情 報 学 基 礎 1－3
（1986・7・24）

高能率情報検システムの実現について

－キーワード転置ファイルの効率的構成－

篠原 武　二村 祥一　松尾 文碩
九州大学大型計算機センター

著者らは，AIRと名付けた情報検索システムの設計･開発を行ってきた．AIRの特長は，その優れた効率にあり，他の同種のシステムより，ディスク使用効率，検索処理速度，データベース更新処理速度のすべてにおいて格段に優っている．しかし，最近の調査によって，AIRに改良の余地があることがわかった．本稿では，AIRをさらに高能率化するためのキーワード転置ファイルの設計について議論する．キーワードが生起する文献の間隔が非常に偏った分布をすることに注目し，転置ファイルのサイズを40%程度に圧縮する単純かつ効率的な技法を提案する．また，高頻度キーワードと低頻度キーワードを別々に管理する，転置ファイルの構成を提案する．新しい構成の転置ファイルを用いると，文献の追加に伴う転置ファイルの更新処理を高速に行え，しかも追加を繰り返しても検索処理速度に悪影響しない．

On Realization of Efficient Information Retrieval System

– Efficient Organization of Inverted Files for Keywords –

by
Takeshi SHINOHARA, Fumihiro MATSUO and Shouichi FUTAMURA
Computer Center, Kyushu University
6-10-1, Hakozaki, Higashi-ku, Fukuoka 812, Japan

The authors have designed and developed an information retrieval system named AIR. AIR is distinguished from other systems of the same kind by its high efficiency in all of disk space, retrieval speed, and database maintenance. The recent research, however, identifies some possibilities to improve the efficiency of AIR. In this paper, we discuss the design of inverted files to make AIR more efficient. We propose, with attention to the uneven distribution of document intervals of keyword occurrence, a simple and efficient technique that reduces the size of inverted files to about 40%. We also propose an organization of keyword inverted files which manages keywords of high frequency and keywords of low frequency separately. By using new organization, inverted files can be updated very quickly when documents are added, and further the accumulation of updating does not badly affect the responsibility of retrieval.

1. はじめに

AIR[1]は，九州大学大型計算機センター(以下，九大センターと略す)において設計･開発された文献情報検索システムであり，1983年秋より，これを用いてINSPECなどの文献データベースのオンライン検索サービスを行っている[2]．AIRの特長は，その優れた効率にあり，他の同種のシステムより，ディスク使用効率，検索処理速度，データベース更新処理速度のすべてにおいて格段に優っている．AIRの高性能は，文献データベース中に含まれる単語の統計的性質を利用したデータ圧縮技法[3]，不要語選択法[4]，高速単語索引[5]や，検索実験に基づいた転置ファイル構成法[6]などの技法を用いて実現されている．

しかし，最近の調査によって，不要語選択法の改良や転置ファイルの圧縮など，AIRに改良の余地があることがわかった．これまでの不要語は，文献集合に大きく依存しており，不要語の数も膨大であるため，管理面に問題があった．不要語選択法の改良[7]によると，複数の文献集合に共通の不要語を新しく不要語として採用することにより，転置ファイルのサイズをほとんど増加することなく，不要語数を1,600程度にすることができ，管理面の問題を解決できる．また，転置ファイルの圧縮[8]によると，キーワードが生起する文献の間隔が非常に偏った分布をすることを利用すれば，転置ファイルのサイズを2.5分の1以下に圧縮することができ，検索処理速度を向上することができる．

本稿では，AIRをより効率化するためのキーワード転置ファイルの構成法について議論する．まず第一に，上で述べた圧縮技法を用いることにする．これは，ディスク領域の節約だけでなく，検索処理の高速化にとっても非常に有効である．転置ファイルの構造はデータベース更新処理の効率にも大きな影響を与える．これまでのAIRの転置ファイルには，更新を繰り返すと処理速度が低下するという問題があった．これは，頻度の高いキーワードに対する領域が頻度の低いキーワードに対する領域によって寸断されることや，更新を受ける領域が散在することなどによる．この問題については，低頻度のキーワードの生起特性[9]も考慮して転置ファイルの構造を設計することによって解決できる．

2. キーワード転置ファイルの構造

キーワード転置ファイルは通常，図1に示すように，索引部と文書参照ファイルから構成される．

索引部には，キーワードとそれに対する文書参照ファイル中のデータへのポインタが格納される．索引部は，通常B木を用いて実現されるが，AIRでは2次記憶上におく高さ1の多岐平衡木と1次記憶上おく高順位の単語索引を用いて高速な検索を可能としている[5]．文書参照ファイルの構造は，転置ファイルにキーワード生起位置情報を持つかどうかによって，すなわち隣接演算を転置ファイルだけで実現するかどうかによって異なるが，AIRでは隣接演算は逐字サーチを用いて実現しているので[10]，ここではキーワード生起位置情報を持たない転置ファイルを考察する．したがって，各キーワードに対する文書参照ファイル中のデータは，そのキーワードが現れている文書の参照番号リストである．

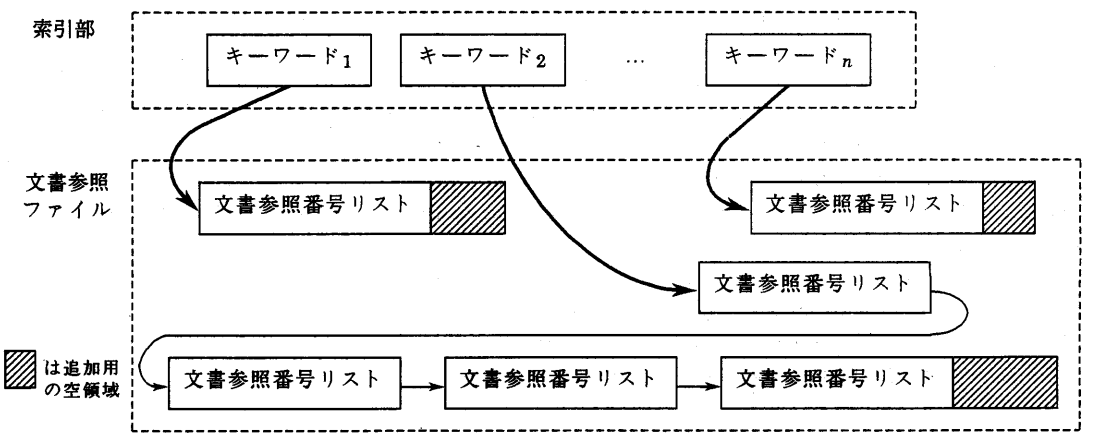


図1. キーワード転置ファイルの構造

キーワード転置ファイルの構造は，利用者にとって重要な検索処理速度や，管理者にとって重要なデータベース更新処理速度など，情報検索システムの効率と関係が深い．

2.1 検索処理速度

キーワード転置ファイルを用いると，各キーワードに対してそれを含む文献の集合を高速に求めることができる．さらに，各キーワードに対して得られた文献集合をブール演算を用いて組み合わせて，複雑な検索を行うこともできる．したがって，キーワード転置ファイルの構造は情報検索システムの最も基本的な効率である検索処理速度を決定するといえる．

キーワード転置ファイルのうち，索引部については，上でも述べたように，高速単語索引の技法を開発しており，この技法を用いればブール演算を必要としない単純な検索処理の応答速度が飛躍的に向上することを確認している[11]．しかし，ブール演算を伴う検索処理は高速単語索引を用いても応答速度は10%程度しか改善できなかった．この結果から，索引部の構造はブール演算の処理効率にそれほど大きな影響を与えないことがわかる．

一方，文書参照ファイルの構造はブール演算の処理効率との関係が深いにもかかわらず，これに関する研究はこれまでほとんど行われていない．筆者らの経験によると，ブール演算の処理時間の大部分は2次記憶参照によるものである．現在のAIR(AIR第1版と呼ぶ)では，1回のディスク参照の単位を大きくすることにより参照回数を減らし処理の高速化を行っている．今回の改良では，文書参照ファイルの圧縮技法を用いて，さらにブール演算の高速化を行う．

2.2 キーワード転置ファイルの更新処理

キーワード転置ファイルの更新処理の効率は，ブール演算処理の効率と同様に，非常に重要である．実際，九大センターでAIRを開発するまでに使用していたシステムでは，この処理に半日以上かかることもあり，定期的に検索サービスを停止しなければな

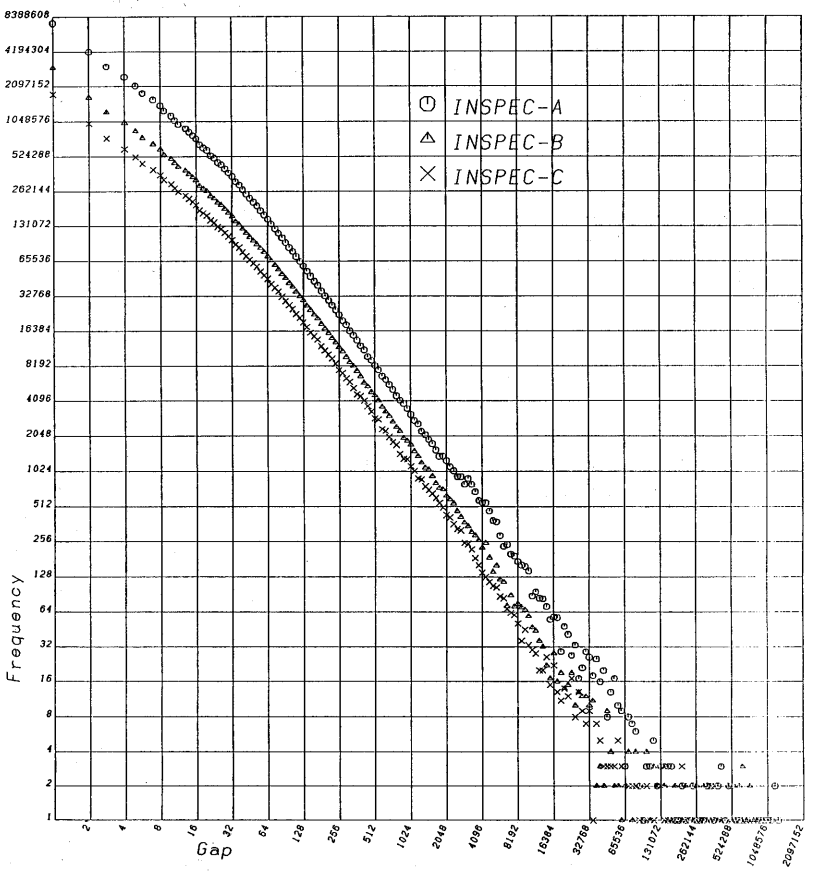


図2. INSPEC文献データベースにおけるGapの分布

らなかった．AIRはこれを10倍以上高速にすることに成功している．

しかし，AIR第1版では，更新を積み重ねると徐々に処理速度が低下することがわかった．この原因は，頻度の高いキーワードに対する大きな文書参照番号リストが頻度の低いキーワードに対するものによって寸断されることや，更新処理を受けるディスク領域が散在するようになることにあると考えられる．

そこで，今回の改良では，更新処理を高速に行うことができ，しかも更新を繰り返しても処理速度が低下しないようにキーワード転置ファイルの構造を再設計する．

3. 文書参照ファイルの圧縮

まず，文書参照ファイルの圧縮について考察しよう．文書参照ファイルは各キーワードが出現する文書の参照番号を小さい順に並べたリスト$r_1, r_2, \ldots, r_n$から構成されている．このとき，$g_1=r_1$, $g_i=r_i-r_{i-1}$ $(i=2, \ldots, n)$と定義する．このキーワードの出現文書間隔g_iをgapと呼ぶ．ここで提案する圧縮技法はgapの分布が非常に偏っていることを利用する．

3.1 INSPECにおけるGapの分布

九大センターでは，AIRを用いてINSPEC文献データのオンライン検索サービスを行っている．そのキーワード転置ファイルの文書参照ファイル中のgapの大きさと頻度の関係を調べてみた．図1は分野別にわけたINSPECの3つのデータベースINSPEC-A(物理学，171万件)，同-B(電気･電子工学，88万件)，同-C(制御工学･計算機科学･情報工学，60万件)に対する調査結果を表したもので，各データベースの全gap数はそれぞれ7,122万，3,213万，2,015万であった．この図から，gapの分布は，分野にほとんど関係なく，非常に偏っていることがわかる．

3.2 Gapの符号化(Gap Code)

Gap gの符号$c(g)$を図2のように定義する．この符号化法は，INSPECの文献データ圧縮に用いている英文テキストの圧縮法QOC[3]と基本的に同じであり，4ビット固定長のprefixと可変長の部分からなる．Prefixは可変長の部分の長さを表している．しかし，4ビットでは最大15の長さしか表せないので，prefixが15の場合は残りを24ビット固定長としている．

3.3 Gap Codeの圧縮効率

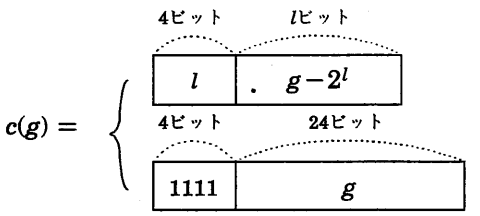
Gap CodeのINSPEC文献データベースに対する圧縮効率を求めてみよう．Gap gが現れる確率を$p(g)$，gapの最大値をG，Gap Codeの符号長を$|c(g)|$で表すと，gapのエントロピーHおよびGap Codeの平均符号長Iは，それぞれ

$$H = \sum_{g=1}^{G} -p(g)\log_2 p(g),$$

$$I = \sum_{g=1}^{G} p(g)|c(g)|$$

で求められる．圧縮効率はH/Iで与えられる．

表1はINSPEC文献データベースに対して求めたH, IおよびH/Iである．この結果からGap Codeの圧



c(g) =

g	c(g)
0	0000
1	0001 0
2	0001 1
3	0010 00
4	0010 01
5	0010 10

図3. Gapの符号化

表1. Gap Codeの圧縮効率．

データベース	全gap数	エントロピー (bit)	平均符号長 (bit)	圧縮効率 (%)
INSPEC-A	71,218,294	8.2736	8.7691	94.35
INSPEC-B	32,128,800	8.4964	8.9684	94.74
INSPEC-C	20,154,138	8.6925	9.1534	94.96

縮効率はほぼ最適であることがわかる．AIR第1版では文書参照番号を表すのに24ビット用いているので，Gap Codeを用いればキーワード転置ファイルを2.5分の1程度に圧縮できることもわかる．また，別の調査によって，gapの分布は文書の順序にほぼ無関係であり，Gap Codeはほとんどの文献データベースに対して有効であることもわかっている．

4. キーワード転置ファイルの設計

キーワード転置ファイルの設計において注意すべきことは，次の3点である：

(1) 各キーワードに対する文書参照番号リストへのアクセス速度．

(2) 文書参照ファイル全体でのディスク使用効率．

(3) 更新処理の速度．

ただし，更新処理は文献データの追加に伴って行うものだけを考える．これは，AIRが対象とするのは，INSPECのような文献データであり，削除·修正などの追加以外の更新処理を必要としないものであるからである．

ディスク装置とのアクセスはブロック単位で行われる．(1)は文書参照番号リストを出来るだけ少ないブロックに格納すればよい．(2)は出来るだけ詰め合わせて文書参照番号リストを格納すればよいが，更新処理を行うと，ファイル全体の再構成をしないと文書参照番号リストが寸断されてしまう．そこで，(1)と(2)をある程度両立させ，(3)を犠牲にしないために，文書参照番号リストの末尾に適当な大きさの追加用空領域を設ける方法が考えられる．この方法はAIR第1版で用いられている．しかし，この方法では不十分であることがわかる．

4.1 AIR第1版の問題点

AIR第1版のキーワード転置ファイルでは，構築直後の更新処理は，各文書参照番号リストの末尾の空領域にデータを追加するだけでよいので，高速に行え，しかも文書参照番号リストが寸断されることもない．しかし，更新を繰り返して追加用の空領域がなくなると，新たに確保したディスク領域をを連結するので，文書参照番号リストは徐々に寸断され，検索処理速度ばかりでなく，更新処理速度も低下する．したがって，AIR第1版では定期的に転置ファイルを再構成する必要がある．

キーワード転置ファイルの更新は，文献毎に処理するよりも，追加された文献を一括して処理する方が効率がよい．一括処理の場合には，キーワードの辞書式順序で文書参照番号リストを更新するので，追加によって書き換えられるディスクの領域がキーワード順に並んでいると，1回のブロックへのアクセスで複数のキーワードの処理が行え，しかも同一ブロックを2回以上アクセスする必要がない．AIR第1版のキーワード転置ファイルでは，データベース構築直後には追加のための空領域がキーワード順に並んでいるが，更新を繰り返すとその順序が乱れてくる．これが更新処理速度の低下の最も大きな原因である．

文献データベースのキーワードの頻度は非常に偏っている．頻度の高いキーワードに対する文書参照番号リストは追加される文献数にほぼ比例して大きくなるので，追加のための空領域の大きさは頻度の関数として設定できる．頻度の低い個々のキーワードに対してはそのようなことは成り立たないが，同頻度のキーワードのうちで追加が行われるものの個数を追加される文献数によって予測することができる．頻度の低いキーワードの個数は非常に多いため，個々のキーワードに対して空領域を設けるとディスクの使用効率を悪くする．したがって，文書参照番号リストの管理は高頻度のものと低頻度のものに分けて行う必要がある．AIR第1版では，すべてのキーワードを一様に管理しているので，高頻度キーワードの文書参照番号リストが低頻度のものによって寸断され，検索処理速度の低下を招いている．また，更新を繰り返した場合に追加のための空領域の順序が乱れるのも，低頻度キーワードの影響が大きいと考えられる．

4.2 高頻度キーワード

高頻度キーワードの文書参照番号リストの追加用領域の大きさは，キーワードの頻度と文献追加の割合によって決定しなければならない．追加用領域が大き過ぎると1回のディスクブロックの読み書きで処理できるキーワード数が減少し，逆に小さ過ぎると領域が不足し領域を追加する回数が増加する．

追用領域を格納するブロックには追加用領域だけを格納し，キーワード順に格納するようにする．このようにすれば，確実に1回のディスクブロックの更新によって複数のキーワードの処理ができる．追加用領域が不足すると新たにディスク領域を確保し連結しなければならないが，図4に示すように，新しい領域を追加用領域の後ろに連結せずに直前に連結しデータを移動すれば，追加用領域はもとの位置に置くことができる．

また，連結を繰り返すと文書参照番号リストを格納するのに使用されるブロック数が多くなり，検索処理速度が低下するので，これを防止するために，連結

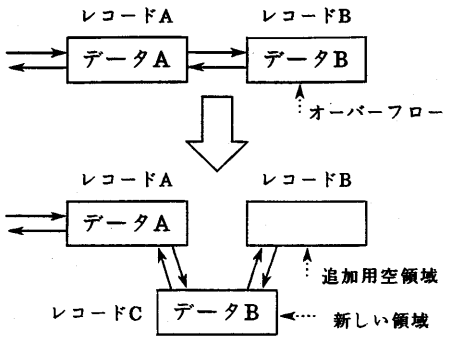


図4. 領域の追加

を数回行う毎に領域の詰め合わせを行い，追加用領域の大きさも変更する．もちろん，新しく確保した追加用領域はキーワード順に並ぶように配置しなければならない．このために，追加用領域の大きさによってキーワードをグループ化し，各グループ毎に追加用領域をキーワード順に格納する．

高頻度のキーワードが出現する文献の間隔(gap)は小さいものが多いので，その文書参照番号リストはGap Codeによって効率よく圧縮できる．

4.3 低頻度キーワード

低頻度キーワードは，頻度によってグループ化し，高頻度キーワードの追加用領域と同様にキーワード順に並べておく．低頻度のキーワードが追加によって頻度が増加し領域が不足した場合には，単にもとの領域を解放し別の頻度グループに領域を移動する．頻度1および2のキーワードについては，索引部に文書参照番号リストへのポインタの代わりに直接文書参照番号を格納する．もちろん，追加を繰り返すと低頻度から高頻度へ移動する場合もある．

低頻度キーワードの文書参照番号リストに対しては，Gap Codeの圧縮効率が低いので圧縮は行わない．

5. AIR第2版のキーワード転置ファイル

これまでの考察に基づいてAIR第2版のキーワード転置ファイルの構造を決定する．

5.1 索引部

索引部は第1版と同様の高速単語索引を用いる．各索引レコードには，キーワード，キーワード長，文献数，文書参照番号リストの先頭および末尾へのポインタが格納される．ただし，文献数が1または2の場合には，ポインタの部分に直接文書参照番号を格納する．

5.2 文書参照ファイル

文書参照ファイルには各キーワードに対する文書参照番号リストを格納する．頻度が128以下のキーワードを低頻度，それ以上のものを高頻度と呼ぶ．

文書参照ファイルはディスクブロックの集まりである．ブロック長は使用装置のトラック長によって決定する．トラック長が32Kバイト以下の場合には，トラック長をブロック長とする．そうでない場合には，トラック長をn等分したもので32Kバイトを超えない最大のものをブロック長とする．ただし，実際にはn等分したものより少し小さくなる．現在九大センターで用いているIBM 3380タイプの装置では，トラック長は47Kバイトであるので，ブロック長は23.2Kバイトとする．

各ブロックの先頭は管理のために用いる．ブロックの残りを32等分したものをセクターと呼ぶ．高頻度キーワードの文書参照番号リストはセクターに格納する．ブロックの先頭には，次ブロック・前ブロックへのポインタ，ブロック内でのセクターの使用状況をあらわすビット列，追加用領域をキーワード順に配置するためのブロック内での最大・最小のキーワードを格納する．セクターはGap Codeによる圧縮の単位となる．文書参照番号リストの各セクターは双方向ポインタで連結する．同一ブロック上の連続したセクターをレコードと呼ぶ．レコードは，1, 2, 4, 8, 16および32セクターの大きさのものを用いる．同一ブロックには同じ大きさのレコードを格納する．

高頻度キーワードを文書参照番号リストの大きさによって分類する．4セクター以下のものを第1類，5~8, 9~16, 17~32, 33~64セクターのものをそれぞれ第2, 3, 4, 5類，それより大きいものを第6類と呼ぶ．第1類は1セクターの大きさのレコードに格納する．第2類から第5類までのものは，それぞれ先頭の4, 8, 16, 32セクターを対応する大きさのレコードに格納し，残りをその4分の1の大きさのレコードに格納する．第6類のものは，末尾は16セクター，それ以外は32または16セクターの大きさのレコードを用い，先頭から詰め合わせて格納する．たとえば，25個のセクターを必要とする文書参照番号リストは，第4類であり，先頭から16番目までのセクターは大きさ16のレコード，17番目から24番目までは大きさ4のレコード，最後は大きさ4のレコードに格納され，最後のレコードには3セクター分以上の領域が追加用に残される．この方法では，第6類以外の文書参照番号リストは最大5個のレコードに格納される．

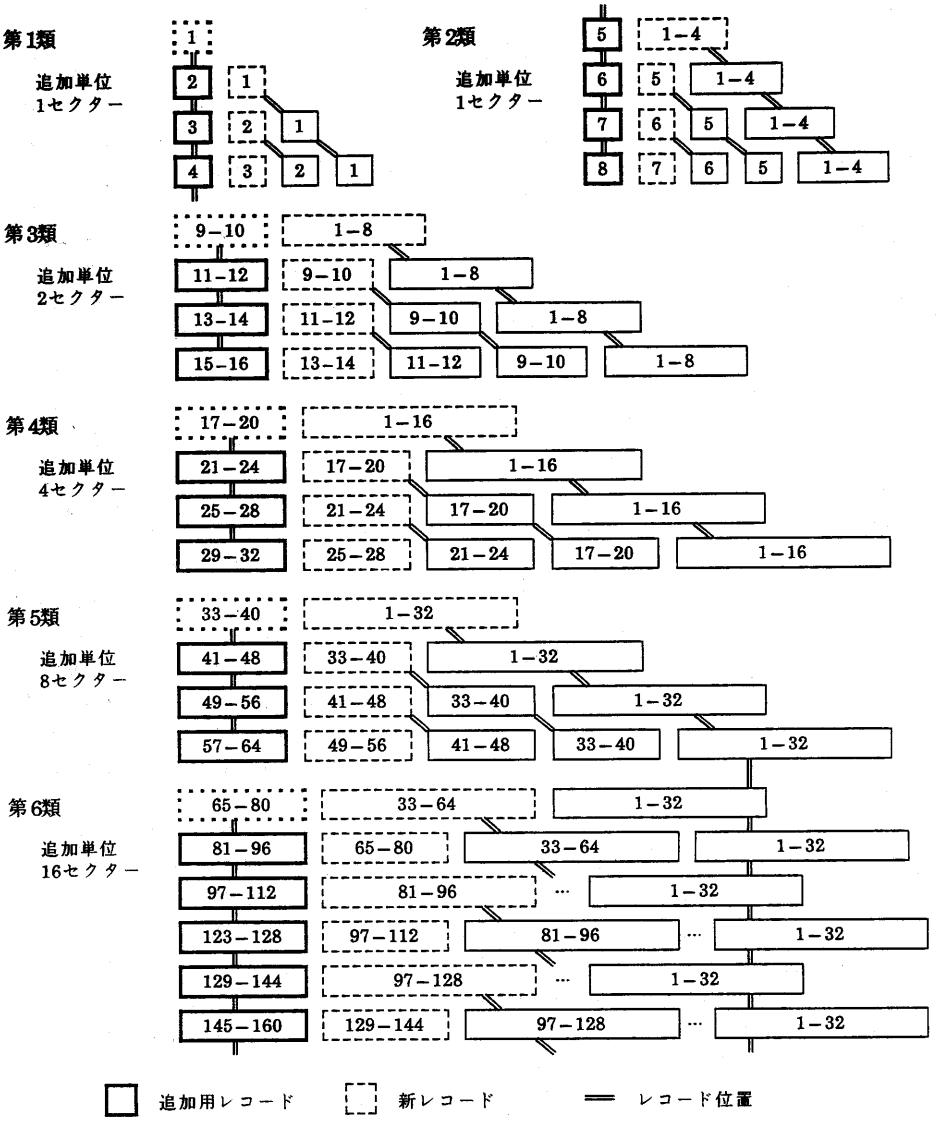


図5. 高頻度キーワードの文書参照番号リストの構造

図5は，各類の文書参照番号リストの格納のようすを表したものである．各文書参照番号リストの右端が先頭のセクターを格納するレコード，左端の太線で囲んだものが末尾の追加用のレコード，破線で囲んだレコードは一番新しく確保されたレコードである．二重線で結ばれたレコードは，追加によって文書参照番号リストが大きくなっていくときに移動しないレコードを表している．

各類に属する文書参照番号リストの末尾の追加用レコードは同じである．同じ大きさの追加用レコードは専用のブロックにキーワード順に並べて格納する．このようにすると，各類の追加処理は追加用レコードが一杯にならない限り，追加用レコードが格納されているブロックを一方向にアクセスするだけで完了する．追加用レコードが一杯になった場合には，同じ大きさのレコードを新しく確保し，追加用レコードの内容を新レコードに移動し，新レコードを追加用

レコードの直前に連結する．ただし，各類の最大の大きさを超えるときは，対応する類へ移動する．新レコードを追加用レコードの後ろに連結しない理由は，追加用レコードをキーワード順に維持するためである．また，類を移動する場合には，新しい追加用レコードをその類のキーワードに対応するブロックに格納しなければならない．そのために，追加用レコードを格納するブロックには，最大･最小のキーワードを格納しておく．

最後に，低頻度キーワードの取り扱いについて述べよう．低頻度のキーワードに対してはGap Codeは用いない．頻度が1または2のものは，すでに述べたように，索引部のレコードに文書参照番号を格納する．頻度が3以上128以下のものは，頻度によって分類し，高頻度キーワードの追加用レコードと同様に，各類毎にキーワード順に配置する．あまり分類を小さくすると，追加処理時に必要なバッファが多くなるので，3~4, 5~8, 9~16, 17~32, 33~64, 65~128の6つに分類する．このようにするとディスク領域の使用効率が50%程度となる場合も考えられるが，平均的には70%程度であり，しかもINSPECにおいて低頻度キーワードが占める領域は全体の10%に満たないので問題はない．

6. おわりに

本稿では，高能率の情報検索システムの実現について，著者らが開発した高性能のAIRをさらに効率よくするための，キーワード転置ファイルの再設計を中心に議論してきた．新しく設計したキーワード転置ファイルでは，低頻度と高頻度のキーワードを別々に取り扱うので，低頻度のものが高頻度のものに干渉し検索処理速度が低下することを防げる．また，高頻度キーワードに対する転置ファイルの領域は，Gap Codeによって効率よく圧縮できる．追加処理によって更新を受ける領域をキーワード順に配置するので追加処理も高速に行える．しかも，追加を繰り返しても，領域の再配置を適時行うので，追加や検索の処理速度は低下しない．以上のように，今回の再設計によって改良の目的は十分達成できるであろう．現在，新しい設計に基づいてAIRの改訂作業を急いでいる．効率の向上などについては，新システムを用いて実験を行い報告する予定である．

参考文献

[1] Matsuo, F., Futamura, S. and Shinohara, T.: Efficient Storage and Retrieval of Very Large Document Databases, Proceedings of thee Second International Conference on Data Engineering, (1986), 456-463.

[2] 二村，篠原，松尾: 情報検索システムAIRによるINSPECの検索，九州大学大型計算機センター広報，Vol.17, No.1, (1984), 1-22.

[3] 松尾，二村，吉田: 準最適テキスト圧縮符号，九州大学工学集報，Vol.55, No.2, (1982), 103-106.

[4] 松尾，二村，高木，吉田: INSPECデータベース転置ファイル生成における不要語選択法，九州大学工学集報，Vol.54, No.2, (1981), 99-105.

[5] 松尾，二村，高木，吉田: 高速検索のための単語辞書索引の一構成法，九州大学工学集報，Vol.54, No.3, (1981), 183-187.

[6] 篠原，二村，松尾: 転置ファイル内のキーワード生起位置情報について，情報処理学会第26回全国大会講演論文集，(1983), 7F-3.

[7] 二村，松尾，篠原: 不要語の選択法について，情報処理学会第31回全国大会講演論文集，(1985) 8B-9.

[8] 篠原，松尾，二村: 高速ブール演算のための効率的転置ファイル構成法，情報処理学会第31回全国大会講演論文集，(1985), 7B-9.

[9] 松尾，二村，篠原: 科学技術文献に関するZipf-Boothの法則について，情報処理学会第29回全国大会講演論文集，(1984), 3G-1.

[10] 篠原，二村，松尾: 逐字文字列照合法による隣接演算の実現とその評価，情報処理学会第30回全国大会講演論文集，(1985), 4U-6.

[11] 二村，篠原，松尾: 情報検索システムAIRの性能評価，情報処理学会第29回全国大会講演論文集，(1984), 7F-5.