

Versão 5.0 | Java e .NET

# db4o - Banco de objetos de código aberto

db4o é o banco de objetos open source que possibilita aos desenvolvedores Java e .Net reduzir o tempo e o custo de desenvolvimento e alcançar níveis nunca vistos de performance.

O design ímpar do mecanismo nativo da base de objetos faz com que ele seja a escolha ideal para aplicações embarcadas em equipamentos e dispositivos, em software de caixinha (prateleira) executado tanto em desktop como dispositivos móveis, ou em sistemas de controle de tempo real – em resumo: em todos os ambientes Java e .Net onde não há um DBA presente

## Bancos Relacionais, Mapeadores Objeto-Relacional e Banco de Objetos db4o



Todos os desenvolvedores orientados a objetos sabem das dificuldades de passar um modelo orientado a objetos para uma persistência relacional. Até agora eram forçados a escolher entre velocidade e orientação a objetos: acesso nativo ao SQL é rápido, mas trabalhoso, requerendo um enorme esforço extra para codificação. Mapeadores objeto-relacional oferecem uma ponte conveniente, no entanto eles degradam seriamente a performance.

db4o elimina a troca OO por performance: permite você armazenar até as mais complexas estruturas de objetos com facilidade enquanto atinge altos níveis de performance. Avaliações de banco de dados mostram que o db4o pode ser até 44x mais rápido que o Hibernate e MySQL, uma combinação popular de mapeador objeto-relacional e banco de dados.

A maior razão para utilizar banco de dados relacionais hoje em dia é o legado, como por exemplo, o armazenamento de antigas informações da empresa e o conjunto de aplicações existentes para eles. Mas além da persistência centrada em servidor, existe uma multidão de dispositivos, celulares e aplicações desktop onde a tecnologia convencional de banco de dados não alcança. Aqúj a tecnologia do db4o assegura novos níveis de performance, funcionalidade e eficácia de custo.

**db4o é o banco de objetos open source que possibilita aos desenvolvedores Java e .Net reduzir o tempo e o custo de desenvolvimento e alcançar níveis nunca vistos de performance.**

**O design ímpar do mecanismo nativo de base de objetos faz com que ele seja a escolha ideal para aplicações embarcadas em equipamentos e dispositivos , em software de caixinha (prateleira) executados tanto em desktop como dispositivos móveis, ou em sistemas de controle de tempo real – em resumo: em ambientes onde não há um DBA presente**

## Produto - Aplicação, Recursos, Benefícios

db4o é desenvolvido para prover um completo banco de dados embutido para equipamentos, dispositivos móveis, plataformas desktop e servidor em ambientes orientado a objeto. Onde bancos de dados relacionais falham em prover administração zero, alta performance, sincronismo suave e fácil refatoramento (manutenção/alteração do modelo), db4o é a resposta. Nativo para Java e .Net, é a única biblioteca de desenvolvimento (.jar/.dll) que se integra facilmente às aplicações e executa de forma altamente confiável e escalável tarefas de persistência com somente uma simples linha de código, não importando o quão complexas as estruturas possam ser.

Como resultado, os desenvolvedores podem:

- Eliminar ferramentas e códigos para o mapeamento objeto-relacional, os quais comprovadamente levam ao aumento da complexidade do código e consumo de recursos enquanto inibem o desempenho, a facilidade de manutenção e alteração do código. Com o db4o, os usuários ganham até 90% de tempo e redução de custos para o desenvolvimento de softwares no tocante à persistência.
- Construir aplicações sem amarras com o armazenamento de dados que não necessitam de administração em tempo de execução, altamente confiáveis e muito mais rápidas do que engines convencionais ou proprietários de armazenamento de dados.
- Benefícios do paradigma orientado a objetos, sem estar preso pelo banco de dados, permitindo modelos de objetos mais naturais e ricos em recursos sem direcionar para cima custos e consumo de recursos.
- Mudanças, refactor e reutilização de componentes de software com habilidade para adicionar novos recursos sem quebrar o código legado ou incorrer em altos custos – permitindo mais flexibilidade para permanecer no topo da competição.

db4o já passou da marca de 250.000 downloads. Ele já foi instalado com sucesso em empresas do ramo de transporte, rede, ciências naturais, industrial, consumo, aplicações corporativas e uma miríade de outros ramos. Usuários e consumidores do db4o estão em mais de 168 países, da Albânia ao Zimbábue, e uma variedade de líderes como Boeing, Bosch, BMW e Hertz e uma variada faixa de empresas iniciantes altamente inovativas.

Construído em nova tecnologia de banco de objetos, o db4o atualmente é o único banco de dados que é nativo tanto para Java e .Net – fornecendo portabilidade entre plataformas que libera usuários de altas taxas de licenciamento dos fornecedores proprietários. Oferecendo uma grande quantidade de funcionalidades exclusivas e orientadas a objeto, o db4o incentiva os benefícios das linguagens orientadas a objeto: sua replicação orientada a objeto, suas queries orientadas a objeto (Queries

### db4o Estudos de Caso

[www.db4o.com/about/customers](http://www.db4o.com/about/customers)



*AVE High Speed Trains*



*BOSCH's Packaging Robots*



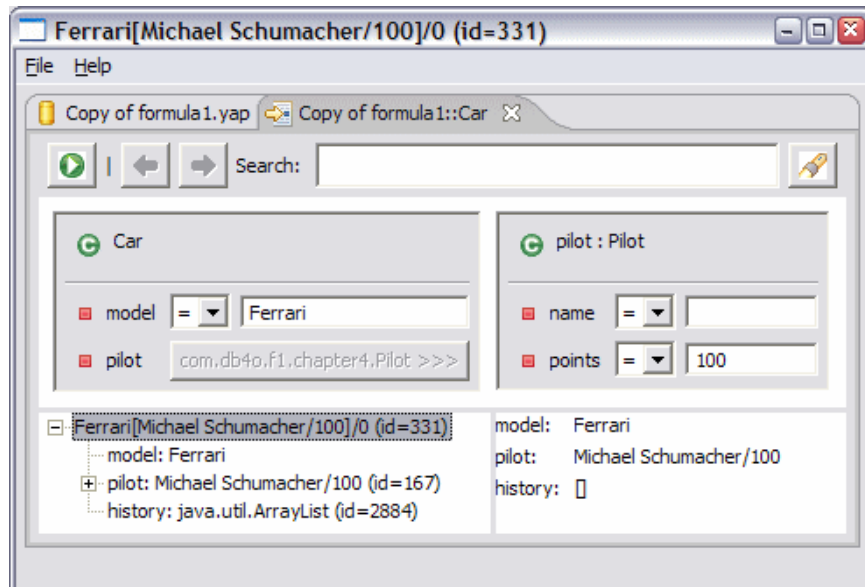
*Clarity Medical's RetCam II*



*Eastern Data's Mobile Apps*

[www.db4o.com/about/customers](http://www.db4o.com/about/customers)

nativas, QbE, S.O.D.A) e a GUI ObjectManager para mostrar arquivos de base de objetos são inigualáveis no mundo dos banco de dados.



*O ObjectManager do db4o pode ser usado para consultar e listar arquivos de objetos do db4o*

Acima de tudo, db4o é fácil de aprender, implementar e usar. O poderoso engine do db4o permite aos usuários armazenar objetos com somente uma linha de código, cortando custos e tempo de desenvolvimento da camada de persistência ao mínimo.

Esses benefícios são quadruplicados quando é necessário alterar um software, adicionar novos recursos e reutilizar componentes. O db4o automatiza o refactoring pelos desenvolvedores, pois todas as APIs não nativas e strings são eliminadas. Se um update é aplicado na base já instalada, o versionamento automático de esquemas gerencia automaticamente as mudanças sem a necessidade de processos de conversão. Os desenvolvedores não necessitam de nenhum processo de pré ou pós compilação. E processos regulares de replicação são totalmente transparentes a mudanças de software, evitando quebrar arquiteturas distribuídas de dados.

Novos usuários poderão facilmente efetuar o download e iniciar no db4o em 5 minutos ou menos. Um tutorial interativo está incluído no download, ajudando os desenvolvedores a iniciar e ter uma transição fácil do modelo relacional para o modelo orientado a objetos.

**Whitepapers do db4o disponíveis para download gratuito:**

- The Database Behind the Brains, by Rick Grehan
- Complex Object Structures, Persistence, and db4o, by Rick Grehan
- Native Queries for Persistent Objects, by William Cook and Carl Rosenberger
- Agile Techniques for Object Databases, by Scott Ambler

[www.db4o.com/about/productinformation/whitepapers](http://www.db4o.com/about/productinformation/whitepapers)

**Benchmarks**

[www.db4o.com/about/productinformation/benchmarks](http://www.db4o.com/about/productinformation/benchmarks)



## Recursos e benefícios

Recursos chave	Benefícios chave
<p><b>O banco-de-uma-linha-de-código</b>                      Uma linha de código armazena qualquer objeto                      Modelo de Classes = esquema de objetos                      Processo de produção suave</p> <p><b>Embarcável</b>                      Administração zero                      Versionamento automático de esquema                      400 KB de tamanho</p> <p><b>Suporte a múltiplas plataformas</b>                      Nativo para Java e .NET                      CPU embarcadas, dispositivos móveis, desktop e servidores                      Execução <i>cross-platform</i></p> <p><b>Traz mais OO para o banco de dados</b>                      Replicação orienta a objetos                      Queries nativas                      ObjectManager browser</p>	<p><b>Corta 90% do custo</b> para desenvolver persistência</p> <p><b>10% mais rápido para comercializar</b> sua aplicação</p> <p>Roda até <b>44x mais rápido</b> que aplicações tradicionais</p> <p>Distribuível em grandes volumes sem administração local</p> <p>Construa Sistemas leves e realmente orientados a objeto</p> <p>Construa aplicações distribuídas e totalmente sincronizadas</p> <p>Menos erros, melhor refatoramento e longevidade do software.</p>



### **O banco de dados de uma-linha-de-código economiza seu tempo**

É tão fácil quanto isto: coloque a biblioteca de desenvolvimento do db4o em seu ambiente de desenvolvimento, abra um arquivo de banco de dados e grave qualquer objeto – não importa o quão complexo seja . Exemplo em java:

```
public void store(Car car){
    ObjectContainer db =
        Db4o.openFile("car.yap");
    db.set(car);
    db.commit();
    db.close();
}
```

Essa incomparável facilidade de utilização resulta em uma dramática redução de tempo de desenvolvimento.

Elimine todo o trabalho de desenhar, implementar e manter o esquema da base de dados, pois o modelo de classes é o esquema do banco de dados. Elimine a necessidade de gerenciar qualquer coisa adicional como strings de conexão, xml ou arquivos não nativos que necessitam pré-compilação e conseqüentemente tornam cada vez mais lento seu processo de produção.

Você estará rodando uma aplicação em menos de 5 minutos, auxiliado pelo aclamado tutorial interativo "Formula 1".

### **Você economizará muito tempo quanto estiver escrevendo o seu software**

Você economizará tempo cada vez que for alterar seu software, como por exemplo, refatorando o código, adicionando novas features, ou reutilizando componentes. Alterar seu modelo de objetos, por exemplo, não é somente transparente, mas também infalível pois a natureza nativa e não intrusiva do db4o deixa o ambiente de desenvolvimento fazer todo o trabalho para você. Você não necessita de debuggers, nem processos de build especiais, e você não necessita se preocupar com instalações já existentes, pois o db4o cuida de qualquer modificação de objeto para sua base instalada inteira. Alterar software deixa de ser menos pesado e torna-se mais um prazer, fazendo-o mais produtivo.

Em essência, o db4o fará com que persistir objetos seja tão simples quanto usar serialização plana, mas também fornece a você todas as funcionalidades de um banco de dados como queries, e –notavelmente – permite alterar o modelo de objeto sem quebrar a serialização.

### **O banco de dados embutido**

O db4o é desenvolvido para ser embutido em clientes ou outros componentes de software de forma completamente invisível para o usuário final. Deste modo, o db4o não precisa de um mecanismo separado de instalação, mas vem com uma biblioteca facilmente distribuível com um tamanho de aproximadamente 400Kb. Pelo fato do db4o rodar no mesmo processo da sua aplicação, você possui total controle do gerenciamento de memória e pode debugar todo

o sistema. Se sua aplicação está rodando, o banco também está rodando. Sem exceções.

### **Portabilidade e distribuição *Cross-Platform***

Poucos bancos de dados rodam nativamente em tantas plataformas orientadas a objeto – e nenhum em ambientes heterogêneos com essas plataformas. As capacidades do db4o possibilitam que você desenvolva aplicações para distribuição em múltiplas plataformas (como por exemplo o mercado de PDA) ou em combinações de clientes Windows e servidores Java.

O db4o suporta Java JDK 1.1.x até 5.0 e roda em J2EE e J2SE. db4o também roda com dialetos J2ME que suportam reflection, como CDC, PersonalProfile, Symbian, Savaje e Zaurus. Dependendo da demanda dos clientes, db4o também será executado em dialetos sem reflexão, como CLDC e MIDP, incluindo RIM (Blackberry) e Palm OS.

O db4o também roda em todas as plataformas .Net incluindo .NET, a CompactFramework e Mono, suportando todas as linguagens gerenciadas como: C#, VB.NET, ASP.NET, Boo, e Managed C++.

### **Clientes parcialmente conectados e arquitetura de dados distribuídas**

Enquanto as aplicações com db4o estão prioritariamente em clientes *standalone*, como smartphones ou um carro, alguns desses clientes hoje em dia estão parcialmente conectados a um middleware, servidores ou outros dispositivos. Portanto, db4o não oferece apenas um modo embutido, mas também um modo pronto client/server.

A exclusiva funcionalidade de replicação orientada a objeto do db4o permite uma fácil sincronização de dados entre clientes e servidores ou entre pontos. A implementação é muito simples: conecte duas instâncias de banco de objetos (em local e/ou modo servidor) para procurar objetos alterados e sincronize com apenas uma linha de código. Os relacionamentos pai-filho das classes permitem uma maneira fácil de descrever o que deve ser duplicado juntamente com a classe pai. Você pode armazenar também as regras de negócio necessárias para resolver conflitos de sincronização entre objetos para criar objetos que podem se mover livremente em arquiteturas distribuídas.

Em suma, os diversos modos de execução do db4o e sua exclusiva replicação orientada a objetos permitem uma arquitetura de dados distribuída poderosa e eficiente, o que é tipicamente requerido pela computação orientada a serviços.

## Queries Nativas

Com a versão 5.0 do db4o, db4objects é a primeira a implementar Queries Nativas(NQ).  
Queries nativas

As Queries Nativas fazem parte de uma tendência das empresas de software prover consultas a banco de dados com a mesma semântica da linguagem de programação. Um exemplo de queries nativas é o LINQ da Microsoft que deverá ser incorporado na próxima versão do framework .Net.

No lugar das antigas APIs baseadas em string (como SQL, OQL, JDOQL, EJBQL, e SODA), NQ permite aos desenvolvedores simplesmente usar a própria linguagem de programação para acessar o banco de dados e desta forma evitando uma constante perda de produtividade na troca entre linguagem de programação e API de acesso a dados.

Por exemplo, compare esta query nativa in C# para .NET 2.0:

```
IList<Student>students = db.Query<Student>(delegate(Student student) {
    return student.Age < 20
        && student.Grade == gradeA;
});
```

... ou em Java:

```
List<Student> students = database.query<Student>( new Predicate(){
    public boolean match(Student student){
        return student.getAge() < 20
            && student.getGrade().equals(gradeA);}});
```

... com o mesmo em JDOQL:

```
String param = "String gradeParam"
String filter = "age < 20 & grade == gradeParam";
Query q = persistenceManager.createQuery(Student.class, filter);
q.declareParameters(param);
Collection students = (Collection)q.execute(gradeA);
```

Como você pode ver, Queries Nativas eliminam todas as strings das queries – elas são 100% fortemente tipadas, 100% refatoráveis, e 100% orientadas a objeto. No exemplo acima, JDOQL pode aceitar qualquer coisa em sua string “filter”. Como isso não pode ser verificado em tempo de compilação, você pode ter erros de tipo em tempo de execução.

Como resultado, acesso a dados com Queries Nativas fazem com que os desenvolvedores de software sejam mais produtivos. Elas também facilitam freqüentes alterações e customizações no modelo de objetos.

### Outras APIs e Borwser gráfico de objetos

Duas APIs orientadas a objeto adicionais, Query By Example e S.O.D.A oferecem opções adicionais de query para casos específicos e/ou aplicações legadas.

Com Queries por Exemplo, você tem uma forma extremamente simples de executar consultas baseada em templates, exemplo:

```
Car car = new Car();
car.setName("Ferrari");
List cars = db.get(car);
```

S.O.D.A. é uma poderosa API de consulta baseada em nós de grafos para criação de queries dinâmicas. Ela permite executar consultas no servidor, reduzindo o tráfego de rede e o tempo de execução da query.

Exportação para XML é facilmente conseguida agora com qualquer biblioteca que exporta objetos java para XML, como Xstream.

O ObjectManager é uma ferramenta visual para acessar, consultar, percorrer e editar arquivos do db4o, quando as classes da aplicação não estão presentes.



## Especificações | db4o V 5.0

Platforms	
Java	<ul style="list-style-type: none"> <li>• J2EE</li> <li>• J2SE</li> <li>• J2ME com reflexão: CDC, PersonalProfile, Symbian, Savaje e Zaurus. Em estudo: J2ME w/o reflection (CLDC e MIDP) , incluindo RIM/Blackberry e Palm OS</li> <li>• Servlet/JSP framework</li> <li>• Java Web Start</li> </ul>
.NET	<ul style="list-style-type: none"> <li>• .NET 1.x</li> <li>• .NET 2.0</li> <li>• CompactFramework</li> <li>• Windows Mobile 5.0</li> <li>• Mono</li> </ul>
Languages	
Java	<ul style="list-style-type: none"> <li>• JDK 1.1.x - JDK 5.0</li> </ul>
.NET	<ul style="list-style-type: none"> <li>• Todas linguagens gerenciadas (C#, VB.NET, ASP.NET, Boo, C++ gerenciado etc.)</li> </ul>
Comandos	
Sessões	<ul style="list-style-type: none"> <li>• Início e fim</li> </ul>
Arquivos da base de dados	<ul style="list-style-type: none"> <li>• Criar, Abrir, Fechar e Apagar</li> </ul>
Transações	<ul style="list-style-type: none"> <li>• Commit e Rollback</li> </ul>
Objetos	<ul style="list-style-type: none"> <li>• Armazenar, recuperar, alterar (incluindo em cascata), replicar, apagar(inclusive em cascata)</li> </ul>
Messaging	<ul style="list-style-type: none"> <li>• TCP/IP</li> </ul>
Transparência	
Linguagem	<ul style="list-style-type: none"> <li>• Tipos primitivos</li> <li>• Strings</li> <li>• Arrays</li> <li>• Arrays multidimensionais</li> <li>• Classes internas</li> <li>• Coleções Java/C#</li> <li>• Classes sem construtores públicos.</li> <li>• Structs NET</li> </ul>

	<ul style="list-style-type: none"> <li>• Blobs (armazenados fora do arquivo da base)</li> </ul>
Não intrusivo	<ul style="list-style-type: none"> <li>• Sem necessidade de derivar de uma classe específica</li> <li>• Sem necessidade de implementar de uma interface específica</li> <li>• Sem modificações do código fonte</li> <li>• Sem necessidade de implementar [Serializable]</li> </ul>
Campos privados	<ul style="list-style-type: none"> <li>• Armazenáveis</li> </ul>
I/O de arquivo	<ul style="list-style-type: none"> <li>• Plugável</li> </ul>
Reflector	<ul style="list-style-type: none"> <li>• Plugável</li> <li>• Genérico</li> </ul>
<b>Linguages de query / APIs</b>	
Orientado a Objeto	<ul style="list-style-type: none"> <li>• Queries nativas</li> <li>• Query By Example (QbE)</li> <li>• S.O.D.A.</li> </ul>
SQL	<ul style="list-style-type: none"> <li>• SQL-92 somente leitura (em estudo)</li> </ul>
XML	<ul style="list-style-type: none"> <li>• Com produtos de terceiros(exemplo, Xstream)</li> </ul>
<b>Modes / Concurrency</b>	
Modos de operação	<ul style="list-style-type: none"> <li>• Local</li> <li>• Client/Server</li> </ul>
Threads	<ul style="list-style-type: none"> <li>• Múltiplas</li> </ul>
Transações	<ul style="list-style-type: none"> <li>• Múltiplas, paralelas</li> </ul>
Semáforos	<ul style="list-style-type: none"> <li>• Disponível</li> </ul>
Modo somente leitura	<ul style="list-style-type: none"> <li>• Disponível</li> </ul>
<b>Escalabilidade e Performance</b>	
Avaliação de Performance	<ul style="list-style-type: none"> <li>• Até 44x mais rápido que Hibernate/MySQL</li> </ul>
Exemplos de escalabilidade	<ul style="list-style-type: none"> <li>• Armazena 200,000 objetos/segundo</li> <li>• Armazena 20,000 classes</li> <li>• Armazena 300,000 objetos em um PDA</li> </ul>
Operação em memória	<ul style="list-style-type: none"> <li>• Disponível</li> </ul>
Client-side	<ul style="list-style-type: none"> <li>• Disponível processamento monousuário</li> </ul>
Server-side	<ul style="list-style-type: none"> <li>• Disponível execução de consultas no servidor</li> </ul>
DB-aware Collections	<ul style="list-style-type: none"> <li>• Disponível</li> </ul>
Cache de objetos	<ul style="list-style-type: none"> <li>• Disponível</li> </ul>
Indexação	<ul style="list-style-type: none"> <li>• Disponível</li> </ul>

<b>Replicação</b>	
UUID	<ul style="list-style-type: none"> <li>• Identificador Único Universal através de todas as instâncias de banco de dados</li> </ul>
Movimentação de Objetos	<ul style="list-style-type: none"> <li>• Mover/copiar Simples entre bases de dados</li> </ul>
<b>Segurança e Encriptação</b>	
Proteção do arquivo da base	<ul style="list-style-type: none"> <li>• Senha</li> </ul>
Encriptação do arquivo da base	<ul style="list-style-type: none"> <li>• Encriptação simples</li> <li>• I/O de arquivo extensível permitindo encriptação customizada</li> </ul>
Transações	<ul style="list-style-type: none"> <li>• ACID</li> <li>• Commit-recovery em falhas do sistema</li> </ul>
Thread Safety	<ul style="list-style-type: none"> <li>• Disponível</li> </ul>
Recuperação automática	<ul style="list-style-type: none"> <li>• Disponível</li> </ul>
Backup online	<ul style="list-style-type: none"> <li>• Disponível</li> </ul>
Gerenciamento de espaço disponível	<ul style="list-style-type: none"> <li>• Disponível</li> </ul>
<b>Internacionalização</b>	
Unicode	<ul style="list-style-type: none"> <li>• Disponível</li> </ul>
<b>Refactoring</b>	
Versionamento do esquema	<ul style="list-style-type: none"> <li>• Reconhecimento e manutenção automáticas</li> </ul>
Junção de esquemas	<ul style="list-style-type: none"> <li>• Disponível</li> </ul>
Alteração de nomes	<ul style="list-style-type: none"> <li>• Classes e nomes</li> <li>• Acesso aos valores de campos removidos pela API</li> </ul>
<b>Memória e tamanho do arquivo</b>	
Tamanho inicial da biblioteca	<ul style="list-style-type: none"> <li>• ~400 KB</li> </ul>
Mínimo de RAM utilizada	<ul style="list-style-type: none"> <li>• ~1 MB</li> </ul>
Tamanho máximo do BD	<ul style="list-style-type: none"> <li>• 254 GB / arquivo de objetos</li> </ul>
Tamanho da classe	<ul style="list-style-type: none"> <li>• Não aumenta quando persistida</li> </ul>
Tamanho da instância	<ul style="list-style-type: none"> <li>• Não aumenta quando persistida</li> </ul>