

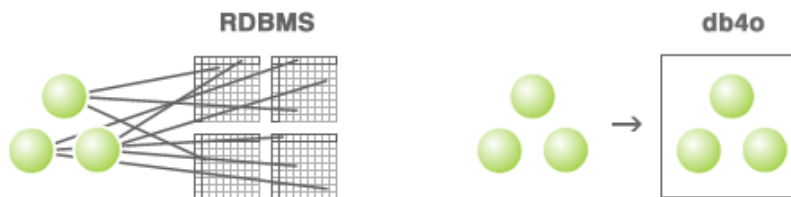


Version 6.0 | Java und .NET

db4o Open Source Objektdatenbank

db4o ist die Open Source Objektdatenbank, die es Java und .NET Entwicklern ermöglicht, Entwicklungszeit und -kosten auf ein Minimum zu reduzieren und gleichzeitig einen neuen Grad an Performance zu erreichen. Das einzigartige Design von db4os nativer Objektdatenbank Engine macht es zur idealen Lösung für eingebettete Anwendungen in Geräten und Anlagen, Softwarepaketen auf mobilen oder Desktop Plattformen und in Echtzeit-Steuerungssystemen – kurz gesagt: in allen Java und .NET Umgebungen, in denen es keinen Datenbank Administrator gibt.

Relationale Datenbanken, Objekt-Relationale Mapper und db4os Objektdatenbank



Jeder Entwickler von objekt-orientierter Software kennt das Problem der Überführung von Objektorientierung in relationale Persistenz. Bis jetzt war man gezwungen sich zwischen Geschwindigkeit oder Objektorientierung zu entscheiden: Nativer SQL Zugriff ist schnell, aber auch umständlich, da hierfür eine erhebliche Menge zusätzlicher Code erforderlich ist. Objekt-relationale Mapper sind zwar eine Möglichkeit das Problem zu umgehen, jedoch nur zu Lasten stark verminderter Performance.

Mit db4o gehört der OO-gegen-Performance Kompromiss der Vergangenheit an: db4o ermöglicht das einfache Speichern der komplexesten Objektstrukturen bei gleichzeitig maximaler Performance. [Datenbank-Benchmarks](#) beweisen, dass db4o bis zu 55 mal schneller ist, als Hibernate und MySQL, eine weit verbreitete Kombination von objekt-relationalen Mapper und SQL Datenbank.

Der Hauptgrund für das Festhalten an relationalen Datenbanken heutzutage ist „Erbe“, z.B. zur Erhaltung alter Firmendaten und den zugrunde liegenden Anwendungsprogrammen. Außerhalb der server-fixierten Persistenz gibt es jedoch eine Vielzahl von mobilen und Desktop-Anwendungen, für die konventionelle Datenbanktechnologie keine zufriedenstellende Lösung bietet. Hier garantiert db4os Technologie die Erreichung eines neuen Grades an Performance, Funktionalität und Kosteneffektivität.

Nichts desto trotz ist mit dem [db4o Replication System \(dRS\)](#) weiterhin volle Datenkompatibilität zu „vererbten“ RDBMs wie Oracle oder MySQL gewährleistet.

db4o ist die Open Source Objektdatenbank, die es Java und .NET Entwicklern ermöglicht Entwicklungszeit und -kosten auf ein Minimum zu reduzieren und gleichzeitig einen neuen Grad an Performance zu erreichen.

Das einzigartige Design von db4os nativer Objektdatenbank Engine macht es zur idealen Lösung für eingebettete Anwendungen in Geräten und Anlagen, Softwarepaketen auf mobilen oder Desktop Plattformen und in Echtzeit-Steuerungssystemen – kurz gesagt: in allen Java und .NET Umgebungen, in denen es keinen DBA gibt.



Produkt – Anwendung, Besonderheiten und Nutzen

db4o ist als vollausgestattete, einbettbare Datenbank-Engine in Anlagen, mobilen Geräten und Desktop- und Server-Plattformen in einer objektorientierten Umgebung gedacht. In allen Einsatzbereichen, in denen relationale Datenbanken auf Grund mangelnder Wartungsfreiheit, großem Footprint, schwacher Performance, problembehafteter Synchronisation und schwerer Refakturierbarkeit ausscheiden, ist db4o die Lösung. Dank db4os nativer Java und .NET Programmbibliothek (.jar / .dll) gliedert es sich nahtlos in die Anwendung ein und erledigt höchst verlässliche und skalierbare Persistenz-Aufgaben mit nur einer einzigen Zeile Code, egal wie komplex die Objektstrukturen auch sein mögen.

Für Entwickler bedeutet dies:

- Keine zusätzliche Software und Programmcode für objekt-relacionales Mapping mehr, die Code-Komplexität und Ressourcenverbrauch erhöhen, während sie Performance und Refakturierbarkeit deutlich mindern. Mit db4o können Zeit- und Kostenersparnisse von bis zu 90% bei der Persistenz-Entwicklung verwirklicht werden.
- Entwickeln von Anwendungen mit nahtlos integriertem Datenspeicher, der keine Administration benötigt und zugleich höchst zuverlässig und weitaus schneller arbeitet als konventionelle oder proprietäre Datenbank-Engines.
- Profitieren von objekt-orientierten Modellen ohne den Beschränkungen der Datenbank zu unterliegen. Dies ermöglicht die Verwendung von komplexeren Objektmodellen, ohne Kosten und Ressourcenverbrauch in die Höhe zu treiben.
- Ändern, Refakturieren und Wiederverwenden von Software-Komponenten mit der Möglichkeit, neue Features hinzuzufügen ohne den alten Code zu verlieren – dies ermöglicht den höheren Grad an Flexibilität, der nötig ist, um dem Wettbewerb voraus zu sein.

db4o wird von einer einzigartigen Community von Java und .NET Entwicklern getragen, die heute schon 18.000 registrierte Mitglieder zählt und täglich wächst. Das Produkt wurde bereits rund 1.000.000 mal heruntergeladen und ist erfolgreich in [Transport](#), [Netzwerken](#), [Naturwissenschaften](#), [Industriearomatisierung](#), [Konsumgütern](#) und [Unternehmensanwendungen](#) im Einsatz. db4os Anwender und Kunden kommen aus 170 verschiedenen Ländern, von Albanien bis Zimbabwe, und reichen von weltweit führenden Unternehmen wie [Boeing](#), [Bosch](#), [Intel](#), [Ricoh](#), und [Seagate](#) bis hin zu einer großen Anzahl hoch-innovativer Start-Ups.

Aufbauend auf neuester Datenbank-Technologie, ist db4o momentan die einzigste Datenbank, die sowohl nativ zu Java als auch zu .NET ist. Dies ermöglicht eine plattformübergreifende Portabilität, die Anwender von den hohen Lizenzgebühren der proprietären Anbieter befreit. Mit seinem vielfältigen Angebot an objekt-orientierten Datenbank-

db4o Fallstudien

www.db4o.com/about/customers



AVE Hochgeschwindigkeits-Züge



BOSCHs Verpackungsroboter

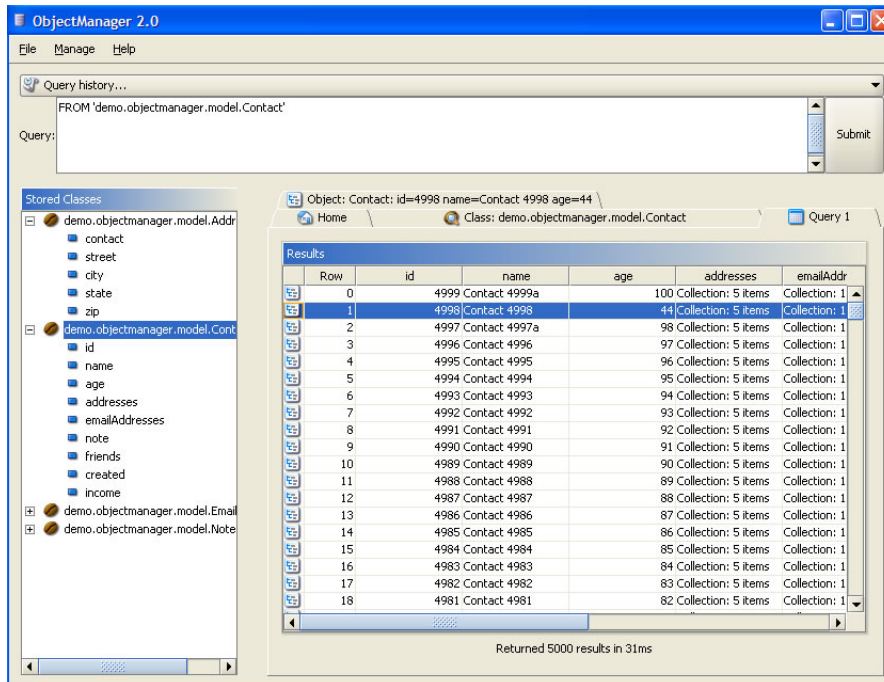


Seagates Mirra Server



Eastern Datas Mobile Apps

Funktionalitäten, werden die Vorteile von OO Programmiersprachen durch db4o erst richtig nutzbar: db4os objekt-orientierte Replikation (dRS), objekt-orientierte Queries (Native Queries, QbE, S.O.D.A.), und [ObjectManager](#) Oberfläche, zum Navigieren in Objektdatenbank Dateien, sind in der Datenbankwelt einzigartig.



Mit dem db4o ObjectManager können db4o Datenbank-Dateien navigiert und abgefragt werden

Vor allem aber ist db4o sehr leicht zu erlernen, anzuwenden und zu implementieren. db4os leistungsfähige Datenbank-Engine erlaubt es Anwendern, Objekte mit nur einer einzigen Zeile Code zu speichern und somit die Entwicklungszeit und –kosten der Persistenzschicht auf ein Minimum zu reduzieren.

Diese Vorteile kommen besonders bei der Änderung und der Wiederverwendung von Softwarekomponenten zum Tragen. Dank db4os Entwicklungsumgebung wird der Refactoring-Prozess automatisiert, da alle nicht-nativen APIs und Zeichenketten eliminiert werden. Bei der Installation von Updates auf Endgeräten wird der Konvertierungsprozess überflüssig: db4os automatische Schemaerkennung kümmert sich um die Änderungen im Datenmodell. Mit db4o brauchen Entwickler keine zeitraubenden Enhancer, Pre- oder Post-Compiler mehr. Auch die Replikationsarchitektur ist robust gegenüber Softwareänderungen, so dass Updates einfacher und kostengünstig erstellt und installiert werden können.

Neue User werden feststellen, dass db4o in weniger als 5 Minuten heruntergeladen und anwendungsbereit ist! Im Download ist ein umfangreiches, interaktives Tutorial enthalten, das Entwicklern den Einstieg und das Umdenken von relationaler zu objekt-orientierter Persistenz erleichtert.



db4o Whitepaper sind zum kostenlosen Download verfügbar, z.B.:

- The Database Behind the Brains
- Complex Object Structures, Persistence, and db4o

www.db4o.com/about/productinformation/whitepapers

Benchmarks

www.db4o.com/about/productinformation/benchmarks

Features und Vorteile

Schlüssel-Features	Schlüssel-Vorteile
<p>Die Eine-Zeile-Code-Datenbank</p> <p>Eine Zeile Code speichert jedes Objekt</p> <p>Klassenmodell = Objektschema</p> <p>Reibungsloser Produktions-Prozess</p>	<p>Reduziert 90% der Kosten der Persistenz-Entwicklung</p> <p>10% schnellere Marktreife für Ihre Anwendung</p>
<p>Einbettbar</p> <p>Keine Administration</p> <p>Automatische Schema-Erkennung</p> <p>600 KB Footprint</p>	<p>Läuft bis zu 55x schneller als konventionelle Systeme</p> <p>In großer Anzahl ohne lokale Administration einsetzbar</p>
<p>Support mehrerer Plattformen</p> <p>Nativ zu Java und .NET</p> <p>Embedded CPU, mobile device, Desktop- und Server Plattformen</p> <p>Läuft plattformübergreifend</p>	<p>Entwickeln von kompakter und echter objekt-orientierter Software</p> <p>Entwickeln von distribuierten, voll-synchronisierten Daten-Architekturen</p>
<p>Bringt mehr OO in die Datenbank</p> <p>Objekt-orientierte Replikation (dRS)</p> <p>Native Queries</p> <p>ObjectManager Browser</p>	<p>Weniger Fehler, bessere Refakturierbarkeit und Software-Langlebigkeit</p>



Die Eine-Zeile-Code-Datenbank spart Ihnen Zeit

Es ist so leicht: Fügen Sie db4o's Programmibliothek (.jar / .dll) in Ihre Entwicklungsumgebung ein, öffnen Sie eine Datenbankdatei und speichern Sie jedes beliebige Objekt – egal wie komplex – mit nur einer einzigen Zeile Code. Z.B. in Java:

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

Diese bisher unerreichte, einfache Handhabung führt zu drastisch verkürzter Entwicklungszeit.

Eliminieren Sie die gesamte Arbeit des Designs, der Implementierung und der Wartung des Datenbankschemas, denn das Klassenmodell ist das Datenbankschema. Beschleunigen Sie ihren Produktionsprozess, indem Sie die Notwendigkeit, jeglichen datenbankbezogenen Overhead wie Strings, XML und andere nicht-native Dateien anzupassen, eliminieren.

Mit db4o können Sie in weniger als 5 Minuten loslegen, unterstützt durch das beliebte, interaktive Formula-1 Tutorial.

Sie sparen sehr viel Zeit bei der Entwicklung ihrer Software.

Sie sparen jedes Mal noch mehr Zeit, wenn Sie ihre Software anpassen müssen, z.B. beim Code-Refactoring, beim Hinzufügen von neuen Features oder beim Wiederverwenden von Software-Komponenten. Das Ändern ihres Objektmodells beispielsweise, ist nicht nur extrem transparent, sondern auch kinderleicht, weil dank db4os nativer und nicht-intrusiven Natur, die Entwicklungsumgebung die gesamte Arbeit für Sie übernimmt! Sie brauchen keine Debugger, keinen Build-Prozess und Sie müssen sich auch keine Sorgen um bereits bestehende Installationen machen, da sich db4o um alle Objektmodifikationen Ihrer installierten Basis kümmert. Das Ändern von Software wird weniger zum Albtraum sondern mehr zum Vergnügen, indem es Sie produktiver denn je macht.

Im Wesentlichen macht db4o die Objektpersistierung genau so leicht wie den Einsatz gewöhnlicher Serialisierung. Gleichzeitig haben Sie dabei jedoch noch die volle Breite der Datenbank-Funktionalitäten. Dazu zählen beispielsweise Querying und – insbesondere – die Berücksichtigung von Objektänderungen ohne die Serialisierung aufzuheben.



Die einbettbare Datenbank

db4o wurde derart konzipiert, dass es – für den Endbenutzer komplett unsichtbar – in Clients oder anderen Softwarekomponenten eingebettet werden kann. Daher benötigt db4o keinen separaten Installationsmechanismus, es besteht aus nur einer kinderleicht einsetzbaren Programmbibliothek mit einem sehr kleinen Footprint von etwa 600KB. Weil db4o im selben Prozess wie Ihre Anwendung läuft, haben Sie volle Kontrolle über die Speicherverwaltung und können Geschwindigkeits-Profilung und Debugging über das gesamte System anwenden. Wenn Ihre Anwendung läuft, dann läuft auch die Datenbank – ohne Ausnahme.

Am Wichtigsten ist aber, dass db4o extrem flexibel hinsichtlich dem Update einer bestehenden Basis mit einem veränderten Objektmodell ist. Da db4o immer davon ausgeht, dass kein Datenbank Administrator anwesend ist, lässt es die Anwendung nahtlos vom alten zum neuen Objektmodell wechseln. Im Gegensatz zu anderen Datenbank-Lösungen, sei es relational oder eine nicht-native Objektdatenbank, benötigt db4o keine Datenmodell-Updateverwaltung – somit wird ein Haufen Arbeit und eine Fehlerquelle komplett eliminiert.

Portabilität and plattformübergreifender Einsatz

Nur wenige einbettbare Datenbanken laufen nativ auf so vielen objekt-orientierten Plattformen – und keine davon in heterogenen Umgebungen. db4os Leistungsfähigkeit ermöglicht es Ihnen, Anwendungen sowohl für unterschiedliche Plattformen zu entwickeln (z.B. im PDA-Markt), als auch in heterogenen Umgebungen: Mit Alias-Klassen, die die unterschiedlichen Namenskonventionen abstimmen, können alle Java und .NET db4o Instanzen (Clients oder Server) persistente Objekte teilen, ohne dabei Klassen auf dem Server einsetzen zu müssen.

db4o unterstützt Javas JDK 1.1.x bis 6.0 und läuft auf J2EE und J2SE. db4o läuft auch mit J2ME Dialekten, die Reflection unterstützen, wie beispielsweise CDC, PersonalProfile, Symbian, Savaje und Zaurus. Je nach Kundenwunsch, läuft db4o auch auf Dialekten ohne Reflection, wie z.B. CLDC und MIDP, einschliesslich RIM (Blackberry) und Palm OS.

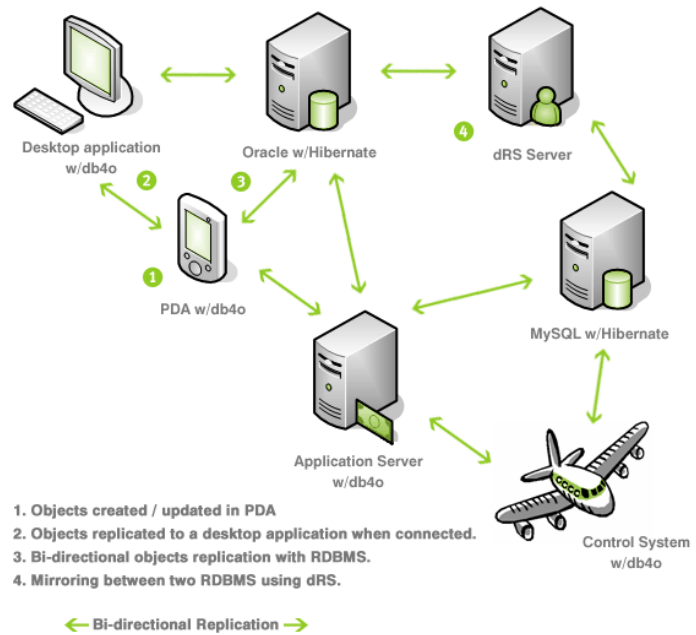
db4o läuft auf allen .NET Plattformen einschliesslich .NET, dem CompactFramework und Mono. Es unterstützt alle .NET Sprachen wie: C#, VB.NET, ASP.NET, Boo und Managed C++.



Partiell verbundene Clients und verteilte Datenarchitekturen

db4os primärer Einsatzbereich sind Standalone-Clients, wie z.B. in einem Smartphone oder in einem Auto. Jedoch sind die meisten Clients heutzutage wenigstens teilweise mit Middleware, Servern oder anderen Geräten verbunden. db4o bietet deshalb neben dem eingebetteten auch einen Client-Server-Modus und ist somit bereit, um in das Unternehmen oder den serverseitigen Software-Block eingegliedert zu werden.

db4os einzigartige objekt-orientierte Replikation (dRS) ermöglicht die einfache Daten-Synchronisierung zwischen db4o Datenbanken, relationalen Datenbanken wie Oracle oder MySQL, oder jeder anderen beliebigen Kombination. Relationale Datenbanken werden beim dRS mit dem objekt-relationalen Mapper Hibernate angesprochen. Die Implementierung ist wie immer kinderleicht: Einfach zwei Datenbankinstanzen verbinden um nach aktualisierten Objekten zu suchen und Objekte mit nur einer einzigen Zeile Code synchronisieren. Durch Eltern-Kind Beziehungen in den Klassen wird eine natürliche Beschreibung der Objektbestandteile, und derjenigen Teile die mit dem Eltern-Objekt repliziert werden müssen, festgelegt. Sie können auch Business Logic speichern, die notwendig ist um Synchronisationskonflikte innerhalb des Objekts zu lösen (in der Datenschicht, nicht der Anwendungsschicht). Dadurch können Sie „Smart Objects“ erstellen, die frei innerhalb von verteilten Architekturen bewegt werden können, z.B. Module die von unterschiedlichen Teams geschrieben wurden. Diese Objekte sind auch toleranter hinsichtlich eines Refactorings (weitere Informationen zum dRS gibt es hier: www.db4o.com/about/productinformation/features/drs.aspx).



Somit ermöglichen db4os verschiedene Betriebsarten und sein einzigartiges objekt-orientiertes Replikationssystem höchst leistungsfähige und effiziente verteilte Datenarchitekturen, welche typischerweise Voraussetzung für service-orientierte EDV sind.

Native Queries

Mit db4o Version 5 hat db4objects als Erster Native Queries (NQ) eingeführt. Native Queries führen den Industrietrend der Bereitstellung von sprachgesteuerten Datenbankabfragen an, entsprechend dem Ziel von Microsofts LINQ (.NET Language Integrated Queries) Projekt.

Im Gegensatz zur Verwendung von string-basierten APIs (wie SQL, OQL, JDOQL, EJBQL, und SODA), verwenden Entwickler mit NQs einfach die Programmiersprache selbst (z.B. Java, C# oder VB.NET) um auf die Datenbank zuzugreifen. Dadurch wird ein konstanter, produktivitäts-reduzierender Kontextwechsel zwischen Programmiersprache und Datenzugriffs-API vermieden.



Vergleichen Sie beispielsweise diese Native Query in C# mit .NET 2.0:

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

... oder in Java:

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

... dasselbe in 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);
```

Wie Sie sehen, eliminieren Native Queries alle Strings aus den Queries – sie sind 100% typsicher, 100% refakturierbar und 100% objekt-orientiert. Im obigen Beispiel würde JDOQL ihren Code akzeptieren, sogar wenn das „age“ Feld ein Datentyp wäre, und der Typ-Fehler würde nicht bemerkt werden, bis der Code ausgeführt wird. Mit der Unterstützung Ihrer IDE sind NQs komplett typsicher und akzeptieren von vornherein keine typ-fehlerhaften Queries.

Als Ergebnis werden Software-Entwickler durch den Datenzugriff mit NQs weitaus produktiver. Auch häufige Refactorings und/oder Veränderungen und Anpassungen der Objektmodelle werden vereinfacht. Mit nur einer einzigen Änderung könnten Sie z.B. das Feld „age“ in obiger NQ in „_age“ umbenennen. Falls Sie das in JDOQL (oder einem beliebigen anderen string-basierten API) versuchen, würde die Anwendung nicht laufen.

Weitere APIs, UI Browser und Reporting

Zwei zusätzliche objekt-orientierte Abfrage-APIs, Query by Example und S.O.D.A., bieten zusätzliche Abfragemöglichkeiten für spezifische Use Cases und/oder Legacy-Anwendungen.

Mit Query by Example (QbE) haben Sie ein extrem einfach zu bedienendes API zur Hand, das bestehende Setter benutzt, um Abfrage-Templates zu erstellen, z.B.:

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

S.O.D.A. ist ein leistungsfähiges, nodebasiertes Abfrage API zur dynamischen Abfrageerstellung während der Laufzeit. Es erlaubt das Auslösen jedes beliebigen Custom-Codes auf Servern, reduziert dadurch den Bandbreiten-Bedarf und beschleunigt die Ausführung der Abfrage.



Exportieren nach XML ist jetzt mit jeder Bibliothek möglich, die Java Objekte in XML umschreibt, wie z.B. XStream. Dies erlaubt die Integration von db4o in nachrichten-orientierte (SOA) Architekturen.

db4o stellt kein SQL Interface bereit, da Entwickler keinen Grund haben auf ihre Objekte direkt mit einem nicht-nativen API zuzugreifen. Trotzdem haben sie aus Kompatibilitätsgründen die Möglichkeit, mit Hilfe des db4o Replication Systems (dRS) ihre Objekte in jede beliebige relationale Datenbank zur weiteren Datenverarbeitung zu replizieren.

Der ObjectManager ist ein visuelles Tool zum Zugreifen, Navigieren, Abfragen und Bearbeiten von db4o Datenbankdateien, sogar dann, wenn die Anwendungsklassen nicht präsent sind. Hauptsächlich als Entwicklungstool zum Debugging gedacht, hilft es auch denjenigen, die neu im Umgang mit Objektdatenbanken sind, den Übergang vom relationalen in das objekt-orientierte Denken zu erleichtern.

Mehrere Dritthersteller bieten Tools für objekt-orientiertes Reporting an (siehe unten).



Spezifikationen | db4o V 6.0

Plattformen	
Java	<ul style="list-style-type: none"> • J2EE • J2SE • J2ME mit Reflection: CDC, PersonalProfile, Symbian, Savaje und Zaurus. On Demand: J2ME ohne Reflection (CLDC und MIDP), inklusive RIM/Blackberry und Palm OS • Servlet/JSP Framework • Java Web Start • Harmony
.NET	<ul style="list-style-type: none"> • .NET 1.x – 3.0 • CompactFramework 1.0 – 2.0 • Windows Mobile / PocketPC • Mono
Plattformübergreifend	<ul style="list-style-type: none"> • Verbindung zu .NET Server mit Java Client • Verbindung zu Java Server mit .NET Client
Sprachen	
Java	<ul style="list-style-type: none"> • JDK 1.1.x - JDK 6.0
.NET	<ul style="list-style-type: none"> • Alle .NET tauglichen Sprachen (C#, VB.NET, ASP.NET, Boo, Managed C++ etc.)
Commands	
Sessions	<ul style="list-style-type: none"> • Start und end
Datenbankdateien	<ul style="list-style-type: none"> • create, open, close und delete
Transaktionen	<ul style="list-style-type: none"> • Commit und Rollback
Objekte	<ul style="list-style-type: none"> • store, retrieve, update (inkl. cascaded), replicate, delete (inkl. cascaded)
Messaging	<ul style="list-style-type: none"> • TCP/IP
Transparenz	
Sprachkonstrukte	<ul style="list-style-type: none"> • Primitive Types • Strings • Arrays • Mehrdimensionale Arrays • Inner classes • Java/C# Collections



	<ul style="list-style-type: none"> • Klassen ohne Public Constructors • .NET structs • Blobs (ausserhalb der DB Datei gespeichert)
Nicht-intrusiv	<ul style="list-style-type: none"> • Ohne Ableitung von einer spezifischen Basisklasse • Ohne Implementierung eines spezifischen Interface • Ohne Änderungen des Quellcodes • Ohne Implementierung von Serializable
Private Fields	<ul style="list-style-type: none"> • Storable
File I/O	<ul style="list-style-type: none"> • Pluggable
Reflector	<ul style="list-style-type: none"> • Pluggable • Generic
Aliases	<ul style="list-style-type: none"> • Class Aliasing für Class-to-Class Mappings
Query Sprachen / APIs	
Objekt-orientiert	<ul style="list-style-type: none"> • Native Queries (NQ) • Query By Example (QbE) • S.O.D.A.
SQL	<ul style="list-style-type: none"> • Via Replikation zu vielen relationalen Datenbanken
XML	<ul style="list-style-type: none"> • Über Produkte von Drittherstellern (z.B. XStream)
Modus / Concurrency	
Operationsmodi	<ul style="list-style-type: none"> • Lokal • Client/Server
Threads	<ul style="list-style-type: none"> • Mehrere
Transaktionen	<ul style="list-style-type: none"> • Mehrere, parallel
Semaphores	<ul style="list-style-type: none"> • Verfügbar
Read-Only Modus	<ul style="list-style-type: none"> • Verfügbar
Skalierbarkeit und Performance	
Performance Benchmark	<ul style="list-style-type: none"> • Bis zu 55x schneller als Hibernate/MySQL
Skalierbarkeits-Beispiele	<ul style="list-style-type: none"> • Speichert 200,000 Objekte/Sekunde • Speichert 20,000 Klassen • Speichert 300,000 Objekte auf einem PDA
In Memory Modus	<ul style="list-style-type: none"> • Verfügbar
Client-Seite	<ul style="list-style-type: none"> • Single-Process Ausführung verfügbar
Server-Seite	<ul style="list-style-type: none"> • Server-Side Query Ausführung verfügbar



DB-aware Collections	<ul style="list-style-type: none"> • Verfügbar
Objekt Caching	<ul style="list-style-type: none"> • Verfügbar
Pagination	<ul style="list-style-type: none"> • Server-Side Cursors (lazy queries)
Indizierung	<ul style="list-style-type: none"> • BTree Feld Indizes • BTree Query Prozessor
Replikation	
db4o Replication Service (dRS)	<ul style="list-style-type: none"> • 100% objektorientiert: repliziert Objekte, nicht Tabellen • uni- und bi-direktional • db4o zu db4o • db4o zu relationalen Datenbanken (RDBMS), angesprochen durch Hibernate (nur Java) • Hibernate/RDBMS zu Hibernate/RDBMS (nur Java)
UUID	<ul style="list-style-type: none"> • Unique Universal Identity auf allen DB Instanzen
Synchronisierung	<ul style="list-style-type: none"> • Querying • Update • Delete • Conflict Resolution
Reporting	
Für .Java Objekte	<ul style="list-style-type: none"> • Actuate BIRT • Elixir Report • Jaspersoft JasperReports • Jinfonet JReport
Für .NET Objekte	<ul style="list-style-type: none"> • Microsoft Visual Studio 2005 - ReportViewer
Sicherheit und Verschlüsselung	
DB File Protection	<ul style="list-style-type: none"> • Passwort
DB File Verschlüsselung	<ul style="list-style-type: none"> • Simple database encryption • Pluggable File I/O für eigene Verschlüsselung • eXtended Tiny Encryption Algorithm (XTEA) - 128-bit Schlüssel
Availability, Reliability, Zero-Admin	
Transaktionen	<ul style="list-style-type: none"> • ACID • Commit-Recovery bei Systemabstürzen
Thread Sicherheit	<ul style="list-style-type: none"> • Verfügbar
Automatic Recovery	<ul style="list-style-type: none"> • Verfügbar



Online-Backup	<ul style="list-style-type: none"> • Verfügbar
Free Space Management	<ul style="list-style-type: none"> • Verfügbar
Internationalisierung	
Unicode	<ul style="list-style-type: none"> • Verfügbar
Refactoring	
Schema Versioning	<ul style="list-style-type: none"> • Automatische Erkennung und Wartung
Schema Merge	<ul style="list-style-type: none"> • Verfügbar
Umbenennen	<ul style="list-style-type: none"> • Klassen, Felder • Zugriff auf die Werte von gelöschten Feldern über API • Über API oder die ObjectManager UI
Speicher- und Dateigrösse	
DB Bibliothek Footprint	<ul style="list-style-type: none"> • ~600 KB
Minimaler RAM Footprint	<ul style="list-style-type: none"> • ~1 MB
Maximale DB Datei Grösse	<ul style="list-style-type: none"> • 254 GB / Datenbank Datei
Klassengrösse	<ul style="list-style-type: none"> • Keine Erhöhung bei Persistierung
Instanzgrösse	<ul style="list-style-type: none"> • Keine Erhöhung bei Persistierung