

# 組込み機器のための 自作のデータ管理手法に代わるデータ管理手法

**背景：**市場で生き残るためには、情報機器、例えばセットトップボックスやネットワーク製品等は、急速にその機能を拡張する必要があります。具体的には、ソフトウェア的な「知性」を付け加えて、かつてない複雑なデータを大量に管理することを意味します。このような要求には通常はデータベース管理システム(DBMS)で対応します。しかしながら、これまでのデータベースは、ビジネスの処理をそのルーツとしており、価格に敏感なハイテク製品にとって、高価なCPUと大量なメモリを要求します。

このような状況により、ビジネス指向の不必要な処理層を省きつつ、可能な限りタイトなインテグレーションを実現するツールを提供する新しい種類のデータ管理ツールが求められています。このデータシートでは、新しい、デバイス内蔵型のデータベースに必要な機能について検証し、これらのニーズを満たすデータベース製品「eXtremeDB」に焦点を当てていきます。

**ガイロジック株式会社**

<http://www.gailogic.co.jp/db>

**db@gailogic.co.jp**

aCopyright 2001, McObject LLC

## 概論

セットトップボックスからスロットマシンに至る各種デバイスは、革新的な新機能を実現するために、信頼できるデータ管理を必要としています。既存のデータベース管理システムでは、そのルーツをビジネスコンピューティングに端を発していることもあって、この多岐にわたるアプリケーションに対応するには、多くの場合、十分なソリューションは提供できておりません。技術的な理由はもとより経済的な理由によっても、最新の組込み機器向けの理想的なデータ管理ツールには、メモリサイズとCPUフットプリントが小さく、リアルタイムパフォーマンスに優れ、複雑なデータストリームも効果的に処理し、デバイス固有のアプリケーションと強固に統合できることが求められます。

本ホワイトペーパーでは、McObject社の新しいデータベース製品eXtremeDBについて検証を進めていきます。eXtremeDBは、メインメモリ上でデータ管理を行う心臓部と、デバイス内蔵型データベースに何が必要か（同様に、何が余分か）を考慮した上で設計されたアーキテクチャによって、上記の革新的な諸機能の多くを実現したデータベース製品です。

## デバイスデータ管理の新分野

ネットワーク接続可能な情報機器の数は、PCの販売が頭打ちになっても依然急激な伸びを示しています。オフィス、自動車、テレビ、ポケットや財布の中、産業・通信・交通システムの内部等、このような場面で使われるデバイスは、おなじみのPCのようなインターフェースを持ち合わせてはいませんが、その代わりに、サイズ、携帯性、使い方の容易さにおいて優位点があります。かつては最小限のインテリジェンスのみ搭載した組み込みシステムとみなされていた製品は、強力なCPUと高度なソフトウェアを実装するように進化していきました。

肥大化する諸機能に対応するために、一般的にアプリケーションは次第に大きく複雑なデータを扱う必要が出てきました。コンピュータシステムにおけるこの真実は、デバイスにも当てはまり、その結果、デバイスの開発者は、自作のデータ管理ツールより信頼性が高いと考えられる市販のデータベースへとシフトし始めました。ある老舗のデータベースソフト会社の調査では、以下のようなアプリケーションでデータベースの需要が出てきた模様です。

- 常に更新される番組表に対応するためにデータ管理を必要としているインターネット対応の新しいセットトップボックス
- 書類をスキャンしてデータベース化してある何千ものアドレスやメールリストにメールすることができる次世代の事務機器
- 支払テーブルのデータ管理やリアルタイムゲーム管理が必要なスロットマシン

- 通話のルーティングのためにリアルタイムのユーザーデータや機器データを必要とする長距離通信会社のテレコムスイッチ
- ユーザーデータやコンフィギュレーションデータが必要なLANベースのオフィスネットワーク
- エンジントラブルの原因をピンポイントで発見し、解決方法を提供するための情報を保存する機能を持った自動車診断設備

### どのデータ管理技術が適当か？

デバイス開発者の方々の市販データ管理ツールに対する関心は高まるばかりですが、既存のデータベースソフトウェアは期待に応える理想的な解決ではありませんでした。現在最も多用されているタイプであるリレーショナルデータベースは、10年以上前に登場して企業向けのビジネス機能をサポートしています。これらのデータベースにはSQL（高レベルインターフェース）やロック調停、キャッシュ制御、異常終了復旧機能等、ビジネス指向の機能のサポートが含まれます。しかし、セットトップボックスや次世代FAXのようなデバイスでは、多くの場合、これらの機能は不必要であり、メモリやCPUおよびストレージ装置等、リソースを過分に使用します。

オブジェクト指向のデータベース管理システムもまた、ソリューションとして紹介されてきましたが、これらは、リレーショナルデータベース同様、ビジネスの目的を達成するために開発されたものであり、情報機器のリアルタイムニーズに応えるには遅すぎます。多くの場合、リレーショナルデータベースやオブジェクト指向データベースの開発元では、機能を削ることで、より軽い製品を投入してきました。しかし、このような機能縮小型データベースには「もし最初から組込み機器向けに設計されたデータベースであればもっと違うものになるのではないか？」という疑問が必然的に生じます。

### デバイスはそれぞれ違う

デバイスは、その固有のニーズによって新しい開発分野を代表する。この新たなアプリケーションにとって理想的なデータ管理を構成するのはどのような要素でしょうか？

### 小さなフットプリント

組込み機器の製造メーカーは、経済的理由から、メモリ、CPU、記録メディアのリソースを削減しようと最大限の努力を惜しみません。大半のデバイスは、しばしば極めて競争の激しい市場で大量出荷をすることを前提に作られます。単価をわずかに下げただけで、マーケットシェアを拡大する場合もあり、製造コストの低下に伴い、価格はコスト割れぎりぎりのところまで下がります。RAMサイズをキロバイト単位で削減したり、少しでも低価格のCPUを採用することで、製品の成否が決まる場合があります。

## リアルタイムパフォーマンス

「待ち時間」というものは、デスクトップ環境やクライアント・サーバ環境では当たり前の概念ですが、コンシューマエレクトロニクスや情報家電等のデバイスにおいては異なっており、迅速な反応が要求されます。このハードリアルタイム性のあるシステムは、予め設定した時間内に反応することを要求されます。デバイスのデータ管理は、本質的に効率の良いアーキテクチャをベースとしていなければならない、パフォーマンスのオーバーヘッドの源は最小限に抑えられているか、排除されていることが必要です。

## 複雑なデータの流れ

ビジネスアプリケーションとは異なり、デバイスが整理されたテーブル構造のデータを管理することはあまりありません。組込み機器は、しばしばツリーや任意の長い配列など、複雑なデータを管理する必要があります。多くの場合、ビジネスデータベースが大変に大きなサイズで複雑なトランザクションに適しているのに対し、組込み機器では、小さく速いトランザクションを行います。例えば、テレコムスイッチでは、利用可能なハードウェアチャンネルに従って通話を接続するし、ゲーム機では「現在進行中」のゲームの中で次々に起こる展開に反応していくことが必要になります。

ビジネスデータベースは、SQLその他の高レベルなツールを使って、非プログラマーでもアクセス可能なように設計されています。対照的に、組込み機器の場合は、限定された使用目的に合わせて設計されているので、あらかじめ開発者によってデータアクセス手段が定義されているのが一般的です。さらに、データ管理用コードとアプリケーションの緊密な統合によって、プログラムの動作は非常に効率が高くなり、CPUその他のリソースも節約可能になります。これらの要因により、開発ツールの必要性がさらに高まります。組込み機器で利用されるデータベースには、エンドユーザ向けにクエリー言語を用意することよりも、開発者向けにC、C++もしくはJavaのような組込み機器開発プロジェクトで通常使用されている言語を用いた強力な開発環境を提供することが重要です。

## eXtremeDBの紹介

McObject社により近日アナウンスされたeXtremeDBは、情報・ネットワーク機器のためにスクラッチから（100%自社において）開発されたデータベースプログラムです。ソフトウェア/データベース/組込み機器開発の経験者によって設立されたMcObject社は、現実の製品開発現場からのフィードバックをその持てる技術に反映させている企業です。その結果生まれたのが、eXtremeDBです。eXtremeDBは、フットプリントが小さな、メインメモリ上で動作するデータベースで、複雑なデータの高速処理に最適です。そのために、強力な開発ツールと、斬新なアーキテクチャを採用し、既存のデータベースソフトウェアが持つパフォーマンスオーバーヘッドを回避することに成功しています。

## eXtremeDBのアーキテクチャ

eXtremeDBのデザインは、組込みシステム固有のパフォーマンスニーズ(同時に、既存のビジネス用データベースが持つ「リソース偏重」という特徴を適用することの不適切さ)を認識しています。一つの重要な事実は、組込みシステムは、本質的に「配布されない」ものである、ということです。データベースと一緒に動作するプロセスやタスクは、全て同一のCPUで動作します。このことは、必ずしもネットワークが存在しないことを意味しません。例えば、セットトップボックスは非常に大規模なネットワークの一部です。しかし、そのセットトップボックスの1タスク(データベースではない)が通信を担います。このことにより、不必要なリモート処理コールメカニズムとその他の複雑な通信ロジックがビジネスデータベースの中心部に作られます。

eXtremeDBはまた、情報機器内蔵のデータベースにとって「ユーザ」の定義が異なることを認識しています。eXtremeDBにとっての「ユーザ」は、「プロセス」または「スレッド」のことを指し、一般的に「プロセス」も「スレッド」もあまり多くありません。そのようなデータベースには、複雑なロック調停、並列処理制御、キャッシュデータの同期、異常終了からの復帰といった機能は必要とされません。この条件に基づいて設計されたeXtremeDBは、既存のデータベースと比較して、より小さく、より速く、さらに、複雑さを排したことにより、よりエラーに強いデータベースとなっています。

確実なデータ統合のために、eXtremeDBはACIDに完全準拠したデータ処理を行います。しかし、データベースの処理キューは、最小のリソースで行えるように設計されています。それはまた、動的なキューを提供し、アプリケーションが異なる種類のトランザクションに優先順位を割り当てることを可能にします。

このスリムなアーキテクチャが、以下に述べる追加機能とともにeXtremeDBのフットプリントを最小にしています。eXtremeDBのフットプリントは、プロセッサやコンパイラにもよりますが、100KBもしくはそれ以下です。スリムなアーキテクチャはまた、民生品でコスト削減のために頻繁に利用される比較的ローパワーなプロセッサでも最適なパフォーマンスを得ることを可能にします。

### メインメモリ、直接アクセスデータベース

eXtremeDBはデータを丸ごとメインメモリに保持することで、ディスクアクセスのためのオーバーヘッドを排除しました。メインメモリ上での処理は、情報機器向けに設計されたトランザクションマネージャと組み合わせて行うことにより、一連の小規模高速データ処理への対応を加速します。メインメモリ上での処理により、データ転送に伴うオーバーヘッドもまた削減されます。これまでのデータ管理技術では、データベースからキャッシュへデータを一度コピーしてから、アプリケーションがそのデータを利用できるように別の場所へ再配置することが必要でした。eXtremeDBを使えば、アプリケーションはデータを直接メインメモリ上で使用するため、データをコピーする必要も、常駐領域から一時領域に移動させる必要も無くなります(図1参照)。

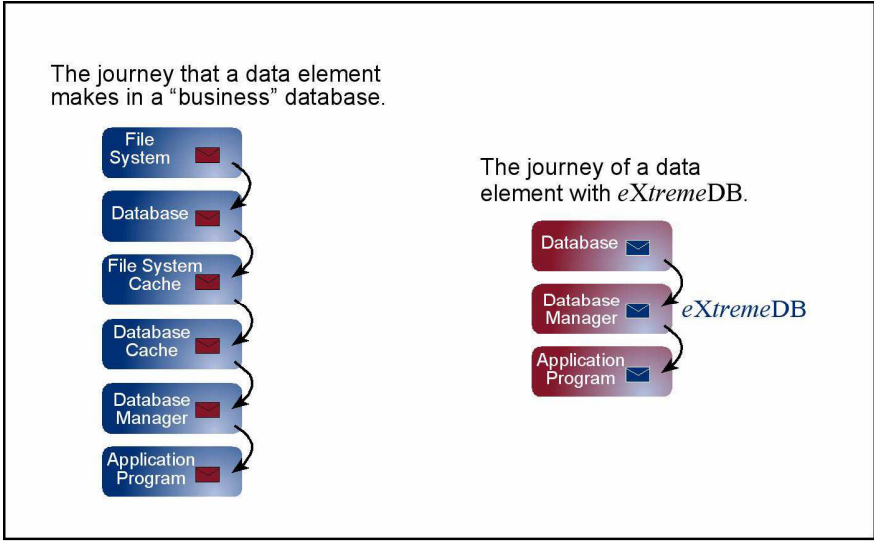


Figure 1

SQLデータベースと異なり、eXtremeDBは情報をアプリケーションが使用するまさにその形態でデータを保持します。他のデータベース技術では、データ形式の「翻訳」が必要になります。例えば、Cデータ要素を関連付けされた索引にマッピングしたり、データテーブルからフィールドを取り出すために追加コードを要求し、それをC構造体にコピーしたりといったことが必要になります。このオーバーヘッドを排除させることにより、eXtremeDBは、メモリやCPUのリソースを削減します。図2は、複雑なオブジェクトを処理する時にリレーショナルデータベースがデータを「解体」して「組み立て直す」ことにより発生するオーバーヘッドを図示したものです。

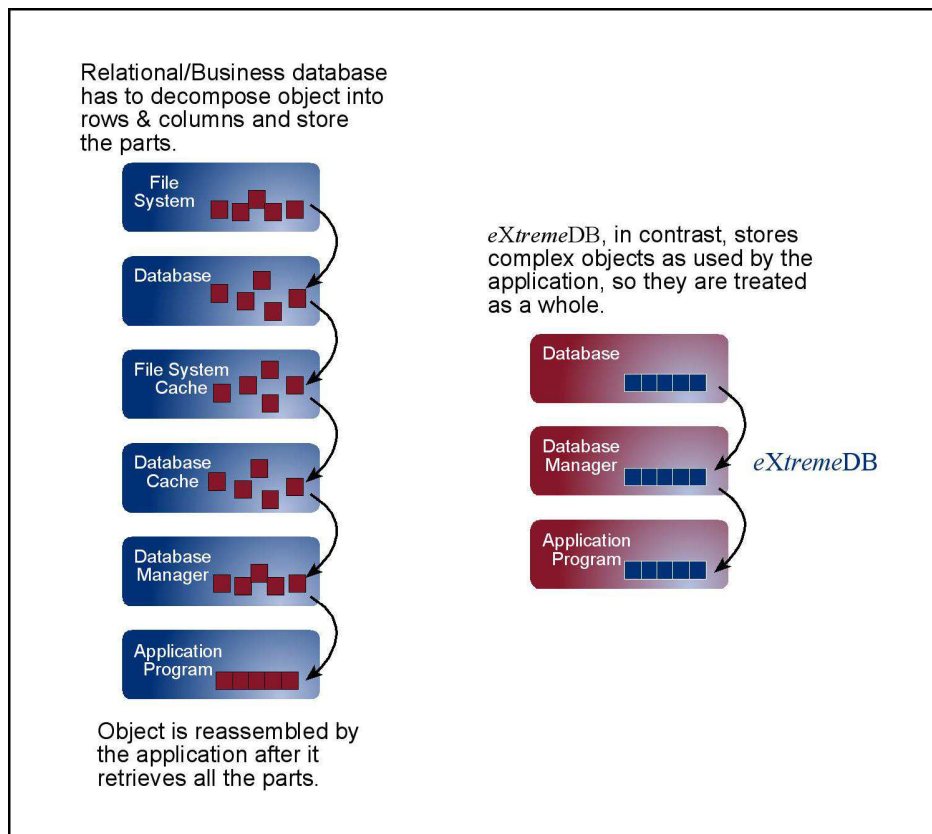


Figure 2

## 開発者への指針

McObjectは、情報機器のために開発されたアプリケーションは、後からデータベースに「挿入する」ものではないことを理解しています。最善の動作効率のためには、むしろデータベースはアプリケーションソフトウェアと緊密に統合化されます。今日、組込みシステムの開発においては、広範にC/C++が使用されています。eXtremeDBは、望ましい開発環境を拡張し、オブジェクト指向の分析と設計に対してデータを中心とした手法を適用します。重要な開発ツールには、複雑なデータタイプと検索手法、豊富で直感的なアプリケーション開発インターフェース、そして強力なデバッグ環境への対応など、eXtremeDBの諸機能が含まれます。

### 複雑なデータと効率的なクエリをサポート

リレーショナルデータベースは、基本となるデータタイプの中の狭い隙間にデータを押し込むことを要求します。それに対してeXtremeDBでは、構造体、ベクタ、BLOBを含む実質上全てのデータタイプをサポートすることによって、緊密かつ効率的なコーディングを促進します。検索のためには、McObject社は、完全一致検索のためのハッシュインデックス、パターン・マッチや部分検索やソートのためのbツリーインデックス、オブジェクト識別子の参照機能を、データへの直接アクセスのために提供しています。コピーデータを保存するのではなく、インデックスにデータへの参照のみが記されます。これにより、メモリの必要量を最小限に抑えることができます。(詳細は、「付録データタイプと保存属性；eXtremeDB対SQLデータベース」をご参照下さい。)

### プログラムインターフェース

McObjectは、eXtremeDBをベースとしたアプリケーションで使われる標準的なデータベース機能のライブラリを提供します。しかし、アプリケーションの中にある常駐型のデータへのアクセスを行うためのAPIのほとんどはデータベースがコンパイルされた時にeXtremeDBによって生成されます。このアプリケーション固有のAPIは開発者の独自データデザインに基づいているため、容易に理解でき、開発する製品のニーズに100%合致したものとなっております。

### デバッグ環境

McObject SDK (開発ライセンス) ランタイムは、データベースのコードが持つ数多くのトラップを利用し、プログラムエラーを検出し、簡単なデータ修復を可能にします。さらに、C環境でのコンパイル時のタイプチェックはMcObjectが保存されているデータにアクセスするために使用する手法に適用されます。特定のオブジェクトへのアクセスを行うために作成された手法は、そのデータタイプをパラメータとして参照します。どのようなエラーが起きても、コンパイラの警告(Warning)が発せられます。

## 結論

組込み機器の開発者には、背反する2つの使命があります。新たな機能の提供(「驚異的」なものが好まれます。)を継続し、価格面でも競合に一步先んじることです。過去5年間、新しい機能が要求されるソフトウェアは、関門を通過してきました。データは十分に複雑になり、開発者はデータベースに目を向けるようになりました。たとえそれが、SQLデータベースを不釣合いな場所に押し込む結果となることを意味してもです。しかし、リレーショナルデータベースがクライアント・サーバ環境で必要であったように、新しいデータ管理技術が情報機器向けに生まれてきているのです。eXtremeDBという製品によって具体化されたように、新たなソフトウェアは開発の柔軟性、パフォーマンス、リソースの節約を、デバイスのデータ管理の3本柱として、その重要性を強調しています。

## 付録ーデータタイプと保存属性

### eXtremeDB対SQLデータベース

以下のテーブルでは、eXtremeDBがサポートするデータタイプと保存属性を、SQLを使ったリレーショナルデータベースのものと比較したものです。より広範囲のデータタイプを（特に、より複雑なデータタイプを）サポートすることは、eXtremeDBの主な特長の一つで、このことはより効率的なコードを書く柔軟性および、デバイスに依存したアプリケーションに結合した複雑なデータを管理する能力において現れています。

**ベクタと構造体**はeXtremeDBに複雑なデータを管理する能力を与えるものです。これらがないと、データベースは単純な（原子的な）フィールド以外のなにものでもなくなってしまいます。リレーショナルデータベースであればテーブル全体を必要とし、さらにテーブルに伴う制御用オーバーヘッドまで必要とするような情報をベクタと構造体によって、単純なデータタイプで、呼び出すことができます。

**OIDとref**はeXtremeDBで開発する開発者が、オブジェクト識別子(OIDs)を持つクラス間の高速で、効果的で自然な関係を確立することを可能にします。

上位のデータタイプである**Date**、**Time**、**Timestamp**、**Decimal**は、必ずしもすべてのアプリケーションで必要とはなりません。フットプリントが重要な時は、上位のデータタイプの実装は、必要か否かにかかわらずすべてのアプリケーションに使用させるよりも、必要に応じてアプリケーション側で実装する方法が優れています。

eXtremeDBのデータ修正モジュールである**Compact**、**Optional**、**Voluntary**は、データベースによるメモリ消費を最小限にするために全て重要な役割を担っています。

**Voluntary**は、CPUサイクルの低減のためにも重要です。**Compact**というクラスを宣言することで、データベースの実行プログラムがクラスのオブジェクトを管理するために課してくるオーバーヘッドを削減することが可能です。**Compact**のオブジェクトは64KBを下回る事が知られ、内部オフセットに2バイトという短い整数と一緒に使用されます。それに比較して、**Compact**以外のオブジェクトでは、4バイトという大きな整数をオフセットに必要となります。

**Optional**フィールドは、データベースの空間を、アプリケーションが明確に使用していない限り消費しません。

**Voluntary**は、アプリケーションが要求した時のみインデックスを作成することを意味します。インデックスが作成されるまで、もしくはドロップされるまで、メモリ空間やインデックスを保持するためにinsert/update/delete動作中のCPUサイクルは必要にはなりません。インデックスの作成は、情報処理リソースの可用性と同期して選択的に行うことも可能です。

**unsigned**型を持つことにより、いくつかの安全策がとられています（重量等、論理的に負の値を受けつけないフィールドに不用意に負の値を書き込むことを避けることができます）。

| eXtremeDB | SQL             | 意味            |
|-----------|-----------------|---------------|
|           |                 |               |
| Char<n>   | Char(N)         | 固定長の文字列       |
| String    | Long<br>Varchar | 可変長の文字列 < 64K |
| Signed<1> | Tinyint         | 1バイトの符号付整数    |
| Signed<2> | Smallint        | 2バイトの符号付整数    |

|             |                   |  |
|-------------|-------------------|--|
| Signed<4>   | Integer           | 4バイトの符号付整数   |
| Unsigned<1> |                   | 1バイトの符号なし整数  |
| Unsigned<2> |                   | 2バイトの符号なし整数  |
| Unsigned<4> |                   | 4バイトの符号なし整数  |
| Float       | Real              | 4バイトの浮動小数  |
| Double      | Float             | 8バイトの浮動小数  |
| Blob        | Long<br>varbinary | 可変長バイナリデータ   |
| Vector      |                   | 様々なタイプの可変長アレイ  |
| OID         |                   | 固有のオブジェクト識別子   |
| Ref         |                   | 他のオブジェクトの識別子への参照   |
| Struct      |                   | 新しいタイプを名づけ、タイプ名を保持するフィールドを指定   |
| Optional    |                   | 保存されないフィールドを宣言   |
|             | Date              | YYYYMMDD形式の日付情報のための、ベンダー固有の保存フォーマット                                  |
|             | Time              | HHMMSS形式の時間情報のための、ベンダー固有の保存フォーマット                                    |
|             | Timestamp         | YYYYMMDD:HHMMSS.dddd date/time形式で日時を表すための、ベンダー独自の保存方法                |
|             | Decimal           | 数字による、ベンダー独自の保存フォーマット  |
| Compact     |                   | クラスが常に64K未満であることを宣言し、McObjectのオーバーヘッドの最小化を実現                         |
| Voluntary   |                   | 命令により作成・削除され、また、必要とされないか、あるいはない方が好ましい時にメモリ空間とCPUサイクルを節約するインデックスを宣言する |