

db4o

Version 8.0 | Java and .NET

Open Source Object Database

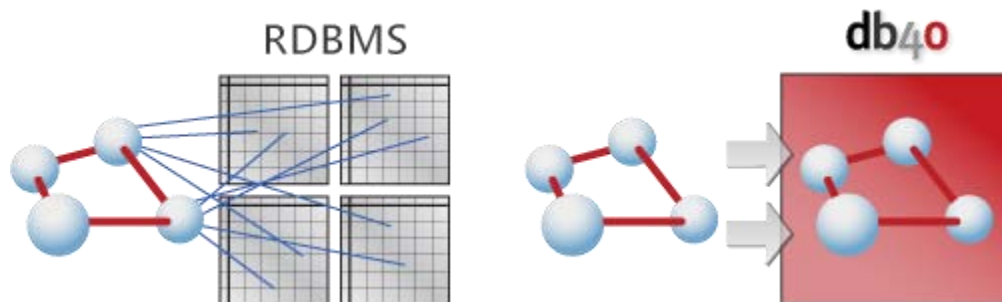
db4o is an open source database, the most powerful and yet simple to use database on the planet. Its power comes from allowing arbitrarily complex, application defined, class models to also represent your database schema. This eliminates time consuming and tedious mapping code and returns wasted mapping CPU processing to the application for raw performance.

Only db4o can bring Unmatched Agility and Lightning Speed to your team's product and development process, enabling faster time to market on Java and .NET platforms.

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 device, mobile or desktop platforms, or in real-time control systems – in short: in all Java and .NET environments where no database administrator 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 for simple information models, but laborious and slow when the models become complex, requiring a great deal of additional code. Object-relational mappers offer a convenient bridge to deal with model complexity, but they still 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 the highest level of performance. Popular open source database benchmark, PolePosition, shows the db4o 8.0 release to be up to 100 times faster than Hibernate and MySQL for non-trivial use cases.

By means of the db4o Replication System (dRS), a runtime feature allowing peer-to-peer replication with enterprise grade databases, developers can remain fully data compatible with legacy RDBMSs such as Oracle and MySQL and high end object database servers such as the Versant Object Database (VOD) system.

Product - Application, Unique Features, Benefits

When relational databases fall short in providing zero-administration, small footprint, smooth synchronization, and seamless refactoring, 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.

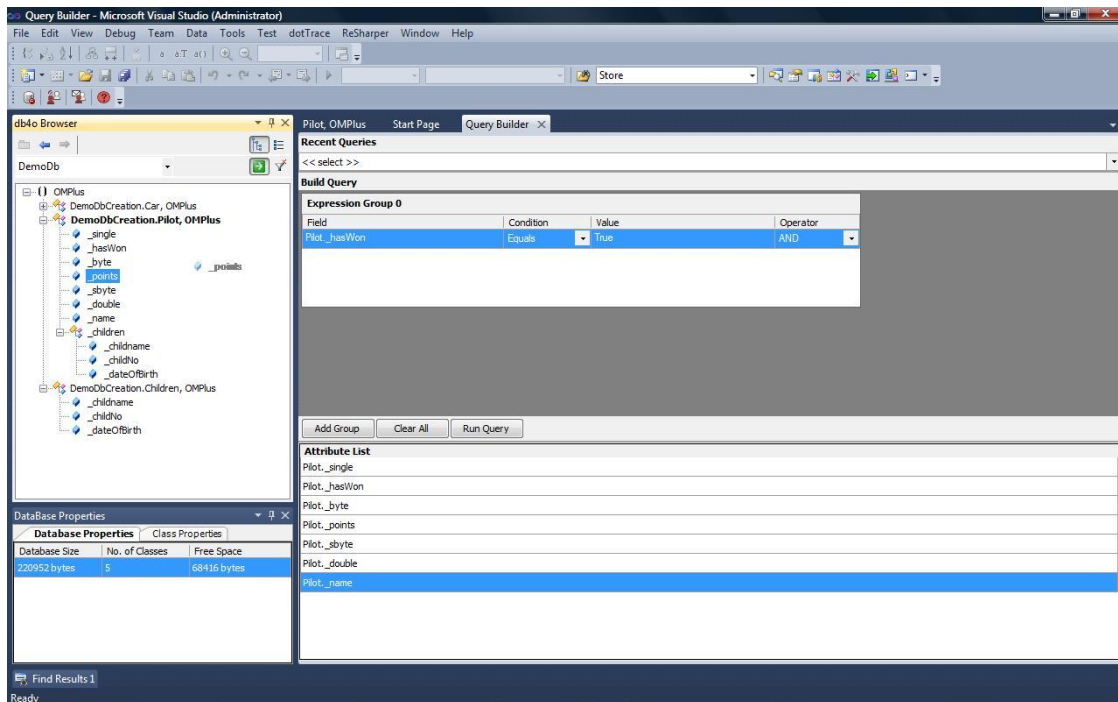
db4o brings the following business advantage to your products:

- Stay completely focused on the business logic, as db4o makes your native data structure persistent without constraints imposed by a 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.
- Design and easily modify “triggers” dynamically, based on application logic using db4o callback mechanism and standard language listener interfaces, instead of rigid server-based “triggers” used in Relational databases.
- Use Transparent Persistence (TP) to significantly simplify handling of deep object graphs in db4o based applications by keeping track of changes and storing all modified objects automatically on commit time.
- 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 40% time and cost savings for software development related to persistence.
 - Build applications with seamlessly integrated transactional data storage that does not need production 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, benefiting from Object Oriented development environments, 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 costs - providing the nimbleness required to stay ahead of the competition.

db4o is driven by the world's largest community of its kind, with nearly 100,000 registered Java and .NET developers and growing. The product has been downloaded nearly 3 million times and been deployed successfully in transportation, networks, natural sciences, trading, SCADA, industrial, consumer and other mission critical applications. Users and customers of db4o currently come from 170 different countries, from Albania to Zimbabwe, and range from world-class leaders like Boeing, CISCO, Indra, Lockheed Martin, Northrop Grumman, Siemens, and Sony to a wide range of highly innovative start-up companies.

Built on next generation object database technology, db4o provides a wide array of unique capabilities which are unmatched in the database world including: benefiting from OO programming environments, object-oriented peer-to-peer replication (dRS), object-oriented queries (LINQ, Native Queries, QbE, S.O.D.A.) and ObjectManager Enterprise technology 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 magnified when it comes to evolving 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. Therefore, you can dynamically adjust your “schema” from release to release, or during development, as db4o takes care of schema changes automatically. When you're deploying an update to your application this translates into no data migration or conversion on the existing database.

A comprehensive interactive tutorial comes with the download, new users will find that they can easily download and get started with db4o in just 5 minutes or less without polluting domain code with database related operations.

db4o Whitepapers are available for free download

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

<http://www.db4o.com/g/whitepapers.aspx>

db4o Benchmarks

<http://www.db4o.com/g/benchmarks.aspx>

Features and Benefits

Key Features	Key Benefits
<p>The One-Line-of-Code-Database</p> <ul style="list-style-type: none"> • One line of code stores any object • Class model = object schema • Smooth production process <p>Embeddable</p> <ul style="list-style-type: none"> • Zero administration • Automatic schema versioning • ~1 MB footprint <p>Multiple platform support</p> <ul style="list-style-type: none"> • Native to Java and .NET • Embedded CPU, mobile device, desktop, and server platforms • IDE friendly • 64bit compatible <p>Brings more OO to the database</p> <ul style="list-style-type: none"> • Object-oriented replication (dRS) to/from db4o, relational DBs and VOD • LINQ support • Native Queries • Object Manager Enterprise - exploration and querying tool 	<p>40% faster to market with your application</p> <p>Full ACID transactional capabilities</p> <p>Slashes 40% of cost to develop persistence</p> <p>Build lean and pure object-oriented software</p> <p>Deployable in large volumes without local administration</p> <p>Build asynchronously-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 = Db4oEmbedded.openFile("car.yap");  
    db.store(car);  
    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 to use 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 application.

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 1 MB. 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.

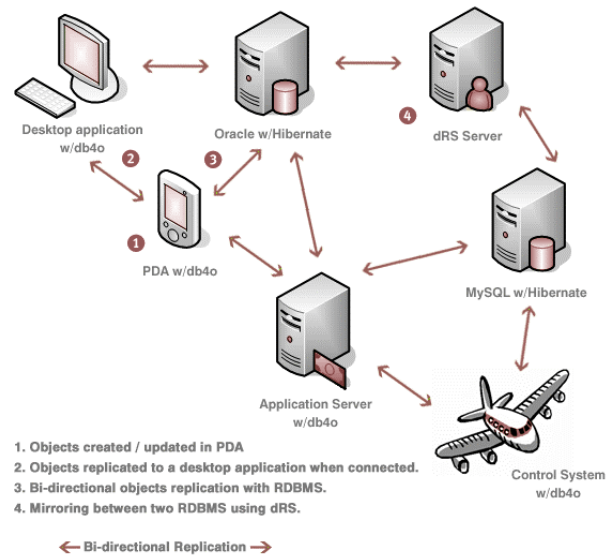
db4o supports Java's JDK 5+ and runs on J2EE and J2SE. db4o also runs with J2ME dialects that support reflection (such as the CDC profile). Moreover db4o is fully supported on the Android OS and is also available as an OSGi bundle.

db4o runs on all .NET platforms including .NET and the Compact Framework, supporting all managed .NET languages such as: C#, VB.NET, ASP.NET, Boo, and Managed C++. Moreover db4o is also supported under Silverlight which makes it suitable for Windows Phone 7 application development. 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.

In order to boost connectivity db4o provides a unique object-oriented replication system (dRS) that allows for easy synchronization of data between db4o databases, Versant Object Database (VOD) instances and relational databases such as Oracle or MySQL. 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. Information on dRS can be found here: <http://www.db4o.com/g/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 and CEP (complex event processing).

Queries in db4o

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

Rather than using string-based APIs (such as SQL, OQL, JDOQL, JPAQL, 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 benefiting 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 3.5:

```
IList<Student> students =
    from Student student in container
    where student.Age < 20 && student.Grade == gradeA
    select student;
```

... or in Java:

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

As you can see, Native Queries eliminate all strings from queries – they are 100% type-safe, 100% refactorable, and 100% object-oriented. But the JDOQL version...

```
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);
```

...would compile and try to run this query, even if the “age” field was a date type and a type mismatch exception 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("Michael") && p.Points > 2
    select p;
```

...are perfectly valid within db4o.

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

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

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

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

Integration, Other APIs, ObjectManager, and Reporting

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 free database exploration tool that comes as a plug in for Visual Studio or Eclipse.

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 premium pairing service and the dDN support case management

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

Specifications

Platforms	
Java	<ul style="list-style-type: none"> ✓ J2EE ✓ J2SE ✓ Android ✓ Compatible Frameworks: Spring, OSGi, DataNucleus, Restlet, Griffon, Guice
.NET	<ul style="list-style-type: none"> ✓ .NET (3.5, 4.0) ✓ CompactFramework (2.0, 3.5) ✓ Windows (XP, Vista) ✓ Windows Mobile / PocketPC / Windows Phone ✓ Mono (partial) ✓ Silverlight ✓ Compatible Frameworks: Castle, Eiffel, Spring.net

Languages	
Java	<ul style="list-style-type: none"> ✓ JDK 5+ ✓ Scala
.NET	<ul style="list-style-type: none"> ✓ C# ✓ Visual Basic

Commands	
Sessions	✓ Start and end
Database files	✓ Create, open, close, and delete
Transactions	✓ Commit, and Rollback
Objects	✓ Store, retrieve, update (incl. cascaded), replicate, delete (incl. cascaded)
Messaging	✓ TCP/IP

Transparency	
Language constructs	<ul style="list-style-type: none"> ✓ Primitive types ✓ Strings ✓ Arrays ✓ Multi-dimensional arrays ✓ Inner classes ✓ 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	✓ Storable

File I/O	✓ Pluggable
Reflector	✓ Pluggable ✓ Generic
Aliases	✓ Class aliasing for class-to-class mappings
Transparent Persistence	✓ Programmatic or via byte-code instrumentation

Query Languages / APIs

Object oriented	✓ LINQ ✓ Native Queries (NQ) ✓ Query By Example (QbE) ✓ S.O.D.A.
SQL	✓ Only via replication to many relational databases ✓ With Third-Party products (e.g. DataNucleus' Sql4o)
XML	✓ With Third-Party products (e.g. XStream)

Modes / Concurrency

Operation Modes	✓ Local ✓ Client/Server
Threads	✓ Multiple
Transactions	✓ Multiple, parallel
Semaphores	✓ Available
Read-Only Mode	✓ Available

Scalability and Performance

Performance benchmark	✓ Up to 55x faster than Hibernate/MySQL ✓ See PolePosition open-source benchmark http://www.polepos.org
In-Memory Mode	✓ Available
Client-side	✓ Single-process execution available
Server-side	✓ Server-side query execution available
DB-aware Collections	✓ Available
Object Caching	✓ Available
Pagination	✓ Server-side cursors (lazy queries)
Indexing	✓ BTree field indexes
BTree Query Processor	✓ Big Sets

Replication

db4o Replication Service (dRS)	✓ 100% object-oriented: simply replicate objects, not tables ✓ Uni and bi-directional ✓ db4o to db4o ✓ db4o to relational databases (RDBMS), via Hibernate (Java only) ✓ db4o to Versant Object Database (VOD, Java only)
---------------------------------------	---

UUID	✓ 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	✓ Microsoft Visual Studio ReportViewer

Security and Encryption	
DB file Encryption	<ul style="list-style-type: none"> ✓ 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	✓ Available
Automatic Recovery	✓ Available
Online-Backup	✓ Available
Free Space Management	✓ Defragmentation API available

Internationalization	
Unicode	✓ Available

Refactoring	
Schema Versioning	✓ Automatic recognition and maintenance
Renaming	<ul style="list-style-type: none"> ✓ Classes, fields by API ✓ Access to values of removed fields by API or ObjectManager UI

Memory and File Size	
DB library footprint	✓ ~1 Mb
Minimal RAM footprint	✓ Application-dependent; deterministic; typically < 3 MB
Maximum DB file size	✓ 254 GB / database file