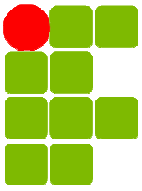


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARAÍBA
Campus João Pessoa

Persistência de Objetos

Fausto Maranhão Ayres

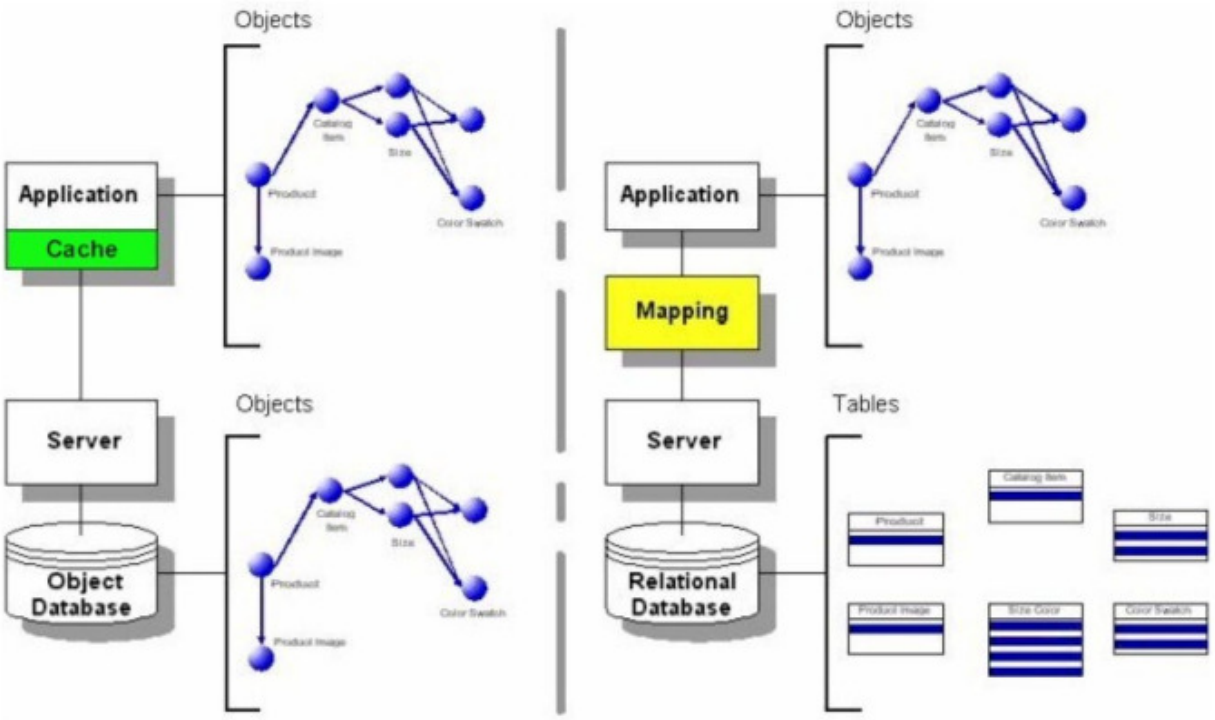
2 Persistência com DB4o (Básico)

Banco de Dados Orientado a Objetos

- Alguns BDOO evoluíram muito nos últimos anos, e atingiram segurança, escalabilidade e performance
- Desenvolvimento ágil
  - Menos tempo de desenvolvimento devido a um único modelo de dados (OO)
- Excelente performance
  - Não há perda de tempo com a conversão de objetos para tabelas

fausto.ayres@ifpb.edu.br

Comparação com BD Relacional



DB4O (Database For Object)

- Nasceu em **2001**
- Nativo para Java, .NET, Android, JRuby
- Gratuito (GPL)
  - Plug-in para o Eclipse (OME)
- Embarcado:
  - Não necessita de administração
  - Arquivos .jar (plataforma java) ou
  - Arquivos .dll (plataforma .net)
- Arquitetura Cliente/Servidor
- Atualização automática do esquema do bd

Casos de Sucesso

- sistema de controle de trens bala espanhol (AVE)






Características

- Principais recursos
  - Segurança (criptografia)
  - Propriedades **ACID**
  - Arquivo de até 254Gb
  - Otimização de consultas
  - Indexação de atributos para busca

- Desempenho
  - usa **400k** de RAM
  - armazena até **200.000** objetos/segundo
  - Ver tabela comparativa com outros SGBDs

Benchmark

- Exemplo: PolePosition

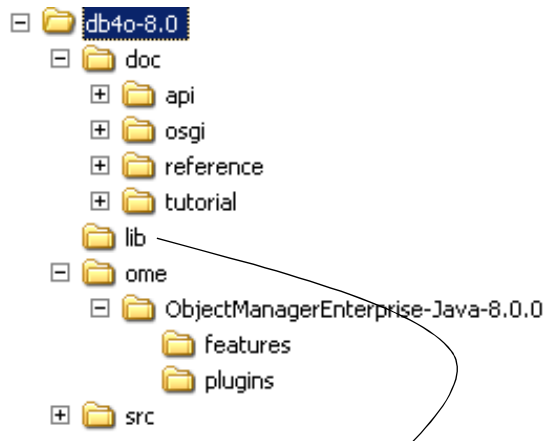
Barcelona Benchmarks	read	write	query	delete
Native/db4o 6.4	1.0	1.0	1.0	1.0
Hibernate/hsqldb	15.8	3.7	2,583.1	4.3
Hibernate/mysql	48.0	26.1	14.4	26.9
JDBC/MySQL	40.8	19.5	9.3	15.8
JDBC/JavaDB	27,843.7	20.5	47,993.1	17.7
JDBC/HSQLDB*	1.9	1.1	2,554.4	0.5
JDBC/SQLite	8.8	519.1	1.1	362.1

fastest

slowest

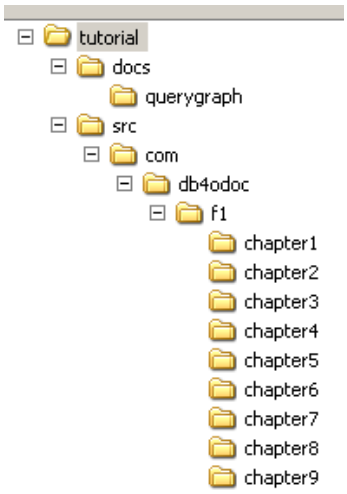
* JDBC/HSQLDB not ACID transaction safe

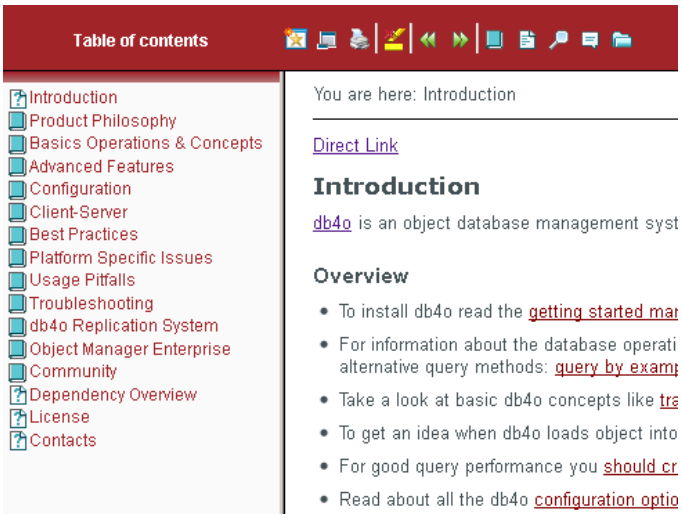
Bibliotecas



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

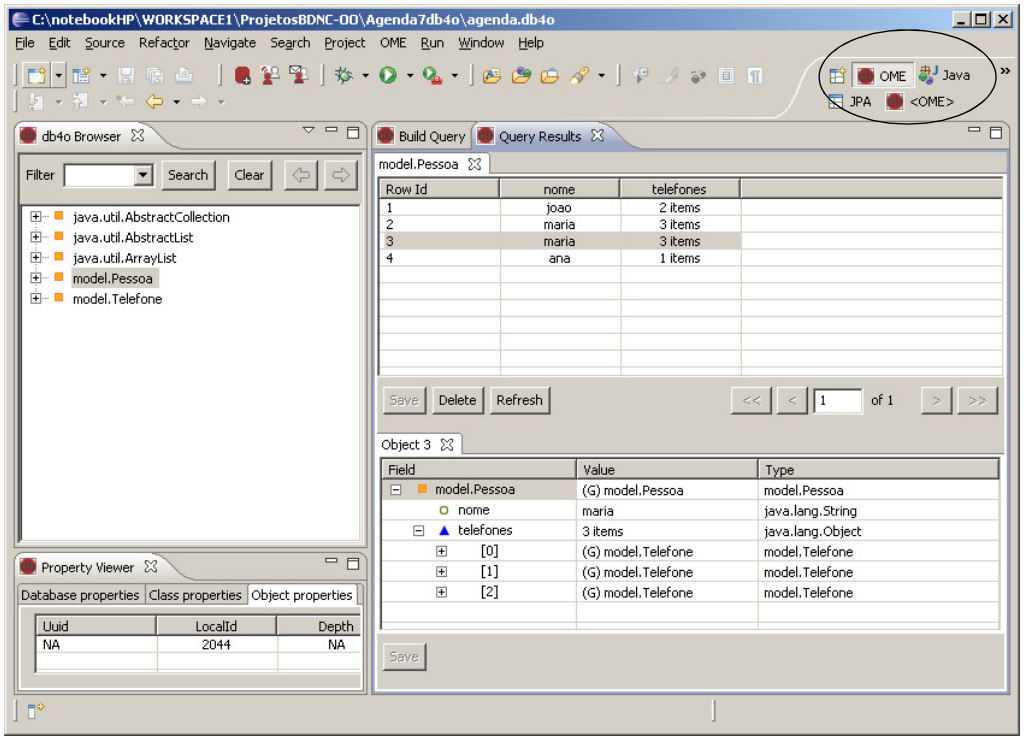
Tutoriais





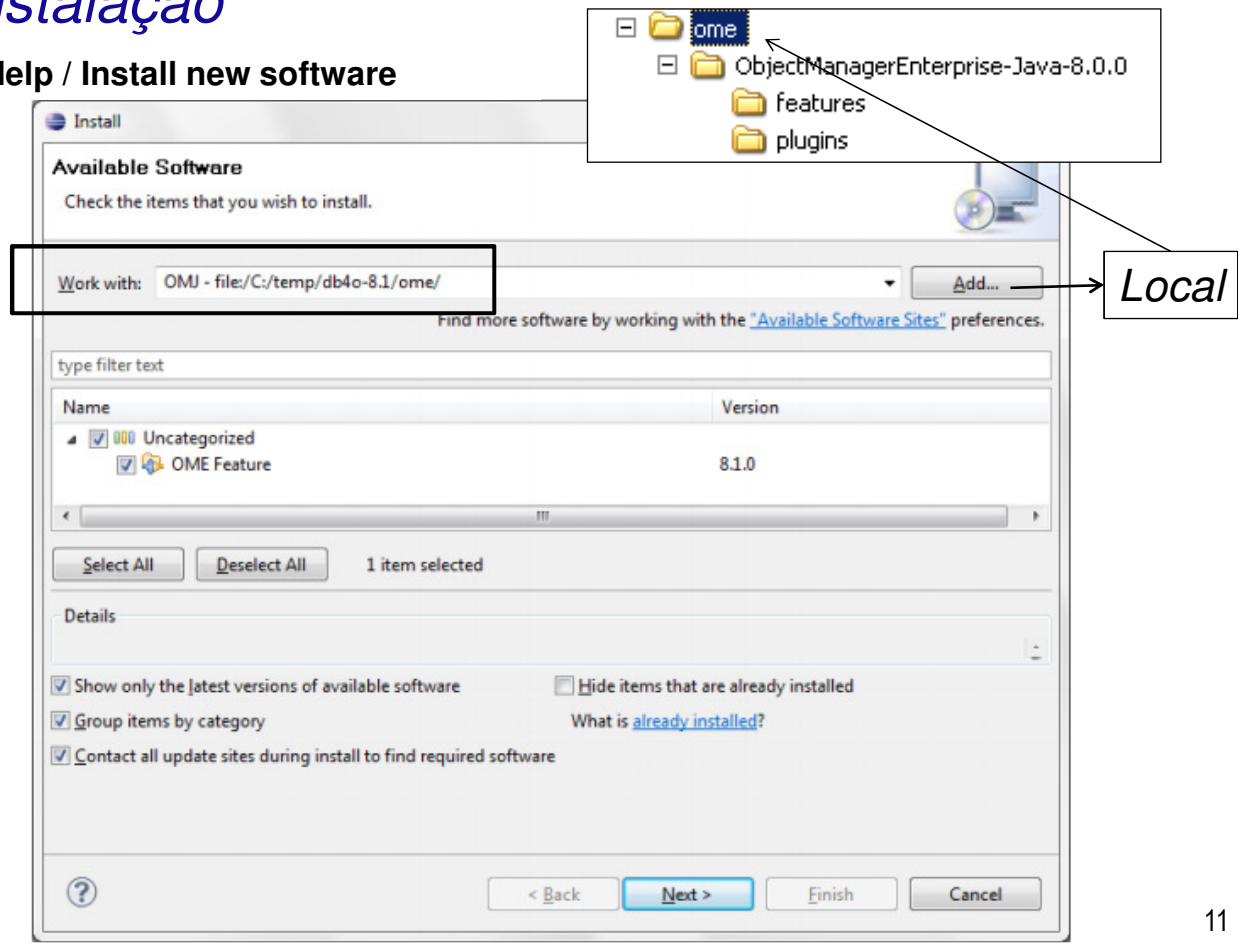
Plugin OME (Object Manager Enterprise)

- Visualiza o grafo de objetos no eclipse

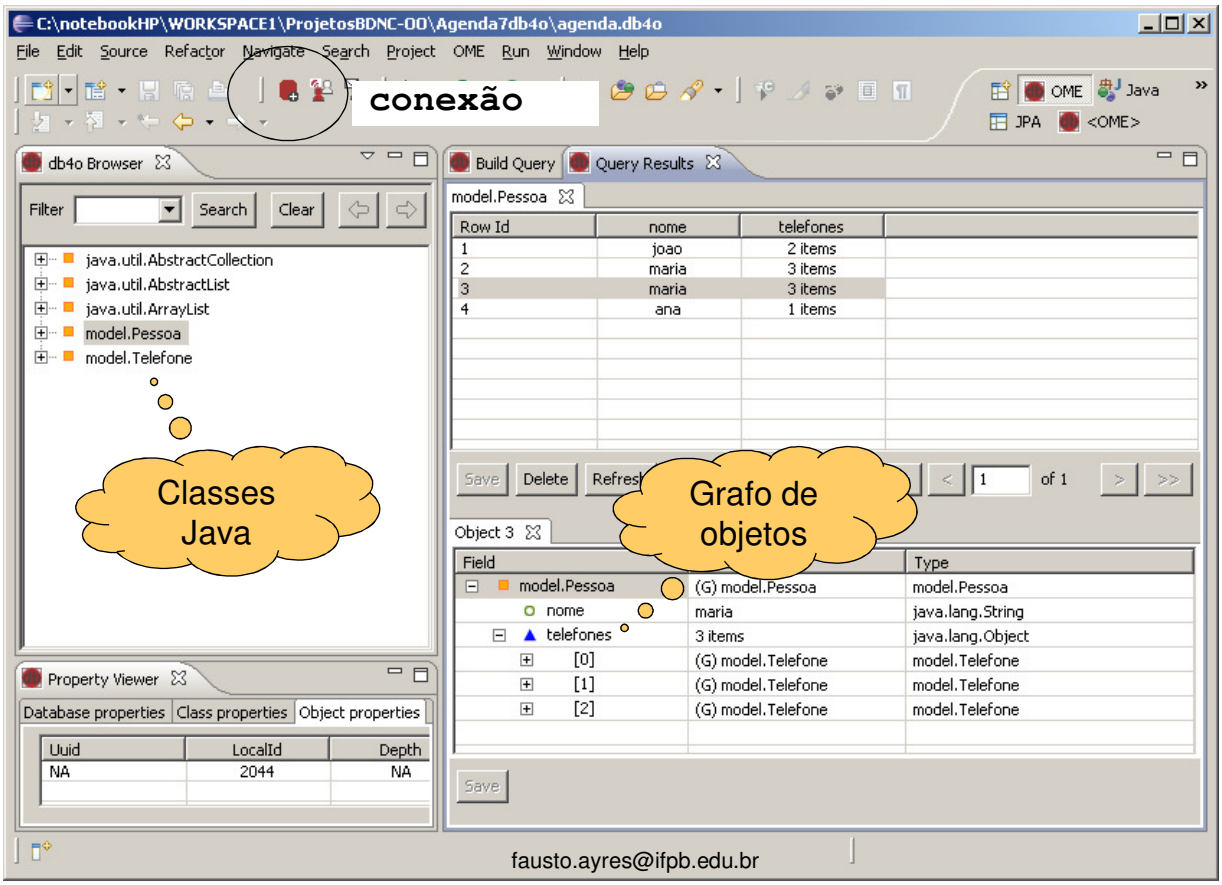


Instalação

Help / Install new software



Visualizando o bd



API DB4O

Instanciação do Gerenciador da Persitência

```
import com.db4o.*;

EmbeddedConfiguration config =
                    Db4oEmbedded.newConfiguration();
config.common().messageLevel(0);  // 0,1,2,3

ObjectContainer manager =
        Db4oEmbedded.openFile(config,"banco.db4o");

manager.store(objeto)
manager.delete(objeto)
manager.query(...)
manager.commit()
manager.rollback()
manager.close()
```

Persistir objetos

```
Exemplo:
...
Pessoa p1 = new Pessoa("joao");
Pessoa p2 = new Pessoa("maria");

manager.store(p1);
manager.store(p2);
manager.commit();
...
```

A primeira operação *store()* inicia a transação.

Efetiva as gravações

Localizar um objeto

usando a API *SODA*

```
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);
   ...
}
```

busca por nome

Alterar um objeto

- Deve-se localizar o objeto antes de altera-lo

```
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);
      p.setNome("joana");
      manager.store(p);
      manager.commit();
}
else
      System.out.println("pessoa inexistente");
```


Apagar um objeto

- Deve-se localizar o objeto antes de remove-lo:

```
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);
      manager.delete(p);
      manager.commit();
}
else
      System.out.println("inexistente");
```

Transação

```
try{
   ...
   manager.store(p1);
   manager.commit();
   ...
}
catch(Exception e){
   manager.rollback();
}
```


Consultas polimórficas

```
List<Pessoa> resultados = manager.query(Pessoa.class);
for(Pessoa p: resultados)
   System.out.println(p);
```

pessoas e alunos

```
List<Aluno> resultados = manager.query(Aluno.class);
for(Aluno a: resultados)
   System.out.println(a);
```

Somente alunos

Apagar todos objetos de uma classe

Apagar pessoas

```
List<Pessoa> resultados = manager.query(Pessoa.class);

for(Pessoa p: resultados)  {
   manager.delete(p);
   manager.commit();
}
```

Apagar todos objetos

```
List<Object> resultados = manager.query(Object.class);

for(Object ob: resultados)  {
   manager.delete(ob);
   manager.commit();
}
```


Refatoração de objetos persistentes

- Exemplo:

1. Inclua o atributo na classe Pessoa;

   private int idade=18;

2. Persista novos objetos Pessoa no bd

3. Observe se os objetos antigos do bd foram refatorados.

OPERAÇÕES EM CASCATA

Configuração da cascata

- Para EXCLUSÃO, ALTERAÇÃO, e LEITURA de relacionamentos em cascata, é necessário a seguinte configuração do bd.

```
EmbeddedConfiguration config =
   Db4oEmbedded.newConfiguration();
config.common().
      messageLevel(0);  // 0,1,2,3...
config.common().
      objectClass(Pessoa.class).cascadeOnUpdate(true);
config.common().
      objectClass(Pessoa.class).cascadeOnDelete(true);
config.common().
      objectClass(Pessoa.class).cascadeOnActivate(true);

ObjectContainer manager =
            Db4oEmbedded.openFile(config, "banco.db4o");
```

Gravação em cascata

- A persistência é feita *automaticamente* em cascata, NÃO É PRECISO CONFIGURAR

```
Pessoa p1 = new Pessoa("joao");
p1.adicionarTelefone(new Telefone ("8800-0000"));
p1.adicionarTelefone(new Telefone ("8800-1111"));

manager.store(p1);
manager.commit();
```

joao e seus telefones serão persistidos em cascata

Exclusão em cascata

```
List<Pessoa> resultados1 = ...localizar joao

if(resultados1.size()>0) {
  Pessoa p = resultados1.get(0);

  manager.delete(p);
  manager.commit();
}

else
  System.out.println("inexistente");
```

joao e seus telefones serão apagados em cascata

Alteração em cascata

Adicionar um novo telefone da pessoa

```
List<Pessoa> resultados = ...localizar joao

if(resultados.size()>0) {
   Pessoa p = resultados.get(0);
   Telefone t = new Telefone("8800-9999");
   p.adicionar(t);
   t.setPessoa(p);
   manager.store(p);
   manager.commit();
}

else
   System.out.println("inexistente");
```

O novo telefone será criado no banco e o relacionamento também

Alteração em cascata

Remover um telefone existente da pessoa

```
List<Pessoa> resultados1 = ...localiza joao...
if(resultados1.size()>0) {
   Pessoa p = resultados1.get(0);
   Telefone t = p.localizar("8800-1111");
   p.remover(t);
   t.setPessoa(null);
   manager.store(p);
   manager.store(t);
   manager.commit();
}

else
   System.out.println("inexistente");
```

Como o telefone não está mais relacionado, ele deve ser atualizado separadamente

E ai o telefone fica "órfão" no bd

Órfão

Para impedir que um objeto fique "órfão", ele deve ser apagado do bd.

```
List<Pessoa> resultados1 = ...localiza joao...
if(resultados1.size()>0) {
   Pessoa p = resultados1.get(0);
   Telefone t = p.localizar("8800-1111");
   p.remover(t);
   //t.setPessoa(null);
   manager.store(p);
   manager.delete(t);
   manager.commit();
}

else
   System.out.println("inexistente");
```

Exclusão do telefone órfão

Criando índices de acesso (opcional)

- Índices otimizam a consulta
- São criados na configuração

```
EmbeddedConfiguration config = Db4oEmbedded.newConfiguration();
config.common().
objectClass(Pessoa.class).objectField("nome").indexed(true);
```