INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARAÍBA
Campus João Pessoa

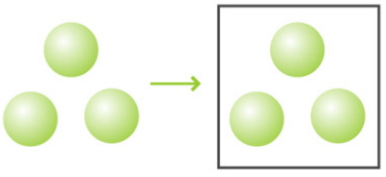
Persistência de Objetos

Fausto Maranhão Ayres

2
Persistência com db4o

Desenvolvimento ágil com BDOO

- BDOO evoluíram muito nos últimos anos e oferecem
  - Segurança
  - Escalabilidade
  - Performance
- Permite **desenvolvimento Ágil,** pois não requer mapeamento de objeto entre memória e banco



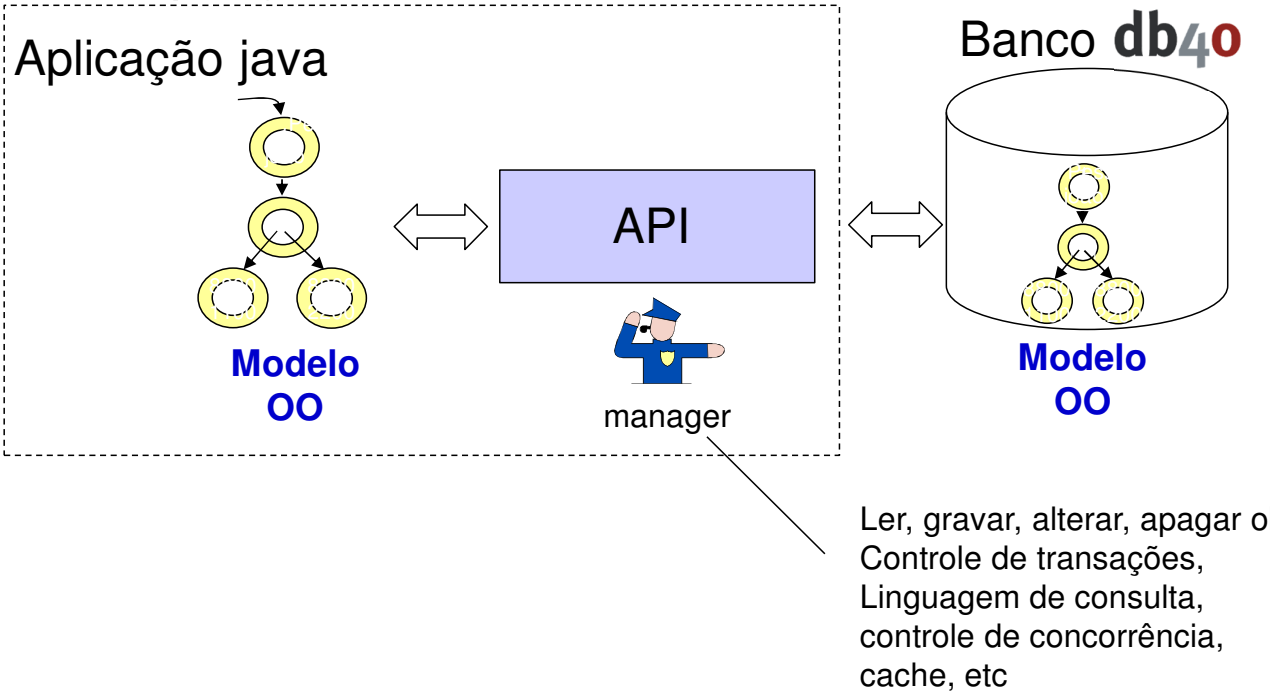
Objetos na memória (Java) *Objetos no Banco*

Mesmo modelo na memória e no banco

fausto.ayres@ifpb.edu.br

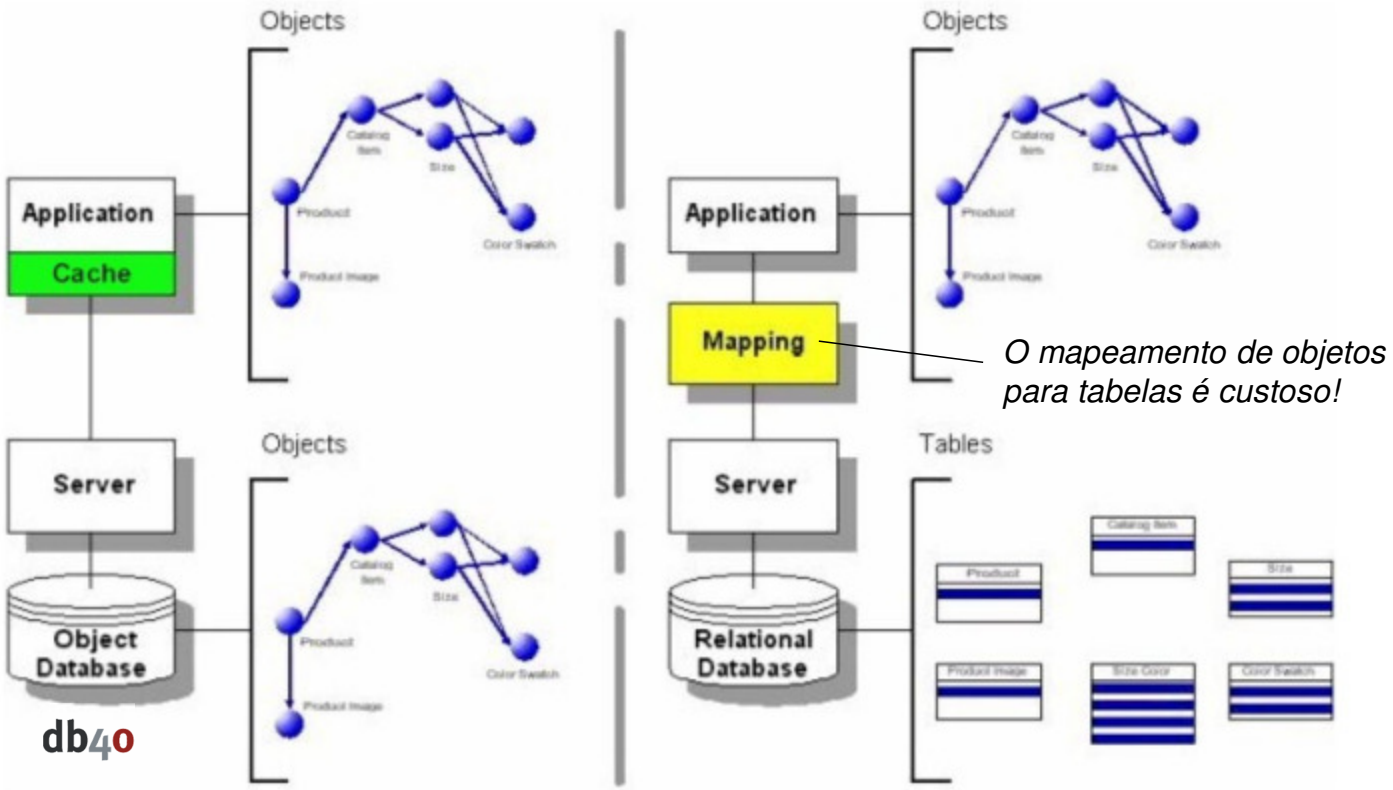
Banco de dados **db4o** (*database for objects*)

- Oferece **persistência transparente** (pouco esforço de programação)



fausto.ayres@ifpb.edu.br

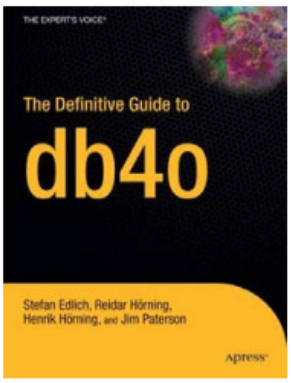
Comparação db4o x Relacional



Sobre db4o

- 100% embarcado (zero administração)
- Pode ser usado no:


- Acesso Cliente/Servidor
- Gratuito (GPL)



Nasceu em 2001 (empresa db4objects)
Foi vendido para Versant e depois para Actian
http://supportservices-old.actian.com/versant/default.html

Casos de Sucesso







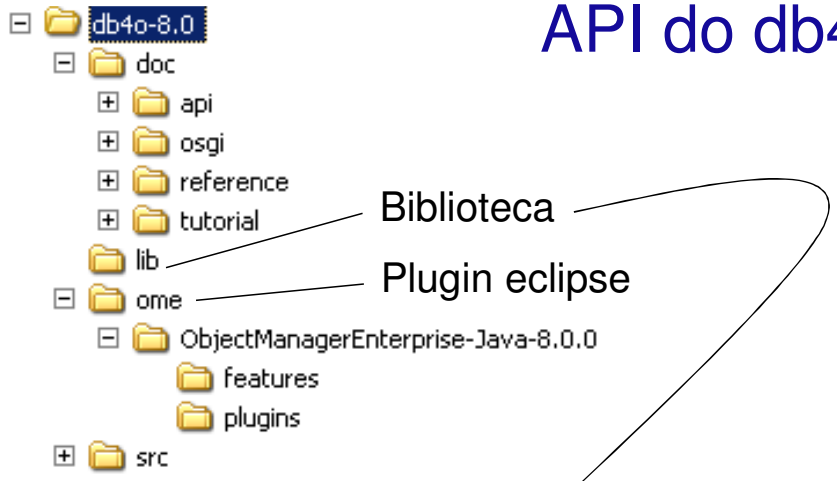
Principais características do db4o

- Necessita apenas 400k de RAM (android)
- Grava até 200.000 objetos/seg
- Garante as propriedades ACID
- Linguagem de consulta cliente/servidor
- Segurança (criptografia)
- Arquivo de até 254G
- Otimização de queries
- Índices

Desempenho

- Benchmark http://www.polepos.org/

API do db4o



ant.jar	ant build tool, used to build db4o sources (http://ant.apache.org/)
bloat-1.0.jar	bytecode optimization library, used to optmize Native Queries and Transparent Activation
db4o-8.0.224.15975-all-java5.jar	no-dependency jar with all db4o binaries for JDK 5 and JDK 6
db4o-8.0.224.15975-bench.jar	IO benchmark library
db4o-8.0.224.15975-core-java5.jar	db4o database engine core for JDK 5 and JDK 6
db4o-8.0.224.15975-cs-java5.jar	db4o client/server components for JDK 5 and JDK 6
db4o-8.0.224.15975-db4ounit-java1.5.jar	testing framework for db4o needs for JDK 5 and JDK 6
db4o-8.0.224.15975-instrumentation.jar	Instrumentation layer on top of bloat
db4o-8.0.224.15975-nqopt.jar	Native Query optimization library. This library should be available i classpath for NQ optimization
db4o-8.0.224.15975-optional-java5.jar	db4o optional components for JDK 5 and JDK 6
db4o-8.0.224.15975-osgi.jar	db4o for OSGi. See OSGi API documentation.
db4o-8.0.224.15975-osgi-test.jar	db4o OSGI tests
db4o-8.0.224.15975-taj.jar	db4o instrumentation classes for Transparent Activation
db4o-8.0.224.15975-tools.jar	db4o tools user interface, contains db4o enhancer and Ant tasks

API DB4O

API db4o

```
import com.db4o.*;

// configuração do banco
EmbeddedConfiguration conf = Db4oEmbedded.newConfiguration();
conf.common().messageLevel(0);  // 0,1,2,3


// instanciação do manager
ObjectContainer manager = Db4oEmbedded.openFile(conf,"banco.db4o");


manager.store(objeto)       //grava, atualiza
manager.delete(objeto)      //apaga
manager.query(...)          //consulta
manager.commit()            //efetiva
manager.rollback()          //cancela
manager.close()             //fecha banco
```

Exemplo base da disciplina

Relacionamento 1:* bidirecional

```
public class Pessoa {

    private int id;
    private String nome;
    private List<Telefone> telefones = new ArrayList<Telefone>();

public class Telefone {

    private String numero;
    private Pessoa pessoa;
```

Gravar um objeto

```
Pessoa p = new Pessoa("joao");

manager.store(p);
manager.commit();
```

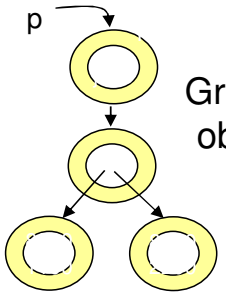
Inicia a transação.
Marca o objeto para gravação

Efetiva a transação
Grava o objeto

Gravar um grafo de objetos (cascata)

```
Pessoa p = new Pessoa("joao");
p.adicionar(new Telefone ("8800-0000"));
p.adicionar(new Telefone ("8800-1100"));

manager.store(p);
manager.commit();
```

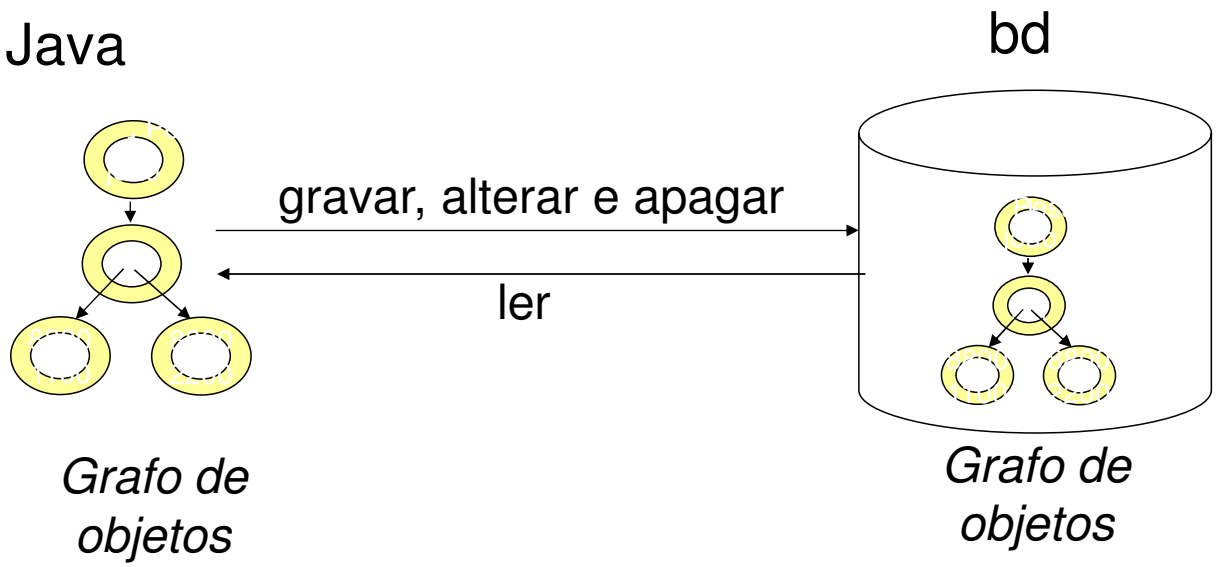


Grafo de objetos

Inicia a transação.
Marca os objetos para gravação

Grava todos os objetos

Operações em cascata



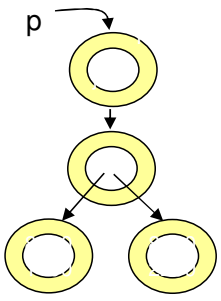
Ler um grafo de objetos (cascata)

- Através de uma Query

```
//consulta
Query q = manager.query();
q.constrain(Pessoa.class);
q.descend("nome").constrain("joao");
List<Pessoa> resultados = q.execute();

if(resultados.size()>0) {
   Pessoa p = resultados.get(0);      //ler
   ...
   System.out.println(p.getTelefones());
}
else
   System.out.println("joao inexistente");
```

Executa a query e retorna uma lista de objetos



Apagar um grafo de objetos (cascata)

```
Query q = manager.query();
q.constrain(Pessoa.class);
q.descend("nome").constrain("joao");
List<Pessoa> resultados = q.execute();


if(resultados.size()>0) {
   Pessoa p = resultados.get(0);      //ler
   manager.delete(p);
   manager.commit();
}
else
   System.out.println("inexistente");
```

Executa a query e retorna uma lista de objetos

apaga o grafo de objetos em cascata

Alterar um grafo de objetos (cascata)

- Exemplo1: Adicionar um telefone a pessoa

```
Query q = manager.query();
q.constrain(Pessoa.class);
q.descend("nome").constrain("joao");
List<Pessoa> resultados = q.execute();

if(resultados.size()>0) {
   Pessoa p = resultados.get(0);
   Telefone t = new Telefone("9999-9999");
   p.adicionar(t);          //adicionar telefone a pessoa
   t.setPessoa(p);          //adicionar pessoa ao telefone
   manager.store(p);
   manager.commit();
}
```

Atualiza os objetos em cascata

Alterar grafo de objetos (cascata)

- Exemplo2: Remover um telefone da pessoa

```
Query q = manager.query();
q.constrain(Pessoa.class);
q.descend("nome").constrain("joao");
List<Pessoa> resultados = q.execute();

if(resultados.size()>0) {
   Pessoa p = resultados.get(0);
   Telefone t = p.localizar("8800-1100");
   p.remover(t);          //remover telefone da pessoa
   t.setPessoa(null);     //remover pessoa do telefone
   manager.store(p);
   manager.store(t);
   manager.delete(t);
   manager.commit();
}
```

Atualização dos objetos no banco

é conveniente deletar o telefone **órfão**

Conceito de objeto órfão

- É um objeto que está *sem relacionamento*
- Pode ser **mantido** no banco ou **apagado**, dependendo da regra de negócio do sistema

CONFIGURAÇÃO DO BANCO

Configuração das operações em cascata

- Antes de abrir o banco

```
EmbeddedConfiguration conf = Db4oEmbedded.newConfiguration();
conf.common().objectClass(Pessoa.class).cascadeOnDelete(true);
conf.common().objectClass(Pessoa.class).cascadeOnUpdate(true);
conf.common().objectClass(Pessoa.class).cascadeOnActivate(true);
conf.common().objectClass(Telefone.class).cascadeOnDelete(true);
conf.common().objectClass(Telefone.class).cascadeOnUpdate(true);
conf.common().objectClass(Telefone.class).cascadeOnActivate(true);


//abrir o banco
ObjectContainer manager = Db4oEmbedded.openFile(conf,"banco.db4o");
```

Criar índices

- Índices permitem a otimização de consultas pelo banco

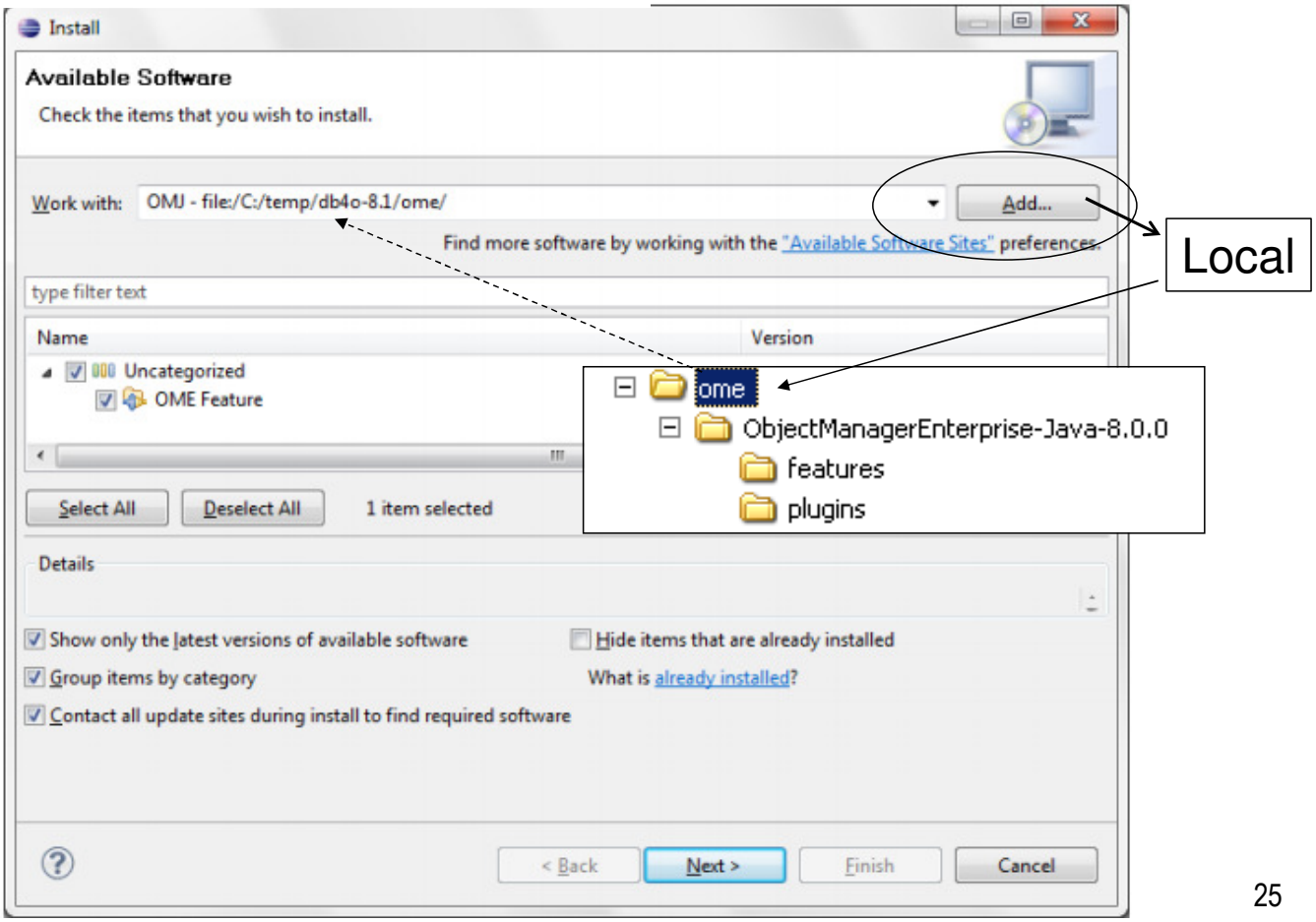
  Ex: indexar o atributo nome

```
...
conf.common().objectClass(Pessoa.class)
                    .objectField("nome").indexed(true);
```


VISUALIZAÇÃO DO BANCO PLUGIN O.M.E.

Instalação do plugin OME- Object Manager Enterprise

Help / install new software



perspectiva OME