*eXtreme*DB® In-Memory Database System

A new approach to data management for embedded devices and real-time enterprise applications. Unparalleled performance, with tools for easier integration.

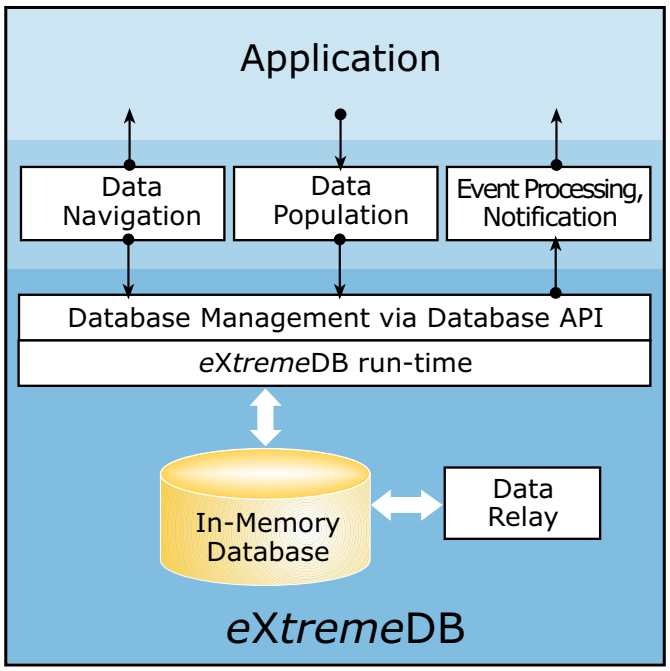
*eXtreme*DB, the embedded database system for software that is *eXtremely* innovative.

Real-time application design is increasingly complex. Performance and reliability requirements on shorter product cycles are driving the incorporation of proven, off-the-shelf software components. McObject's *eXtreme*DB® In-Memory Database System (IMDS) and related product family combine unmatched performance, reliability and developer efficiency in an industrial-strength package.

Today's intelligent devices – such as telecom and networking gear, consumer electronics, and industrial controllers – process growing volumes of complex data. Managing this data requires instant responsiveness with minimal RAM and CPU demands.

Real-time enterprise systems in fields such as capital markets, e-commerce and social networks need maximum speed and scalability, with features including specialized database indexes and multi-core optimization.

Recognizing the limits of traditional database systems, McObject designed *eXtreme*DB as a small-footprint IMDS that meets demanding requirements for performance, reliability and ease of implementation. *eXtreme*DB offers scalability into the terabyte-plus range, with an intuitive native C/C++ API as well as SQL ODBC, JDBC and native Java and C# interfaces. *eXtreme*DB Fusion supports hybrid in-memory and on-disk storage, while the *eXtreme*DB Cluster edition distributes databases across multiple hardware nodes, enabling applications to scale further, respond faster, and expand more economically.



System Architecture

"*eXtreme*DB helped cut 18 programmer months from the development cycle."

- The Boeing Company

The Need for a Real-Time Data Management Solution

On-device data management needs have increased exponentially. Organizations are also recognizing the competitive advantage of high performance in time-sensitive, data-intensive tasks, and are augmenting their IT infrastructure with new real-time systems ranging from securities trading to in-memory data caching.

Selecting a proven commercial database system for real-time enterprise and embedded systems must factor in price and the vendor's track record. High run-time performance is a must. The system must provide a developer-friendly programming interface, resulting in a shorter development cycle and more legible/maintainable code.

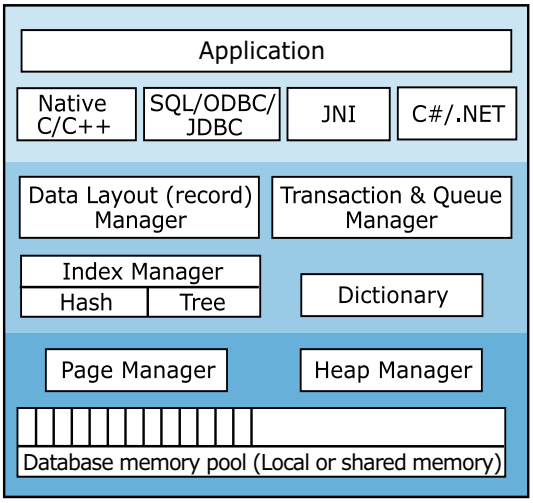
For the runtime environment:
- High throughput and responsiveness via core in-memory architecture
- Small footprint & compact data layout
- Transactions w/ ACID properties
- Direct data access
- Support for multiple data and index types
- Compatible with leading OSs and RTOSs
- Scales up via 64-bit support, clustering & multi-version concurrency control (MVCC)
- Multi-core optimization
- Hybrid (in-memory & persistent) data storage option

For development:
- Source code available
- Intuitive and type-safe native C/C++ API
- Built-in consistency and error checking
- Easy-to-learn standard functions
- Flexible and efficient data queries
- Generates maintainable code
- High performance SQL/ODBC/JDBC; native Java & C#/.NET APIs

Only *eXtreme*DB from McObject addresses all these needs, combining world-class execution with developer-friendly design. Other database systems, based on a "one size fits all" model, force developers to write code that is only understandable to those familiar with a rigid, proprietary API, and impose data storage that is entirely "on-disk" or "in-memory" when a blended approach would be ideal. With the *eXtreme*DB product family, real-time database management reaches new levels of performance, reliability and maintainability.

The Runtime Environment

*eXtreme*DB was designed for performance, starting with a memory-based architecture and direct data manipulation. Typical read and write accesses require a few microseconds, or less. The engine is re-entrant, allowing for multiple execution threads. Transactions support the ACID properties, assuring transaction and database integrity.



*eXtreme*DB Fusion builds on the core architecture by offering on-disk as well as in-memory storage, for flexibility in tuning application performance and data persistence.

*eXtreme*DB's product family offers proven tools to obtain speed and scalability at runtime. Transaction logging is supported, and a 64-bit edition delivers terabyte-plus IMDS scalability. A High Availability edition provides fault-tolerance while *eXtreme*DB Cluster can dramatically increase available net processing power while reducing system expansion costs through the use of low cost (i.e. "commodity") hardware.

***eXtreme*DB**	**Advantage**
Small footprint – 150K or less depending on processor and compiler	Fits most resource-constrained environments
In-memory database – all data is stored in main memory	No disk & file system overhead
Direct data access – application works with data in main memory	Minimizes CPU cycles and latency
No translation – *eXtreme*DB stores data in the exact form used by the application	Eliminates overhead from translating to relational representations
Data integrity – transactions support the ACID properties	Reliable implementations
Re-entrant engine	High performance even when using multiple execution threads
64-bit support, clustering and multi-version concurrency control (MVCC)	Scalability, optimized for multi-threaded applications on multi-core
Cyclic redundancy check (CRC) and RC4 encryption	Database security
Binary schema evolution	Enables fast, efficient changes to database design

The Development Environment

Developers strive to produce readable, maintainable, efficient code in the shortest time. *eXtreme*DB includes numerous features that boost development efficiency.

*eXtreme*DB offers a choice of database interfaces. Its efficient native C/C++ API is type-safe: the compiler can catch data typing and assignment errors, resulting in more reliable run-time code. *eXtreme*SQL provides a high-performance, ODBC- and JDBC-compliant implementation of the SQL interface. Also available: a C# (.NET) Native Interface and a Java Native Interface that enable programmers to define and call *eXtreme*DB databases entirely from within these environments, and leverage the speed of a DBMS run-time that executes in compiled C.

Source code is available for porting and for an in-depth understanding of *eXtreme*DB internals. Customizable collations provide utmost flexibility in specifying character sorting sequences. *eXtreme*DB supports virtually all data types as well as multiple indexes, including hash indexes for exact match searches; a classic B-tree implementation; Patricia tries for network/telecom applications; R-tree indexes for geospatial lookups; KD-trees for multi-dimensional and Query-by-Example (QBE) searches; and object-identifier references, for direct access. For in-memory databases, rather than storing duplicate data, indexes contain only a reference to data, keeping memory requirements to an absolute minimum.

Intuitive Native API

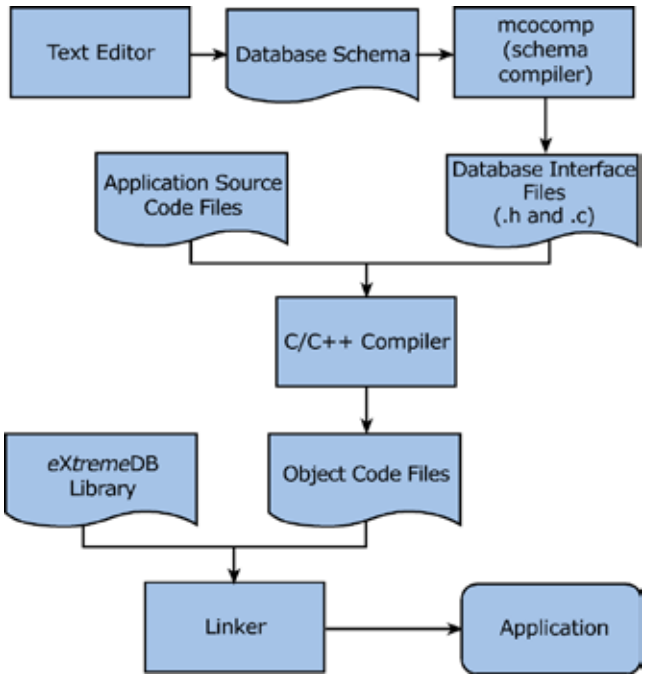
With *eXtreme*DB's native C/C++ API, the developer focuses on the data definition first, then *eXtreme*DB generates the API from this definition via the schema compiler. The result is

- An easy-to-learn API that is optimized for the application
- Code that is more legible as well as easier to write and maintain
- Compile-time type-checking that eliminates coding errors that cause database corruption

Example:

The following is a (simple) class and an example of putting a new value into a record in the database:

```
class Measurement{
      string measure;
      time timestamp;
      unique tree <measure, timestamp>
trend;
};
Measurement_measure_put(&m, meas);
Measurement_timestamp_put(&m, value);
```



"*eXtreme*DB simplifies development and testing, especially in situations where the database must coordinate multiple processes."

- Tyco Thermal Controls

Progressive error detection and consistency checking features

In debug mode, if an application passes an invalid transaction or object handle into a runtime method, *eXtreme*DB raises a fatal exception and stops execution. The developer can examine the call stack for the source of the coding error. In release mode, the transaction is put into an error state to prevent database corruption. The *eXtreme*DB runtime implements many verification traps and consistency checks, which can be removed after debugging to restore valuable clock cycles.

*eXtreme*DB Product Family

- ***eXtreme*DB In-Memory Database System (IMDS)**
  Core *eXtreme*DB edition offers a high-level data definition language, concurrent access, transactions, event notifications, flexible indexing and more.
- ***eXtreme*DB Fusion**
  Combines on-disk and in-memory data storage. Optimize for storage speed, persistence, cost and form factor.
- ***eXtreme*DB Cluster**
  Distributed solution maximizes available net processing power, reliability and scalability, while lowering system expansion costs.
- ***eXtreme*DB High Availability**
  *eXtreme*DB-HA enables multiple fully synchronized database instances, with automatic failover.
- ***eXtreme*DB Transaction Logging**
  Provides in-memory database durability and recoverability via logging. Unique Data Relay feature replicates selected changes to external systems.
- ***eXtreme*DB-64**
  64-bit *eXtreme*DB supports in-memory databases that are hundreds of times larger than 32-bit.
- **Other editions**
  ***eXtreme*SQL** – *eXtreme*DB with SQL, ODBC and JDBC APIs. ***eXtreme*DB Kernel Mode** – runs in OS kernel, eliminating context switches when kernel tasks access database.

Complex data types and efficient queries

- Supports virtually all data types, including structures, arrays, vectors and BLOBs
- Querying methods include hash indexes for exact match searches
- Tree indexes support queries for pattern match, range retrieval and sorting
- "Voluntary" indexes for program control over index population
- R-Tree, KD-Tree and Patricia trie indexes
- Object-identifier references provide direct data access
- Autoid (auto-increment) for system-defined object identifiers
- Rather than store duplicate data, in-memory indexes contain only a reference to data, minimizing RAM demands

"*eXtreme*DB was by far the fastest database we could find. It was more than twice as fast as the second place database."

- Spirent Communications

Target platforms

- Linux, VxWorks, INTEGRITY
- Nucleus, LynxOS, eCos, uCLinux, uC/OS-II
- Windows Embedded platforms
- QNX Neutrino, Quadros RTXC
- Solaris, HP-UX, Windows
- *eXtreme*DB source code for all platforms

32-bit Database specifications

- Maximum objects per database: 2,147,483,647
- Maximum classes per database: 32,767
- Maximum indexes per database: 32,767
- Maximum fields or vectors per class: 32,767
- Maximum fields per index: 32,767
- Maximum elements per vector: 32,767
- Memory requirements: 150K or less
- Databases open simultaneously: 16 (configurable)
- Simultaneous connections per database: 64 (configurable)

Supported data types

- 1, 2, 4, 8-byte signed/unsigned integers
- enum
- float, double, numeric
- date, time
- char (fixed length), string (variable length)
- fixed-size array
- variable-length vector
- structs (nested to any depth)

mcobject® precision data management

McObject LLC
33309 1st Way South, Suite A-208
Federal Way, WA 98003
Phone: +1-425-888-8505
Fax: +1-253-878-5481
Web: www.mcobject.com
E-mail: info@mcobject.com