

eXtremeDB Java Native Interface (JNI)

eXtremeDB JNI provides the fastest database solution in Java. Developers access eXtremeDB's powerful features entirely from within Java applications, while working with 'POJOs.'

eXtremeDB, the real-time embedded database for devices that are eXtremely innovative

Overview

McObject provides a Java Native Interface (JNI) for the eXtremeDB database system, offering the fastest possible DBMS solution in Java. The JNI reduces development time and slashes latency in Java embedded systems and in real-time enterprise applications (e.g. finance, e-commerce, and social networks), while affording the ease of working with plain old Java objects (POJOs).

eXtremeDB JNI delivers the speed of compiled C/C++ for performance-intensive database operations, with the convenience of using a familiar language. eXtremeDB's breakthrough performance stems from a streamlined architecture that manages data entirely in main memory, eliminating file I/O, cache management and other overhead found in database systems that incorporate disk storage.

```

class Address
{
    public String city;
    public String street;
    public String phone;
    public String fax;

    public Address(String city, String
street, String phone, String fax) {
        this.city = city;
        this.street = street;
        this.phone = phone;
        this.fax = fax;
    }
}

class Employee
{
    public String name;
    public int age;
    public long salary;
    public long companyID;
    Address address;
    public byte[] photo;
}
    
```

Java Class Declarations

Figure 1. Class definitions in a Java application.

With eXtremeDB JNI, developers work with the real-time database entirely from within Java. There is no external data definition language, and no requirement to compile a database schema.

Instead, the JNI uses Java's reflection capability to discover database classes and their fields defined in the application, via class definition syntax (see Figures 1 and 2). This streamlines coding, enabling developers to work entirely with Java objects.

Optionally, the Java class definitions can be used to generate a schema that enables C/C++ and Java programs to use the same database.

```

class Address
{
    @Dimension(20)
    public String city;
    @Dimension(60)
    public String street;
    @Dimension(20)
    public String phone;
    @Dimension(20)
    public String fax;

    public Address(String city, String
street, String phone, String fax) {
        this.city = city;
        this.street = street;
        this.phone = phone;
        this.fax = fax;
    }

    // default constructor is needed for
    // eXtremeDB
    public Address() {}
}

@Persistent(list=true)
class Employee
{
    @Indexable(unique=true)
    public String name;
    public int age;
    @Indexable(unique=false)
    public long salary;
    @Indexable(unique=false)
    public long companyID;
    @Optional
    Address address;
    @Blob
    public byte[] photo;
}
    
```

With eXtremeDB JNI, Java class definitions use "@" to indicate a database attribute.

Figure 2. eXtremeDB's Java API uses Java reflection to discover classes, their fields and other attributes at run-time.

Which DBMS in Java?

What are the database options in Java when an application calls for the highest possible responsiveness?

SQL Relational DBMSs (usually w/ JDBC driver) -- The biggest drawback is the "impedance mismatch" that occurs when going between relational structures and SQL (a set-oriented database access language), on one end, and the object-oriented Java language, on the other. The need to "map" between object and relational technologies consumes CPU cycles and hurts performance.

All-Java Databases -- Object databases written in Java improve performance by eliminating impedance mismatch. However, as an interpreted language, Java's performance can't compete with compiled C and C++, and this limitation binds all-Java databases. All-Java relational databases also suffer from that language's higher latency, and from impedance mismatch – a double blow to performance.

eXtremeDB JNI -- eXtremeDB's Java Native Interface uses Java (class definitions) to define the database. Since it relies on the programming language's own syntax, there is (by definition) no impedance mismatch. Database sorting, retrieval and storage take place in fast, compiled C code.

Embedded Systems & Real-Time Enterprise

eXtremeDB JNI offers a proven, off-the-shelf database for Java-based set-top boxes and other embedded systems. McObject's database provides the minimal latency demanded in the embedded market, with a tiny RAM and CPU "footprint" that lowers manufacturing costs.

eXtremeDB JNI also targets real-time enterprise systems such as highly scalable finance, e-commerce, social network and other Web-based applications. Developers deploy the 64-bit edition of *eXtremeDB* (*eXtremeDB*-64) as an in-memory front-end for an enterprise relational database management system (RDBMS). This dramatically accelerates data management, while providing benefits missing in traditional object caching and NoSQL solutions, such as efficient storage and safeguards on data integrity.

The JNI enhances *eXtremeDB*'s value as a cache by leveraging Java's familiarity among Web and enterprise developers. Compared to object caching solutions, it offers persistence: in the event of software or hardware failure, the cached data can be easily recovered, through features such as transaction logging, database replication (*eXtremeDB* High Availability), and the optional disk- or flash memory-based storage provided by McObject's *eXtremeDB* Fusion hybrid database technology.

McObject's *eXtremeDB*

eXtremeDB JNI offers the benefits of the *eXtremeDB* real-time database, which is already used in millions of devices and by millions of visitors to social network sites.

- Speed: core in-memory database system (IMDS) architecture delivers micro-second transactions even on modest hardware
- Optional SQL, XML and type-safe C/C++ interfaces
- Multi-version concurrency control (MVCC) transaction manager and advanced memory management fully leverage multi-threaded, multi-core systems
- Transactions support the ACID (Atomic, Consistent, Isolated and Durable) properties, to safeguard data integrity
- Available source code, for porting to new platforms and highest degree of control over development
- High Availability edition, with asynchronous (1-safe) or synchronous (2-safe) replication, for complete fault tolerance
- Transaction Logging edition supports database recovery, with features that are parameterized for developer flexibility
- 64-bit edition supports very large databases, with linear performance gains proven in benchmark on 160 cores with 15.54 billion row, 1.17 TB in-memory database
- Fusion edition supports in-memory, on-disk and combined storage. Optimize application for performance, persistence, cost and form factor

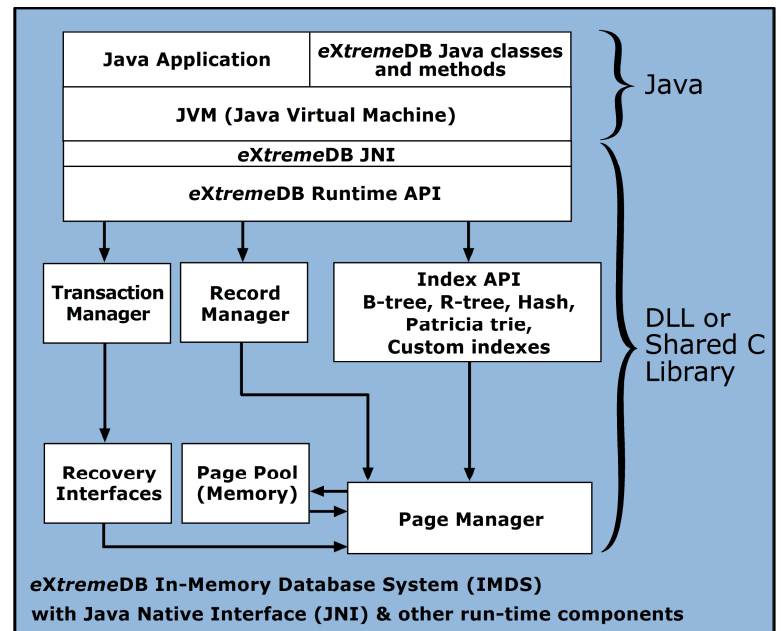


Figure 3. *eXtremeDB* JNI delivers the fastest possible database solution in Java.

Highly efficient indexing

eXtremeDB provides diverse indexes, optimized for a variety of application and data types. Supported indexes include:

- Hash indexes for exact match searches
- Tree indexes for pattern match, range retrieval and sorting
- R-tree indexes for geospatial searches
- Patricia trie indexes for network, telecom
- Object-identifier references for direct access
- Custom indexes

Java Annotations

eXtremeDB JNI uses Java annotations to define database attributes in application code. Recognized annotations include:

- @Blob - Uses *eXtremeDB* BLOB type to store a byte array
- @Dimension - Specifies dimension for fixed size arrays (char, scalar)
- @Encoding - Specifies encoding of string fields
- @Indexable - Includes a given field in an index
- @Indexes - Defines a compound (multi-field) index
- @Optional - Optional structure field, for example:
@Optional
Address address;
- @Persistent - Marks class as stored in the database and provides attributes of the class