4H−9

統合化環境を目指したファイルアクセス法の提案

盛屋 邦彦　鈴鹿 豊明
日立ソフトウェアエンジニアリング（株）

1　はじめに

従来、文字、数値といった情報から、画像、図形、文章といったいわゆるマルチメディア情報を扱うアプリケーションが望まれている。現実にそういったソフトウェアパッケージが増えてきており、今後も益々の需要が見込まれる。

しかし、このような新しい情報を蓄積し、操作する際は依然としてＯＳが提供する従来のファイルアクセス法を利用することが多い。このため、いくつかの構造／操作上に制約があり、アプリケーションプログラムを作りにくい。

したがって、現実に利用できるデータファシリティは少なく、高機能で、より複雑な構造を扱えるファイルアクセス法が必要になってくる。しかし、単純にＯＳが提供するファイルアクセス法を新しい機能を持ったものに置き換えることはできない。なぜなら一方では現在までに作成したデータ処理を行うプログラムは資産として存在しその数は膨大なものであるからである。

そこで統合化ファイルアクセス法（ＩＦＡＭ）を提案した。これは、既存のファイルアクセス法の構造／機能を包含し、旧アプリケーションプログラムがそのまま利用できる。また、本アクセス法は無制限の可変長長大レコードや、マルチキーアクセス、レコード内操作など新しいアプリケーションにも対応できる。

2　ファイルアクセス法の統合化における要件

2.1　従来ファイルアクセス法の機能的な包含

まず最初に考えておかねばならない事は、既存ファイルアクセス法の機能を包含することである。

従来のファイルアクセス法はその性質からストリーム型とランダムアクセス型の２つに大別できると考えられる。

ストリーム型はそれぞれのデータ要素の間に（全）順序関係があり、データ処理もそれを基にして行う。データ要素はいわゆるレコード（更に細かいデータ要素の並び）かそれ以上分けられないデータ要素（ワード、バイト）となり、順序をもとにしたアクセスとなる。

一方、ランダムアクセスファイルでは、データ要素の間には順序関係はないが、各データ要素はキー（インデクス）を持ち、アクセスはそのキーに基づいて行う。データ要素はレコードが一般的である。

A New File Access Method for Integrated Environment
Kunihiko Moriya, Toyoaki Suzuka
Hitachi Software Engineering Co., Ltd.

2.2　新たに備えるべき機能

以下の機能は今までのアクセス法では不備であり、更に新しい情報形態（マルチメディアデータ等）を扱うために必要であると思われる。

1. 長大かつ可変長レコードが扱える
2. データ要素間に順序を持ち、その途中位置にデータ要素の挿入・削除が可能
3. 複数のキー付けが可能

従来のストリーム型のファイルアクセス法では、２や３の機能がなく、ランダムアクセス型のファイルアクセス法では１や３の機能がない。

かつて１、２を満たすファイルアクセス法としてマジックファイルを提案した。[有澤87] マジックファイルはストリーム型のファイルを抽象化したもので、巨大なデータの編集機能（無制限長レコードに対して挿入／削除機能）を持っている。また、データに対するインデクス付けに関しては、要素それ自身につく「マーク」という概念を導入し、論理的なレコードの実現および、そのランダムアクセスを可能とした。マークは通常のインデクス付けのメカニズムとは異なり、位置本位から、要素本位のものとしたところに特徴があった。これにより、更新におけるインデクスのメンテナンスが不要となったことが特徴である。しかし逆に、もともとレコードという概念がないため、論理的なレコードを作らざるを得ず、論理的な効率が良くなかった。

そこで本アクセス法では、このストリームとランダムアクセスの２つの概念を包含することを目標とした。

3　統合化ファイルアクセス法（ＩＦＡＭ）

3.1　ファイルの構造

1. 最小のデータ要素であるアトム（a, b, c, …で表す）
   アトムはＩＦＡＭが扱うデータの最小の単位であり、その表現はＩＦＡＭシステムの処理系の形態に基づく（バイト、ワード、等）。
2. ０個以上任意長のアトムの並びであるレコード（r, s, t, …）
   レコード内の各アトムは全順序関係を持っている。それぞれのレコードの長さはそのファイル内で一定でなくても良い。ファイル内のレコード間には順序関係はない。
3. レコードは一つ以上任意個のタグ（k, l, m, …）を持つ

タグの表現もそのシステムの実現形態による。レコードには複数のタグをつけることができ、同じタグを複数のレコードに付けられる。

4. 0個以上のレコードの集まりであるファイル（A, B, C, …）
ファイルは複数あってよく、空ファイルが許される。

構造としては通常のランダムファイル型のファイル構造に近い。特徴はタグとレコードの対応がm:nであることと、可変長のレコードが持てることである。この意味で、従来のランダムアクセスファイル型のアクセス法の構造を表現可能である。一方、ストリーム型の構造はレコード内のアトムの並びで表現する。しかし実際には、「順序に意味を持たせられる」ことが重要であり、操作の説明なしでは説明は不十分である。この問題については次節で議論する。

3.2 ファイルの操作

1. ファイル単位操作
   (a) ファイルの生成／削除
   (b) 指定ファイルの全レコード数の入手
2. レコード単位操作
   (a) 指定レコードの生成／削除
   (b) 指定レコードへのタグの追加／削除
   (c) 指定レコードの全タグ数の入手
   (d) 指定レコードに付加されたタグ集合の入手
   (e) 指定タグを持つレコード集合の入手
   (f) 全レコード削除（ファイルを空にする）
   (g) レコード長（指定レコード内総アトム数）の入手
3. アトム単位操作（レコード内操作）
   (a) レコード内へのアトム列の挿入
   (b) レコード内のアトム列の上書き
   (c) レコード内からのアトム列の削除
   (d) レコード内のアトム列の読み出し

タグには、特別のものとしてレコードの識別子があり、これはシステムにより自動的に付けられる。このレコード識別子により、操作すべきレコードを指定／特定できる。

IFAMのもう一つの特徴はアトム単位操作（レコード内の操作）があることである。既存のランダムアクセス型のファイルアクセス法ではレコードが最小単位であり、アプリケーションプログラムとそのレベルでのやりとりしか許されていなかった。一方、ストリーム型ではレコードが適当に小さな情報単位（例えばバイト）を扱っていた。IFAMでは、これらの2つのタイプのアクセス法の特性を合わせ持っている。すなわち、レコードを最小単位で見れば、ランダムアクセス型であり、アトムを最小単位（レコード内部）で見れば、ストリーム型である。実際にレコードの長さは論理的には無制限であり、1つのレコードをストリームファイルと見ることが可能である。

一般的にはレコードの長さは物理的な媒体の容量の限界に抑えられる。すなわち、レコード長は2次記憶の容量だけでなく、その処理系が採用するオペレーティングシステムの仮想記憶容量にも依存する。これらの物理的な制限を回避する意味でも、レコード内操作は有用である。すなわち部分的なデータ操作は処理系に依存しない、より論理的な操作を提供しているといえる。更に、既存のアクセス法になかった、途中位置でのデータ（アトム）の挿入削除も可能とした。

4 IFAMの実現

以上で述べてきたIFAMも効率の良い実現方法に支えられなければ意味がない。筆者らは前節の操作をファイル中にあるN要素数に対して少なくともO(log N)以上の効率で実現する方法を検討した。以下で述べるのはその一方法の概要である。

実現構成はは大きく分けるとタグとレコードの関係をm:nとするタグ管理部と可変無限長レコードを実現する要素管理部の各モジュールに分けられる。

タグ管理部はインデックスの一般的な実現手法であるB^+-treeを採用した。一般のB^+-treeではリーフノードにレコードの実体が入るが、ここではそのタグを持つレコードの集合（レコード識別子の木）へのポインタが入る。

次にレコード自体の実現について述べる。上で述べたようにレコードは無限可変長で、しかも途中への挿入削除がある。これを効率良く実現する手法としてOB-tree[STON84]を用いた。これは、B^+-treeのリーフノードに逐次的に要素を蓄積し、中間ノードにはキーの代わりに自分の子部分木中の要素数（これを重みという）を持たせる手法である。このOB-treeには、「レコードを指定してそのタグを得る。」という逆参照のために、先頭にそのレコードに付いている全てのタグを格納しておく。タグは固定長なのでルートノードにタグ数を入れておけば、要素の先頭がどこであるかすぐにわかる。

この方法によれば、すべての検索／更新操作の計算量も、通常のB^+-treeと同じであり、要素数に対して対数オーダとなる。

5 むすび

本稿では、ファイルの統合化環境を考察し、その要件を洗い出した。そして、その目的を達成するため、ファイルを抽象化し既存のファイルアクセス法をタイプ分けし、新／旧アプリケーションに対応可能なファイルアクセス法としてIFAMを提案した。さらに、その一実現手法を述べた。

参考文献

[有澤87] 有澤 博、鈴鹿 豊明: 長大テキストの効率良い格納と更新を可能とするファイル構造, 電子情報通信学会技術研究報告 OS87−2, 1987.

[STON84] M. Stonebraker, L. A. Rowe: Database Portals :A New Application Program Interface, Proc. VLDB, 1984.