



Version 7.0 | Java and .NET

db4o Open Source Object Database

db4o is an open source object database that enables faster time to market on Java and .NET platforms. It offers unprecedented flexibility by automatically handling changes in structure (schema) and dynamic events (triggers) updates over the lifetime of the product. Db4o allows the development process to focus on the business logic, while cutting down complexity, and achieving unprecedented levels of performance. The unique design of db4o's native object database engine makes it the ideal choice to be embedded in equipment and devices, in packaged software running on mobile or desktop platforms, or in real-time control systems – in short: in all Java and .NET environments, where no database administrator is present.

db4o is the open source dynamically adaptable object database that enables Java and .NET developers to slash development time and costs and achieve unprecedented levels of performance.

The unique design of db4o's native object database engine makes it the ideal choice to be embedded in equipment and devices, in packaged software running on mobile or desktop platforms, or in real-time control systems – in short: in environments where no DBA is present.

Relational Databases, Object-Relational Mappers and db4o's Object Database



All object-oriented software developers are familiar with the difficulty transitioning from object-oriented thinking to relational persistence. So far, they have been forced to choose between speed and object-orientation: SQL access is fast, but laborious, requiring a great deal of additional code. Object-relational mappers offer a convenient bridge, but they seriously degrade performance and shift the development focus from the business logic to dealing with the translation logic and its restrictions.

db4o eliminates the OO-versus-performance tradeoff: it allows you to store even the most complex object structures with ease, while achieving highest level of performance. Database benchmarks show db4o to be up to 55 times faster than Hibernate and MySQL, a popular object-relational mapper and relational SQL database stack.

Often the reason for using relational databases today is legacy, i.e. retaining old enterprise data and the set of existing applications relying on it. But beyond the server-centric persistence, there are a multitude of device, mobile, and desktop applications for which conventional database technology falls short. db4o's technology ensures new levels of flexibility, adaptability, performance, functionality, and cost effectiveness.

Still, by means of the [db4o Replication System \(dRS\)](#), developers can remain fully data compatible with legacy RDBMSs such as Oracle and MySQL.



Product - Application, Unique Features, Benefits

db4o provides a full-featured, embeddable database engine for equipment, mobile devices, desktop and server platforms in object-oriented environments. Where relational databases fall short in providing zero-administration, small footprint, smooth synchronization, and transparent, automatic upgradeability, db4o is the answer. Native to Java and .NET, db4o's single programming library (.jar / .dll) easily integrates into the application and performs highly reliable and scalable persistence tasks with just one single line of code, no matter how complex the object structures are.

Thus db4o brings the following business advantage to your products:

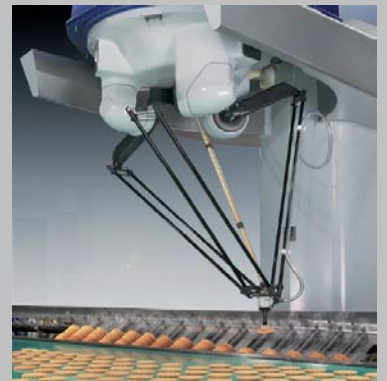
- Stay completely focused on the business logic, as db4o persists your native data structure and logic without constraints imposed by conversion process to tables or O-R mapper.
- Simplify the process and increase quality by cutting down the number of organizations and experts that need to be involved in your data persistence, by not requiring a DBA, SQL or O-R mapping expertise, and eliminating the associated coordination and integration between groups.
- Dynamically adjust your "schema" from release to release, or during development, as db4o takes care of changes automatically with no overhead.
- Design and easily modify "triggers" dynamically, based on application logic using db4o callback mechanism, instead of rigid server-based "triggers" used in Relational databases.
- Cross platform database between Java and .NET applications
- Cut down time to market due to the following engineering advantages:
 - Eliminate tools and code necessary for object-relational mapping, which is proven to drive up code complexity and resource-consumption while inhibiting performance and maintainability. With db4o, users gain up to 90% time and cost savings for software development related to persistence.
 - Build applications with seamlessly integrated transactional data storage that does not need runtime administration, and is highly reliable and much faster than conventional or proprietary database engines.
 - Benefit from pure object-oriented paradigms in native Java or .NET, benefitting from the Object Oriented development environment, and the ability to deploy more complex, natural and feature-rich object models without driving up cost and resource consumption.
 - Change, refactor and reuse software components with the ability to add new software features without breaking legacy objects, or incurring upgradeability or conversion related high costs - providing the nimbleness required to stay ahead of the competition.

db4o Case Studies

www.db4o.com/about/customers



AVE High Speed Trains



BOSCH's Packaging Robots



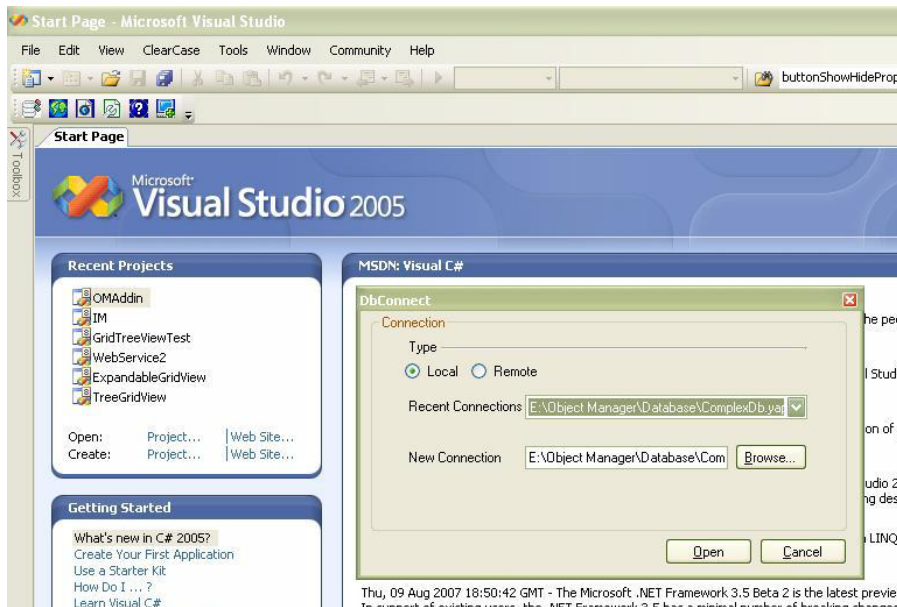
Seagate's Mirra Server



Eastern Data's Mobile Apps

db4o is driven by the world's largest community of its kind, with 40,000 registered Java and .NET developers and counting. The product has been downloaded nearly 2,500,000 times and been deployed successfully in transportation, networks, natural sciences, trading, SCADA, industrial, consumer and enterprise applications. Users and customers of db4o currently come from 170 different countries, from Albania to Zimbabwe, and range from world-class leaders like Boeing, Bosch, Intel, Ricoh and Seagate to a wide range of highly innovative start-up companies.

Built on new object database technology, db4o is currently the only database that is native to both Java and .NET -- providing cross-platform portability that liberates users from proprietary vendors' high licensing fees. db4o provides a wide array of unique capabilities which are unmatched in the database world such as, benefitting from OO programming environments,, object-oriented replication (dRS), object-oriented queries (LINQ, Native Queries, QbE, S.O.D.A.), and ObjectManager Enterprise for browsing and maintaining database files,.



db4o's ObjectManager Enterprise can be used to query, browse and maintain db4o database files

Above all, db4o is very easy to learn, implement, and use. db4o's powerful database engine allows users to store objects with just one line of code, slashing development time and cost for the persistence layer to a minimum.

These benefits are quadrupled when it comes to changing software in order to maintain, improve, add new features or re-use software components. db4o automates refactoring through the development environment, because all non-native APIs and strings are eliminated. If updates are deployed to the installed base, schema versioning handles object model changes transparently and automatically without the need for conversion processes. Developers need no enhancer, no pre-compiling, and no post-compiling processes. And even replication processes are widely agnostic to software changes, avoiding breaking distributed data architectures.



New users will find that they can easily download and get started with db4o in just 5 minutes or less. A comprehensive interactive tutorial comes with the download, helping developers to get started and easily transition from relational to object-oriented thinking.

db4o Whitepapers available for free download, e.g.:

- The Database Behind the Brains
- Complex Object Structures, Persistence, and db4o
- Enabling the Mobile Enterprise with db4o
- db4objects and the Dual Licensing Model

www.db4o.com/about/productinformation/whitepapers

Benchmarks

www.db4o.com/about/productinformation/benchmarks



Features and Benefits

Key Features	Key Benefits
<p>The One-Line-of-Code-Database One line of code stores any object Class model = object schema Smooth production process</p> <p>Embeddable Zero administration Automatic schema versioning 1 MB footprint</p> <p>Multiple platform support Native to Java and .NET Embedded CPU, mobile device, desktop, and server platforms Runs cross-platform</p> <p>Brings more OO to the database Object-oriented replication (dRS) Native Queries ObjectManager tool LINQ support</p>	<p>40% faster to market with your application</p> <p>Full ACID transactional capabilities</p> <p>Slashes 90% of cost to develop persistence</p> <p>Runs up to 55x faster than conventional systems</p> <p>Deployable in large volumes without local administration</p> <p>Build lean and purely object-oriented software</p> <p>Build distributed, fully synchronized data architectures</p> <p>Fewer errors, better maintainability and software longevity</p>



The One-Line-of-Code Database Saves You Time

It's as easy as this: Drop db4o's single programming library (.jar /.dll) into your development environment, open a database file and store any object - no matter how complex - with just one line of code, e.g., in Java:

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

This unmatched ease-of-use results in drastically reduced development time.

Eliminate the entire work of designing, implementing and maintaining the database schema, because the class model is the database schema. Eliminate the need to manage any database-related overhead such as strings, XML, or other non-native files that need post- or pre-compiling or enhancers and consequently slow down your production process.

You are up and running in less than 5 minutes, supported by the acclaimed interactive Formula-1 Tutorial.

You save a lot of time when writing your software.

You save even more time whenever you need to change your software, e.g. refactoring code, adding new features, or reusing software components. Changing your object model, for instance, is not only extremely transparent, but also fool-proof because the native and non-intrusive nature of db4o lets the development environment do all the work for you! You need no debuggers, no build process, and you don't need to worry about existing deployments, because db4o takes care of any object modifications for your entire installed base. Changing software becomes less of a nightmare and more of a pleasure, making you ever more productive.

In essence, db4o makes it as easy to persist objects as it is using plain serialization, but also gives you the full breadth of database functionalities such as querying, transactions and - notably - allows for changing object models without breaking the serialization.

The Embeddable Database

db4o is designed to be embedded in clients or other software components completely invisible to the end user. Thus, db4o needs no separate installation mechanism, but comes as just one easily deployable library with a very low footprint of around 1MB. Because db4o runs in the same process as your application, you have full control over memory management and can perform speed profiling and debugging over the entire system. If your application is running, your database is running - there is no exception.



Most importantly, db4o is extremely flexible when it comes to updating an existing installed base with a changed object model. db4o always assumes the absence of a database administrator and hence lets the application seamlessly switch from the old to the new object model. Unlike any other database solution, be it relational or a non-native object database, db4o does not need any form of data model update management - a whole work package and source of errors that is completely eliminated.

Portability and Cross-Platform Deployment

Few embeddable database products run natively on so many object-oriented platforms - and none across those platforms in heterogeneous environments. db4o's capability enables you to develop applications for deployment not only on various platforms (e.g. in the PDA market) but also in heterogeneous environments: With class aliases reconciling the different naming conventions, any Java and .NET db4o instances (clients or servers) can share persisted objects without the need to deploy classes to the server.

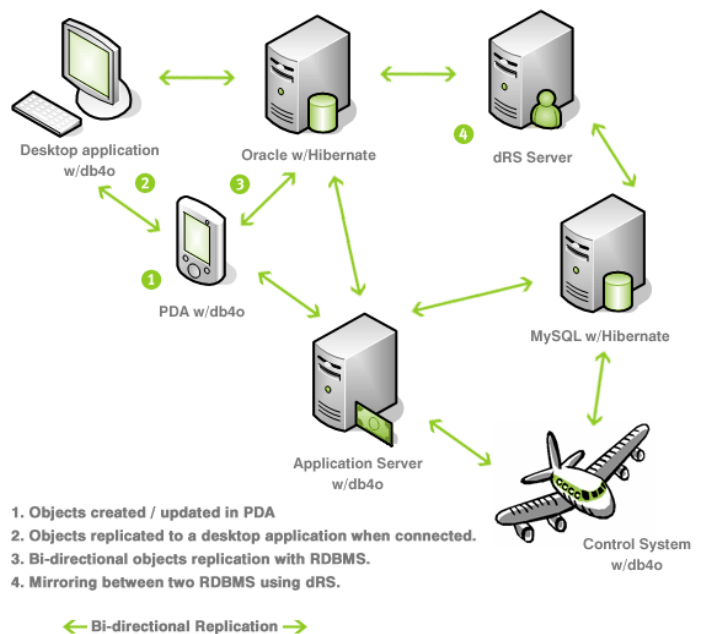
db4o supports Java's JDK 1.1.x through 6.0 and runs on J2EE and J2SE. db4o also runs with J2ME dialects that support reflection, such as CDC, PersonalProfile, Symbian, Savaje and Zaurus. Moreover db4o is also available as an OSGi bundle.

db4o runs on all .NET platforms including .NET, the CompactFramework and Mono, supporting all managed .NET languages such as: C#, VB.NET, ASP.NET, Boo, and Managed C++. db4o also features a LINQ provider making your queries more native and effortless than ever on the .NET platform.

Partially Connected Clients and Distributed Data Architectures

While db4o's prime applications are in standalone clients, such as a smartphone or a car, most of those clients today are at least partially connected to middleware, servers or other devices. db4o therefore not only provides an embedded mode, but also a client/server-mode ready to move into the enterprise or server-side software stack.

db4o's unique object-oriented replication (dRS) functionality allows for easy synchronization of data between db4o databases, relational databases such as Oracle or MySQL, or any combination thereof. Relational databases are wrapped by the Hibernate object-relational mapper. Implementation is, again, as easy as it gets: Connect two database instances to search for updated objects and synchronize objects with just one line of code. Parent-child relationships in classes provide a natural description of what makes up an object and what needs to be replicated along with a parent object. Also, you can store business logic necessary to resolve synchronization conflicts within the object (in the data, not the application layer) to create "smart objects" that can move around freely in





distributed architectures, such as between modules written by different teams. These are also more tolerant when it comes to refactoring (more on dRS here: www.db4o.com/about/productinformation/features/drs.aspx.)

All in all, db4o's diverse execution modes and its unique object-oriented replication functionality allow the highly powerful and efficient distributed data architectures typically required for service-oriented computing.

Native Queries

Since db4o Version 5, db4objects has been the first to implement Native Queries (NQ), leading the industry trend to provide database querying with OO programming language semantics, validated by the recently released Microsoft's LINQ (.NET Language Integrated Queries).

Rather than using string-based APIs (such as SQL, OQL, JDOQL, EJBQL, and SODA), NQs allow developers to simply use the programming language itself (e.g., Java, C#, or VB.NET) to access the database and thus benefitting from compile-time type checking, the full expressive power of OO-languages, and the great convenience of advanced development environments.

For example, compare this Native Query in C# for .NET 2.0:

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

... or in Java:

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

... with the same 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);
```

As you can see, Native Queries eliminate all strings from queries – they are 100% type-safe, 100% refactorable, and 100% object-oriented. For instance, in the above, JDOQL would accept your code, even if the “age” field was a date type and the type mismatch error would not be thrown until you execute the code. With the help of your IDE, NQs are entirely type-safe and would not accept type-mismatched queries in the first place.

As a result, data access with NQs makes software developers much more productive. It also facilitates frequent refactorings and/or changes and customization of object models. With just

one change, for instance, you could successfully rename the “age” field in the above NQ into “_age”. Try this with JDOQL (or any other string-based API) and the application would break.

The same concept applies to our LINQ provider which allows you to smoothly move between Relational db and db4o, for a truly complimentary combination. db4o allows using all the constructs of Microsoft’s Language Integrated Queries (LINQ). db4o LINQ syntax is provided for .NET developers and aimed to make writing db4o code even more native and effortless. Queries like this:

```
IEnumerable<Pilot> result =  
    from Pilot p in container  
    where p.Name.StartsWith("Test") && p.Points > 2  
    select p;
```

are perfectly valid within db4o.



Integration, Other APIs, ObjectManager, and Reporting

Two additional object-oriented query APIs, Query by Example and S.O.D.A., offer additional query options for specific use cases and/or legacy applications:

With Query by Example (QbE), you have an extremely easy-to-use API that employs existing setters to create query templates, e.g.:

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

S.O.D.A. is a powerful node-based query API for dynamic query creation at runtime. It allows triggering any custom code on servers, thereby reducing bandwidth and speeding query execution.

Export to XML is easily achieved now with any library that writes Java objects to XML, such as Xstream. It allows integrating db4o into message-oriented architectures.

db4o does not provide an SQL interface because developers have no need to directly access their objects with a non-native API. However, for compatibility purposes, developers can replicate their objects by means of the db4o Replication System (dRS) into any relational database for further data processing.

ObjectManager Enterprise (OME) is a new tool exclusively for dDN Enterprise subscribers. OME comes as a plug in for Visual Studio or Eclipse or as a standalone.

OME is designed for db4o database browsing and maintenance and includes:

- Classes view: flat and hierarchical filtered structure of persistent classes
- Single class detailed view: fields, base- and subclasses and interfaces, statistics (number of objects in the database)
- Single object detailed tree view (field objects as nodes)
- Query Builder: allows to create complex queries using object fields on different levels of the hierarchy joined together with logical operators
- Multiple objects list view as a query result
- History view: recent queries or objects
- Favorites view: user-defined frequently used queries or objects (coming)
- Access to the XtremeConnect session scheduler and the dDN support case management

Several third-parties provide tools for object-oriented reporting (see below).



Specifications | db4o V 7.0

Platforms	
Java	<ul style="list-style-type: none"> • J2EE • J2SE • J2ME with reflection: CDC, PersonalProfile, Symbian, JavaFX Mobile and Zaurus; on demand: J2ME w/o reflection (CLDC and MIDP) , including RIM/Blackberry and Palm OS • Android (Open Handset Alliance) • Harmony • Supported Frameworks: Spring, OSGi, Tomcat, DataNucleus
.NET	<ul style="list-style-type: none"> • .NET (1.0, 2.0, 3.0, 3.5) • CompactFramework 1.0 – 2.0 • Windows (XP, Vista) • Windows Mobile / PocketPC • Mono • Supported Frameworks: Castle, Eiffel, Spring.net
Cross-Platform	<ul style="list-style-type: none"> • Connect to .NET server with Java client • Connect to Java server with .NET client
Languages	
Java	<ul style="list-style-type: none"> • JDK 1.1.x - JDK 6.0
.NET	<ul style="list-style-type: none"> • All managed .NET languages (C#, VB.NET, ASP.NET, Boo, Managed C++ etc.)
Commands	
Sessions	<ul style="list-style-type: none"> • Start and end
Database files	<ul style="list-style-type: none"> • Create, open, close, and delete
Transactions	<ul style="list-style-type: none"> • Commit, and Rollback
Objects	<ul style="list-style-type: none"> • Store, retrieve, update (incl. cascaded), replicate, delete (incl. cascaded)
Messaging	<ul style="list-style-type: none"> • TCP/IP
Transparency	
Language constructs	<ul style="list-style-type: none"> • Primitive types • Strings • Arrays • Multi-dimensional arrays • Inner classes

	<ul style="list-style-type: none"> • Java/C# collections • Classes without public constructors • .NET structs • Blobs (stored outside of DB file)
Non-Intrusive	<ul style="list-style-type: none"> • Without deriving from a specific base class • Without implementing a specific interface • Without modifications to source code • Without implementing 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 for class-to-class mappings
Query Languages / APIs	
Object oriented	<ul style="list-style-type: none"> • Native Queries (NQ) • LINQ • Query By Example (QbE) • S.O.D.A.
SQL	<ul style="list-style-type: none"> • Via replication to many relational databases
XML	<ul style="list-style-type: none"> • With Third-Party products (e.g., Xstream)
Modes / Concurrency	
Operation Modes	<ul style="list-style-type: none"> • Local • Client/Server
Threads	<ul style="list-style-type: none"> • Multiple
Transactions	<ul style="list-style-type: none"> • Multiple, parallel
Semaphores	<ul style="list-style-type: none"> • Available
Read-Only Mode	<ul style="list-style-type: none"> • Available
Scalability and Performance	
Performance benchmark	<ul style="list-style-type: none"> • Up to 55x faster than Hibernate/MySQL
Examples of Scalability	<ul style="list-style-type: none"> • Stores 200,000 objects/second • Stores 300,000 objects on a PDA
In Memory Mode	<ul style="list-style-type: none"> • Available
Client-side	<ul style="list-style-type: none"> • Single-process execution available



Server-side	<ul style="list-style-type: none"> • Server-side query execution available
DB-aware Collections	<ul style="list-style-type: none"> • Available
Object Caching	<ul style="list-style-type: none"> • Available
Pagination	<ul style="list-style-type: none"> • Server-side cursors (lazy queries)
Indexing	<ul style="list-style-type: none"> • BTree field indexes • BTree query processor
Replication	
db4o Replication Service (dRS)	<ul style="list-style-type: none"> • 100% object-oriented: simply replicate objects, not tables • Uni- and bi-directional • db4o to db4o • db4o to relational databases (RDBMS), wrapped in Hibernate (Java only) • Hibernate/RDBMS to Hibernate/RDBMS (Java only)
UUID	<ul style="list-style-type: none"> • Unique Universal Identity over all DB instances
Synchronization	<ul style="list-style-type: none"> • Querying • Update • Delete • Conflict resolution
Reporting	
For .Java objects	<ul style="list-style-type: none"> • Actuate BIRT • Elixir Report • Jaspersoft JasperReports • Jinfonet JReport
For .NET objects	<ul style="list-style-type: none"> • Microsoft Visual Studio 2005 - ReportViewer
Security and Encryption	
DB file Protection	<ul style="list-style-type: none"> • Password
DB file Encryption	<ul style="list-style-type: none"> • Simple database encryption • Pluggable File I/O for custom encryption • With Third-Party products (e.g., XTEA)
Availability, Reliability, Zero-Admin	
Transactions	<ul style="list-style-type: none"> • ACID • Commit-recovery on system failures
Thread Safety	<ul style="list-style-type: none"> • Available

