C言語に変換できるDeviceSQLによるプログラミング ―― 意外なデータベースの利用方法

使って覚える組み込みデータベース入門（中編）

前編では，データベース・プログラミングの入門編として概要を解説しました．本来なら，今回は，より一歩踏み込んで，「こんなにデータ検索」が楽になる！というような内容を解説するべきなのでしょう．しかし，より多くの読者の方にデータベースへの興味を持ってもらうために，ちょっと意外なデータベースの利用法を紹介します．（筆者）

高橋 君夫

さて，前回（2007年10月号，pp.155-165）はごく簡単なプログラムを例に，データベース・プログラミングは難しくないことを説明しました．確かにデータベースのプログラミングは難しくありません．ただ，カー・オーディオや携帯電話の電話帳など，“いかにも検索します”といった用途ではデータベースの使用を考えるかもしれませんが，読者の皆さんはこのような製品を開発している方ばかりではないと思います．

1. 増加する多国語対応や機器構成パラメータ管理

日ごろ開発者の方々と話をすると，多くの人が，「昔は表示系も単純だった上に，一般的な言語でメニューが表示できれば，みんな我慢して使ってくれた．今はGUIで複雑になった上，多国語をサポートする必要があるから大変だ」と言っています．

また，最近の製品は同じラインナップでも，機能が豊富なため，機能が多少異なった数十種類の機種をサポートしなければならないことがあります．製品に入っているデバイスは機種ごとに微妙に異なっているため，それに伴ってパラメータなどが変わってきます．既存の方法でこれを書くのは本当につらいことです．

実はデータベースを使用すると，前述のように煩雑（はんざつ）になったデータを効率良く処理し，プログラム全体をすっきりさせることができます．

それでは，先に紹介した，「データベースを使った多国語対応と言語切り替え」について説明します．

2. GUIで表示するTextはメモリ上に置きたい

組み込み製品に搭載される非力なCPUで，凝ったGUIを快適に動かすのは大変です．ハードウェアへの負荷を減らすために，ボタンの名前やメッセージなどをメモリに常駐する必要があります．

ここで問題があります．Linuxなどの OSで動くデータベースは，大量のデータを大勢の人が同時に利用することを前提としているため，特定のデータをメモリに常駐できないという点です．これでは英語データベースや中国語データベースといった型式で，GUIの表示のロジックと実際に表示されるデータを分離できても，ディスク・アクセスが発生する可能性があるため，表示速度は一定になりません．しかし，本稿で紹介している組み込みデータベースDeviceSQL Framework（Encirq社，以下DeviceSQL）は，このような要望に応えられます．

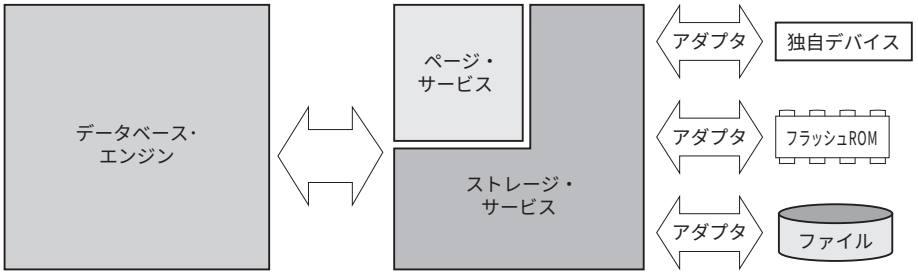

図1
サービスによるストレージの仮想化

KEYWORD ―― DeviceSQL Framework，仮想化層，ストレージ・サービス・アダプタ，表示言語の切り替え，エラー・コード，GUIの表示関数

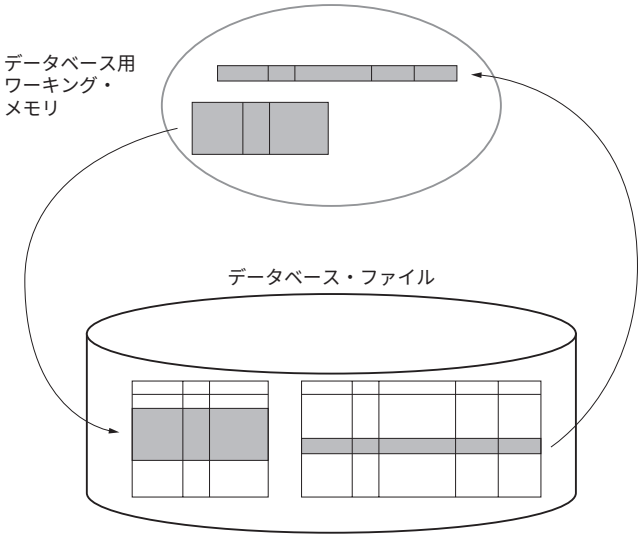

図2　データベースの実体がディスクにある場合

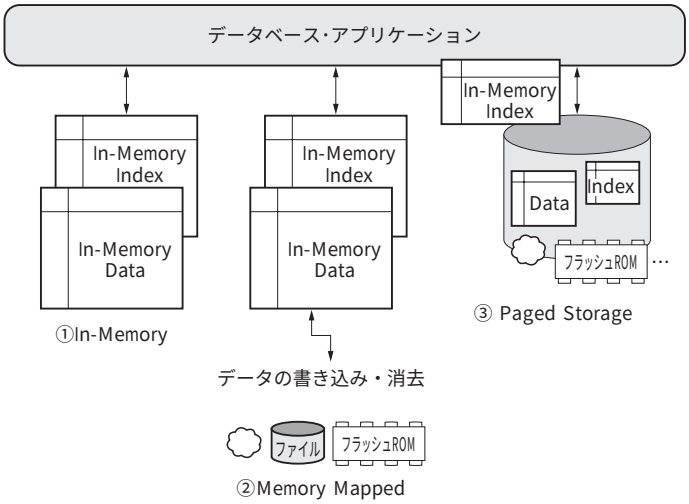

図3　In-Memory，Memory Mapped，Paged Storage

3. サービスとデータの配置

具体的な説明に入ります．まず，「他国語への対応，言語の切り替え」機能は，DeviceSQL の機能を使って実現します．そこで，プログラミングを開始する前に，簡単に DeviceSQL の構造と特徴を説明します．

図1に示すように，DeviceSQL のデータベース・エンジンは直接ストレージとインターフェースしているわけではなく，「サービス」という仮想化層を経由して実際のディスクやフラッシュ ROM とやり取りを行います．

また，実際にストレージ・デバイスとインターフェースしているストレージ・サービスは，アダプタと呼ばれるデバイスの特性の差を吸収するインターフェースを通じて，デバイスとやり取りを行っています．

このような構造をとっているので，特別なコンフィグレーションを行わなくても，各 OS ごと（一般的なリアルタイム OS，および Linux，Windows 系）のストレージ・サービス・アダプタの組み合わせでデータベース・アプリケーションを開発できます．また，ユーザ独自のデバイスにデータベースを作成したという場合，アダプタの開発だけで済みます（特殊なメディアを使わざるを得ない開発者も使用できる）．

表1　DeviceSQL のデータの持ち方

In-Memory	テーブルをストレージではなくメモリ内だけに作成する
Memory Mapped	テーブルはストレージに作成され，データベース・オープン時にメモリに展開される．テーブルの参照ではメモリ内のデータが参照され，データの追加や更新などではメモリ内およびストレージ内の両方のテーブルが書き換えられる
Paged Storage	テーブルはストレージに作成される． テーブルへの参照，追加，更新などは指定されたサイズのバッファを使用して行われる（一般的なデータベースの動き）

4. In-Memory，Memory Mapped，Disk Base

このような内部構成だと何が便利なのでしょうか，また何ができるのでしょうか．

図2に，データベースの実体がディスクにある場合を示します．データベースのエンジンは与えられたワーキング・メモリで動作しなければならないため，メモリにないデータはディスクに取りにいきます．そして，メモリ上で使われないデータは破棄されます．これはキャッシュと同じ動作です．そのため，データの常駐を指定することが難しくなります．

図3に，DeviceSQL のデータの持ち方を示す三つの形態があります．それぞれの詳細を**表1**に示します．ただし，ここでは話を分かりやすくするために，前回解説したスト