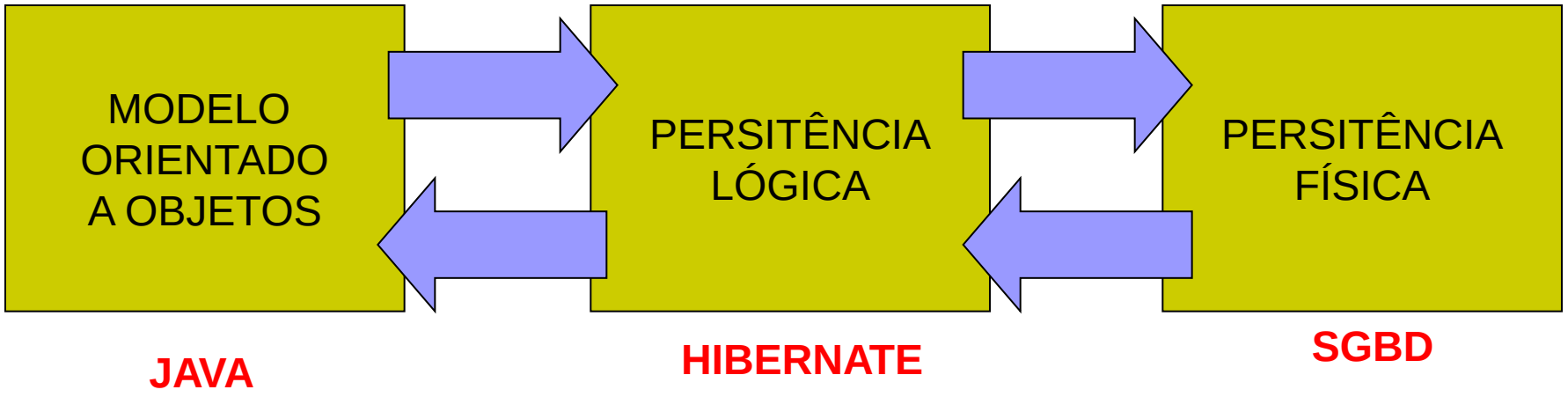
- O Mapeamento OR é uma técnica de desenvolvimento que consiste em representar o objeto de maneira relacional na gravação do banco de dados, e consegue fazer o caminho inverso sem perder informação.

Diagrama ORM

ORM

- O mapeamento OR tem 3 componentes:
  - *Modelo Orientado a objetos*
    - Modelo OO em que os dados estão representados na aplicação
  - *Persistência Lógica*
    - traduz o modelo OO para a maneira que eles serão armazenados na persistência física, e vice-versa.
  - *Persistência Física*
    - Modelo relacional em que os dados serão armazenados.

Diagrama ORM



ORM

- Uma solução ORM consiste nas seguintes 4 partes:
  - Uma API para realizar operações CRUD básicas em objetos de classes persistentes
  - Uma linguagem ou API para especificar consultas que se referem às classes ou às propriedades das classes
  - Uma facilidade para especificar o metadado de mapeamento
  - Uma técnica para que a implementação ORM interaja com objetos transacionais para executar funções de otimização

UNIVERSIDADE FEDERAL DE ITAJUBÁ

HIBERNATE

Hibernate

- O Hibernate é uma das soluções mais difundidas para Mapeamento OR em Java. (Outras também famosas são OJB, JDO e o Toplink).
- Ele está sob a licença GPL.
- Usam o Hibernate em seus projetos: Sony, AT&T, PwC, Cisco... entre outras empresas.

Hibernate

- O hibernate isola o aplicativo do contato direto com o banco de dados, servindo como ponte entre os dois sistemas.
- Não há chamadas SQL misturadas ao código Java.
- Ao contrário de outros modelos de mapeamento OR, o hibernate não é intrusivo. Ou seja, ele não obriga o programador a estender uma classe dele no seu aplicativo.

Hibernate

- Desenvolvimentos:
  - Top Down
    - Modelo de domínio -> MR
  - Botton Up
    - MR -> Modelo de domínio

Hibernate

- Banco Northwind
  - Acrescentar foreign key na tabela order_details
    - ALTER TABLE northwind.order_details ADD CONSTRAINT product_id FOREIGN KEY (productid) REFERENCES northwind.products (productid) MATCH SIMPLE ON UPDATE CASCADE ON DELETE RESTRICT;
  - Acrescentar foreign key na tabela order
    - FOREIGN KEY (customerid) REFERENCES northwind.customers (customerid) MATCH SIMPLE ON UPDATE CASCADE ON DELETE RESTRICT;
    - FOREIGN KEY (employeeid) REFERENCES northwind.employees (employeeid) MATCH SIMPLE ON UPDATE CASCADE ON DELETE RESTRICT

Hibernate

- Desenvolvimentos:
  - Top Down
    - Modelo de domínio -> MR
  - Botton Up
    - MR -> Modelo de domínio

Annotations

```
@Entity
@Table(name = "aluno")
public class Aluno {

   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;

   @notnull
   private int matricula;

   @notnull
   @Length(max = 100)
   private string nome;

   @Temporal(TemporalType.DATE)
   private Date dataNascimento;

   @ManyToOne(fetch = FetchType.LAZY)
   private Cidade cidade;

   @OneToMany(fetch = FetchType.LAZY, mappedBy = "aluno")
   private List<Documento> documentos;

   @ManyToMany(cascade = CascadeType.ALL)
   private List<Disciplina> disciplinas;
   .
   .
}
```

Fontes

- Neto et al., Desenvolvimento de sistema web utilizando arquitetura em três camadas e applets. Disponível em : http://inf.unisul.br/~ines/workcomp/cd/pdfs/2905.pdf. Acesso em maio/2011

- Padrões de Projeto : O modelo MVC - Model View Controller. Disponível em: http://www.macoratti.net/vbn_mvc.htm. Acesso em maio/2011