System development for IoT Edge devices and gateways is increasingly complex. The ability to respond to events immediately is paramount. The security of IoT systems is under ever-increasing scrutiny, and shorter product development cycles drive the need for proven off-the-shelf technology stack components. The eXtremeDB database system and related product family combine unmatched performance, security, reliability and developer efficiency.

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

Some domains require specialized database indexes such as for geolocation, telephone numbers and IP addresses, and fuzzy search.

Recognizing the limits of traditional database systems, McObject designed eXtremeDB as a small-footprint DBMS that meets demanding requirements for performance, reliability and ease of implementation. eXtremeDB offers scalability, with an intuitive native C/C++ API as well as SQL, ODBC, JDBC and native Python, Java and C# interfaces. eXtremeDB supports hybrid in-memory and on-disk storage, while eXtremeDB Cluster, High Availability, and sharding capabilities distribute databases across multiple hardware nodes, enabling applications to scale further, respond faster, and expand more economically.

“eXtremeDB helped cut 18 programmer months from the development cycle.”

- The Boeing Company
The Need for a High Performance, Low Overhead 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 systems ranging from time series data management for IoT systems to in-memory data caching.

Selecting a proven commercial database system for IoT systems must factor in price and the vendor’s track record. High run-time performance is a must. The system must provide developer-friendly programming interfaces for common programming languages, 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 with ACID properties
- Direct data access
- Support for multiple data and index types
- Compatible with leading OSs and RTOs
- Scales up via 64-bit support, sharding, clustering & multi-version concurrency control (MVCC)
- Multi-core optimization
- Hybrid (in-memory and/or persistent) data storage

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 queries
- Generates maintainable code
- High performance SQL/ODBC/JDBC; Python, Java & C#/.NET APIs
- LUA stored procedure language
- The best of NoSQL and NewSQL

Only eXtremeDB 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 eXtremeDB product family, embedded database management reaches new levels of performance, reliability and maintainability.

The Runtime Environment

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

eXtremeDB builds on the core architecture by offering persistent as well as in-memory storage, for flexibility in tuning application performance and data persistence.

eXtremeDB’s product family offers proven tools to obtain speed and scalability at runtime. Transaction logging is supported, and a High Availability edition provides fault-tolerance while eXtremeDB Cluster and sharding can dramatically increase available net processing power while reducing system expansion costs through the use of low cost (i.e., “commodity”) hardware.

<table>
<thead>
<tr>
<th>eXtremeDB Key Feature</th>
<th>Advantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small footprint ~ 150K or less depending on processor and compiler</td>
<td>Fits most resource-constrained environments</td>
</tr>
<tr>
<td>Direct data access – application works with data in main memory</td>
<td>Minimizes CPU cycles and latency</td>
</tr>
<tr>
<td>eXtremeDB stores data in main memory</td>
<td>Preserve CPU cycles and minimize memory overhead by eliminating data copies</td>
</tr>
<tr>
<td>Data integrity – transactions support the ACID properties</td>
<td>Reliable implementations</td>
</tr>
<tr>
<td>Re-entrant engine</td>
<td>High performance when using multiple execution threads</td>
</tr>
<tr>
<td>64-bit support, clustering and multi-version concurrency control (MVCC)</td>
<td>Scalability, optimized for multi-threaded applications on multi-core</td>
</tr>
<tr>
<td>Cyclic redundancy check (CRC) and AES encryption</td>
<td>Database security</td>
</tr>
<tr>
<td>Dynamic DDL</td>
<td>Easy schema migration</td>
</tr>
</tbody>
</table>

Application

<table>
<thead>
<tr>
<th>Native C/C++</th>
<th>SQL/ODBC/JDBC</th>
<th>Java</th>
<th>C#</th>
<th>Python</th>
</tr>
</thead>
</table>

Data Layout (record) Manager | Transaction & Queue Manager

Index Manager | Dictionary

Page Manager | Heap Manager

Database Memory Pool (Local or Shared Memory)
The Development Environment

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

eXtremeDB 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. eXtremeSQL provides a high-performance, ODBC- and JDBC-compliant implementation of the SQL interface. Also available: a C# (.NET) Native Interface, a Java Native Interface and a Python API that enable programmers to define and call eXtremeDB 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 eXtremeDB internals. Customizable collations provide utmost flexibility in specifying character sorting sequences. eXtremeDB supports virtually all data types, including time series, as well as multiple index types, 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; trigram indexes for “fuzzy search”; 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 eXtremeDB’s native C/C++ API, the developer focuses on the data definition first, then eXtremeDB 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:

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

“eXtremeDB simplifies development and testing, especially in situations where the database must coordinate multiple processes.”

- Pentair

Progressive error detection and consistency checking features

In debug mode, if an application passes an invalid transaction or object handle into a runtime method, eXtremeDB 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 eXtremeDB runtime implements many verification traps and consistency checks, which can be removed after debugging to restore valuable clock cycles.
eXtremeDB Product Family

• **eXtremeDB Edge**
  Core eXtremeDB edition offers a high-level data definition language, concurrent access, transactions, event notifications, flexible indexing, and combines on-disk and in-memory data storage enabling optimization for storage speed, persistence, cost and form factor.

• **eXtremeDB HPC**
  Built to handle the scalability and analytics needs of IoT Big Data systems. “In-chip” pipelined vector-based analytics, time series support, LUA stored procedure language, distributed query engine enables parallel query execution across shards for linear scalability.

• **eXtremeDB High Availability (HA) and Cluster**
  Cluster solution maximizes available net processing power, reliability and scalability, while lowering system expansion costs. HA enables multiple fully synchronized database instances, with automatic failover.

• **eXtremeDB Transaction Logging**
  Provides in-memory database durability and recoverability via logging. Data Relay, an open API to replicate eXtremeDB content to non-eXtremeDB systems. Persistent Queue of Events for sophisticated event handling.

• **eXtremeSQL**
  ODBC and JDBC APIs. Facilitates sharding and distributed query processing for maximum horizontal scalability.

• **eXtremeDB/rt**
  eXtremeDB/rt is the first and only deterministic embedded database management system for mission- and safety-critical hard real-time applications, preserving the external and internal consistency of data by enforcing transaction deadlines and the ACID properties.

**Supported data types**

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

**Complex data types and efficient queries**

- Supports virtually all data types including time-series 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, trigram 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

**Database specifications**

- Maximum objects per database: quintillions
- Maximum tables per database: 32,767
- Maximum indexes per database: 32,767
- Maximum columns or vectors per class: 32,767
- Maximum columns 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)

**Target platforms**

- Linux, VxWorks, INTEGRITY
- Nucleus, LynxOS, eCos, uCLinux, uC/OS-II
- Windows Embedded platforms
- QNX Neutrino, ThreadX
- Solaris, HP-UX, Windows
- eXtremeDB source code for all platforms