データベースシステム 96－8
（1993．10．29）

誤ったキーでも検索できるファイル構成法

平出基一　　田中榮一

神戸大学工学部

本研究は，誤ったキーでも検索できるファイルの構成法とその操作方法について述べている．データベースシステムが大型化するにつれ，データが誤って記憶されることも多くなる．また，人は誤ったキーを記憶していることもある．さらに，音声の機械認識では認識出力にいろいろな誤りがあることが多い．このような事態に対処するために類名表記を用いた階層ファイルが幾つか提案されている．しかし，キーの挿入・削除を考慮しているものはない．ここで提案するファイルは，キーの挿入・削除を容易に行えるようにこの階層ファイルの構成にB木を用いたものである．長さ6～10の英単語16,651語を用いて，検索・更新のコスト，および記憶利用率を調べ，本方法の有用性を確かめた．

FILE ORGANIZATION RETREIVABLE BY AN INCORRECT KEY

Motoichi Hirade　　Eiichi Tanaka

Department of Electrical and Electronics Engineering, Faculty of Engineering,
Kobe University,

1-1 Rokkodai-cho Nada-ku, Kobe, 657 Japan

This paper describes file organization retreivable by an incorrect key and the method of operations. The larger a database system becomes, the more erroneous data a file has. Sometimes we remember an incorrect key. Furthermore, in speech recognition it is not seldom for a recognition machine to output an erroneous result. To cope with these problems hierarchical files using class name expressions have been proposed. These files do not have suitable structures for inserting new keys. We propose a new file organization suitable for inserting and deleting keys. An experiment was carried out using 16,651 words of length 6 ~ 10. The result showed that the file had a good performance.

1 まえがき

データベースシステム等に利用されるファイル構成法として，B木およびそれらの拡張版[1, 2, 3, 4]がよく知られている．これらは，正しく記憶されたデータを正しいキーにより効率よく検索できる．しかし，データが大量になると，データが誤って記憶されることもある．また，利用者が検索キーを入力する場合，誤ったキーを入力してしまうこともあるし，音声入力では，確実に正しいキーを入力することは難しい．誤りに対する研究として，[5, 6, 7]では，B木の構造が壊れた場合，すなわち，索引やポインタが誤った場合，その誤りを検出したり，訂正する方法を提案している．一方，データの誤りに対する研究では，類名表記を用いた階層ファイル[9, 10, 11]が提案されている．この階層ファイルでは，検索キーがファイルに存在しないとき，最も近いキーをファイル中から探すことができる．しかし，実際のデータベースシステムでは，新たなキーの挿入やファイルにあるキーの削除が行われるのに対し，この階層ファイルではキーの挿入・削除を行うことを考慮していなかった．

本研究は，類名表記を用いた階層ファイルの構成にB木を用いることにより，キーの挿入・削除を容易に行えるようにするものである．

2 類名表記と重みつきレーベンシュタイン距離

ここでは，キー集合の分類法と，キー間の類似測度に用いるレーベンシュタイン距離について述べる．

2.1 キー集合の分類法

Σを文字の集合とし，次の条件を満たす類 S_iに分類することを考える．

(i) $\Sigma = \bigcup_i S_i$,

(ii) $i \neq j$ のとき，$S_i \cap S_j = \phi$.

ここで，ϕは空集合である．例えば，Σをアルファベットの集合とし，次の4つの類に分類する．

$S_0 = \{a, b, c, d, e, f\}$,
$S_1 = \{g, h, i, j, k, l\}$,
$S_2 = \{m, n, o, p, q, r, s\}$,
$S_3 = \{t, u, v, w, x, y, z\}$.

ここで，$S_0 \sim S_3$を類名と呼ぶ．キーを類名を用いて書くことを考える．例えば，キーappleを上記の類名を用いて書くと $S_0S_2S_2S_1S_0$となる．これをappleの類名表記といい，E(apple)と書く．ここで $S_0S_2S_2S_1S_0$ と書く代わりに02210と書けば類名表記は数値とみなせる．類名の数を n とすると類名表記は n 進数の数値となる．同じ値2210となる類名表記 $S_0S_2S_2S_1S_0$と $S_2S_2S_1S_0$ を区別するため，数値で表した類名表記の先頭に必ず1を付ける．するとappleの類名表記は102210となり，これを E(apple)=102210と書く．類名表記を用いればキー集合を分類できる．図1は，類名表記を用いてキー集合を分類した例である．

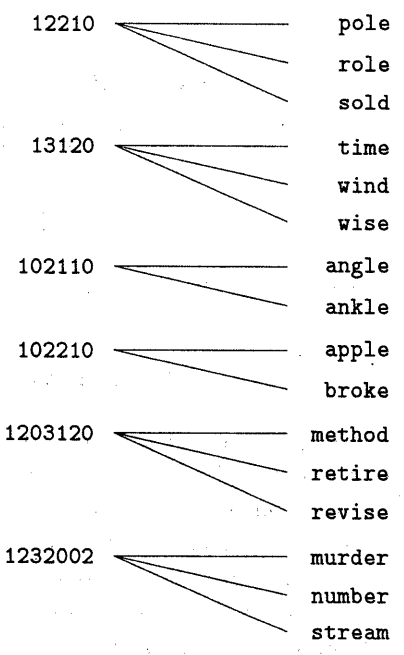


図 1: 類名表記を用いたキー集合の分類の例

2.2 レーベンシュタイン距離

キー間の類似測度として1次元重みつきレーベンシュタイン距離 (WLD) を用いる．系列 $X = x_1x_2\cdots x_m$ と $Y = y_1y_2\cdots y_n$ の間に次の条件を満たす写像 M_s が定義されているとする．x_i が y_j に写像されているとき，(i,j) と書く．

1. $(i,j) \in M_s$ のとき，
   $1 \le i \le m$, $1 \le j \le n$.
2. (i_1,j_1), $(i_2,j_2) \in M_s$ のとき，
   (a) $i_1 = i_2$ iff $j_1 = j_2$,
   (b) $i_1 < i_2$ iff $j_1 < j_2$.

$|M_s|$ を M_s の元の数，u_s を M_s の元 (i,j) のうち $x_i \neq y_j$ であるものの数とし，$v_s = n - |M_s|$, $w_s = m - |M_s|$ とする．u_s, v_s, w_s はそれぞれ，置換，挿入，脱落の数である．このとき，X から Y への距離 $D(X,Y)$ は，次のように定義される．

$$D(X,Y) = \min_s\{p \cdot u_s + q \cdot v_s + r \cdot w_s\}.$$

ここで，p, q, r はそれぞれ，置換，挿入，脱落の重みである．通常，$p < q+r$ と仮定する．$D(X,Y)$ は，次の式で回帰的に計算できる．

$$d(i,j) = \min_s\{d(i-1,j-1) + p(i,j), \\ d(i,j-1) + q, d(i-1,j) + r\}.$$

ここで，$d(0,0) = 0$, $d(i,0) = i \cdot r$, $d(0,j) = j \cdot q$,

$$p(i,j) = \begin{cases} p\ , x_i \neq y_j \text{のとき}. \\ 0\ , x_i = y_j \text{のとき}. \end{cases}$$

このとき，

$$D(X,Y) = d(m,n).$$

ここで，$q = r$ のときは距離公理を満たすが，そうでないときは，$D(X,Y) = D(Y,X)$ が成り立つとは限らない．

3 ファイルの構成

類名表記を用いた階層ファイルで，挿入・削除を容易に行えるようにするには図2に示すように

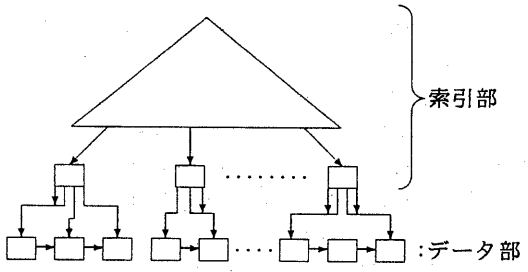
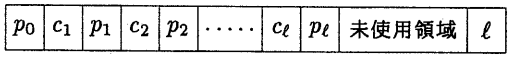


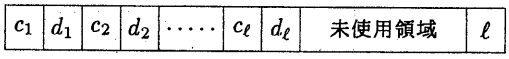
図 2: ファイル構成

ファイルを構成する．ファイルは，類名表記からなる索引部とキーからなるデータ部に分けられる．索引部は B$^+$木として構成する．2次記憶との入出力の最小単位をページとし，B$^+$木の節点をページに割り付ける．また，すべての類名表記は索引部の葉のページに格納し，葉以外のページは，各葉のページの最も大きい類名表記からなる．データ部は複数の連結リストで構成し，連結リストはページを要素とする．また，データ部のページと索引部の葉のページにも親子関係を持たせる．1つの連結リストは同じ親を持つページからなる．

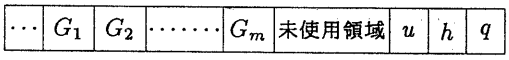
図3(a)~(c)は，ページの構造である．索引部のページには，類名表記 c_i とポインタ p_i あるいは d_i があり，p_i, d_i により親子関係が示される．p_i は索引部の葉以外のページにあり，索引部のページを指す．d_i は索引部の葉のページにあり，データ部のページを指す．また，(c_i,p_i) あるいは (c_i,d_i) の組をエントリと呼ぶ．索引部のページには最大 k



(a) 索引部の葉以外のページ



(b) 索引部の葉のページ



(c) データ部のページ

図 3: ページの構造

個の類名表記が格納でき，実際に格納されている類名表記の数 ℓは，ページの特定の場所に記憶されている．

データ部の $(c_i, n_i, x_{i,1}, x_{i,2}, \cdots, x_{i,n_i})$ の組をグループ G_iと呼ぶ．ここで，n_iは類名表記 c_iを持つキーの数であり，$x_{i,j}$は c_iを持つ j番目のキーである．1つのグループ内のキーは辞書順に並べられている．データ部のページで実際にグループが格納されている領域を使用領域と呼ぶ．使用領域のサイズを表す変数 u と，ページ内の最初の類名表記がある位置を表す変数 h と，連結リストをなすためのポインタ変数 qが，ページの特定の場所に記憶されている．

提案するファイルは次の条件を満たす．

(1) 索引部の1つのページ内の類名表記数 ℓは，

$$\begin{cases} 1 \le \ell \le k & : \text{根のページ} \\ \left\lceil \frac{k}{2} \right\rceil - \sigma_i \le \ell \le k & : \text{葉のページ} \\ \left\lfloor \frac{k}{2} \right\rfloor \le \ell \le k & : \text{その他のページ} \end{cases}$$

である．ここで，σ_iは索引部の葉のページを分割するときに最適な分割点を決めるためのパラメータである．

(2) データ部のページ内の使用領域サイズ u は，

$$\left\lceil \frac{U_{max}}{2} \right\rceil - \sigma_k \le u \le U_{max}$$

である．ここで，$U_{\max}$はページの使用領域サイズの最大値，σ_kはデータ部のページを分割するときに最適な分割点を決めるためのパラメータである．

(3) 索引部の葉以外のページを探索する場合，類名表記 c_1，c_2，$\cdots$，c_ℓがあるページでは，検索キー yの類名表記 $E(y)$ が $E(y) \le c_1$なら p_0，$c_i < E(y) \le c_{i+1}$なら p_i，$c_\ell < E(y)$ なら p_ℓの指すページを次に探索する．

(4) 索引部の葉のページを探索するときに，$c_\ell < E(y)$ となることはない．

(5) 索引部の1つの葉のページ内の隣接するポインタが同じ値を取ることはあるが，異なる葉のページ内のポインタが同じ値を取ることはない．

索引部のすべての葉のページで条件 (4) を満たすためには，最も右側の葉のページに特別な類名表記を置く必要がある．すなわち，考えられるすべての類名表記よりも大きな値を持つ特別な類名表記をあらかじめファイルに格納しておく．また，キーを持たないグループ $(\#, 0)$ をデータ部に格納しておく．

図4は，ファイルの構成例である．#は特別な類名表記を示す．また，類名表記には次の分類を用いている．

$S_0 = \{\text{a, b, c, d, e, f, g, h, i, j, k, l, m}\}$,
$S_1 = \{\text{n, o, p, q, r, s, t, u, v, w, x, y, z}\}$.

4 キーの検索・挿入・削除

この節ではキーの検索・挿入・削除の方法を示す．

4.1 検索

キー yを検索する方法を以下に示す．

(1) yの類名表記 $E(y)$ を作る．

(2) 索引部で $E(y)$ を探し，$E(y)$ を持つグループがあるページを得る．これは，B^+木の検索法と同様である．$E(y)$ が存在しなければ yも存在しないので終了する．

(3) データ部の $E(y)$ を持つグループ G から yを探す．G は複数のページに分かれていることもあるので，場合によっては複数のページの探索が必要なこともある．

類名表記 c と n 個のキーを持つグループ G があり，G が持つキーの中でページ Pにあるキーを $x_i, x_{i+1}, \cdots, x_j$とする．$P$にある G からキー yを探すには以下のように行う．

1. $i \le \alpha \le j$を満たすαの中で，$x_\alpha = y$となるキー x_αがあれば検索は成功する．
2. $i \le \alpha \le j$であるすべてのαについて $x_\alpha \ne y$ の場合，$y < x_j$または $j = n$なら，キー yはファイル中には存在しない．
3. $y > x_j$かつ $j < n$なら，グループは次のページに続いているので次のページを探索する．

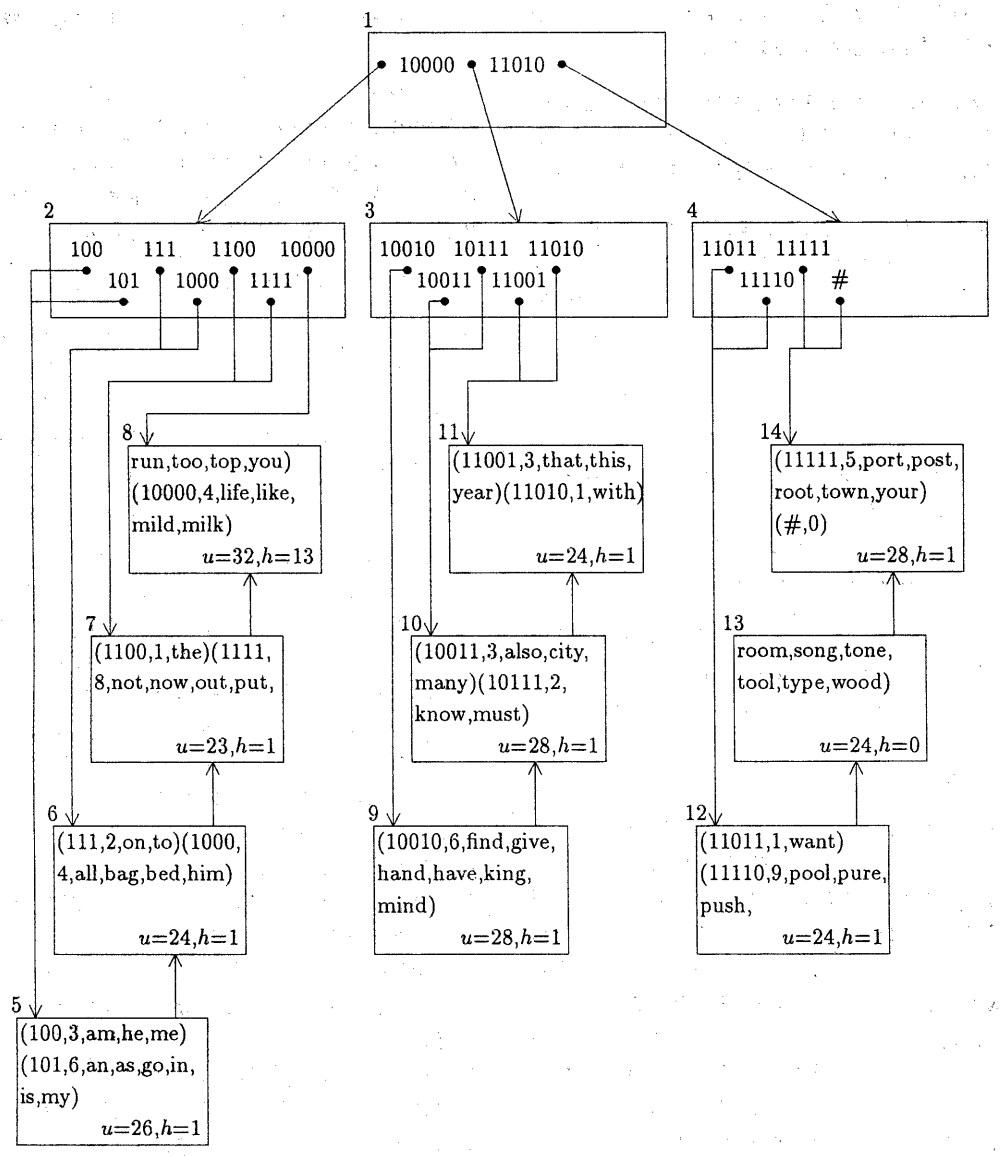

1
10000 11010
2
100 111 1100 10000
101 1000 1111
3
10010 10111 11010
10011 11001
4
11011 11111
11110 #
8
run,too,top,you)
(10000,4,life,like,
mild,milk)
u=32,h=13
11
(11001,3,that,this,
year)(11010,1,with)
u=24,h=1
14
(11111,5,port,post,
root,town,your)
(#,0)
u=28,h=1
7
(1100,1,the)(1111,
8,not,now,out,put,
u=23,h=1
10
(10011,3,also,city,
many)(10111,2,
know,must)
u=28,h=1
13
room,song,tone,
tool,type,wood)
u=24,h=0
6
(111,2,on,to)(1000,
4,all,bag,bed,him)
u=24,h=1
9
(10010,6,find,give,
hand,have,king,
mind)
u=28,h=1
12
(11011,1,want)
(11110,9,pool,pure,
push,
u=24,h=1
5
(100,3,am,he,me)
(101,6,an,as,go,in,
is,my)
u=26,h=1

図 4: ファイル構成例

4.2 誤ったキーによる検索

キー yがファイルに存在しない場合，yが誤っているとすれば yに最もよく似ているキーをファイル中から探すことが望ましい．最もよく似ているキーを探す方法は次のようになる．

(1) $D(E(y),s) \leq d_t$を満たす類名表記 s をすべて発生させ，その集合を Sとする．ここで，d_t はあらかじめ定めた距離のしきい値である．

(2) 集合 Uを既に yとの距離を計算したキーの中で最小距離を持つキーの集合とし，その距離を $d_{\min}$とする．初期値として，$U \leftarrow \phi$, $d_{\min} \leftarrow \max\{p,q,r\} \cdot \mathrm{length}(y)$ を与える．

(3) 集合 Sのすべての要素 s について (4) を行う．

(4) 索引部で sを探す．sが存在するなら yと sを持つグループ内のすべてのキー x 間の距離を計算する．$D(y,x) < d_{\min}$の場合，$U \leftarrow \{x\}$, $d_{\min} \leftarrow D(y,x)$ とし，$D(y,x) = d_{\min}$の場合，$U \leftarrow U + \{x\}$ とする．

(5) Uの要素がただ 1 つなら，その要素が yに最も近いキーである．Uの要素が 2 つ以上あるなら，利用者に Uの要素から選択させる．

4.3 挿入

ファイルに新しいキー yを挿入する場合，まず検索方法を用いて挿入すべき位置を決める．索引部に $E(y)$ が存在する場合は，挿入すべき位置が決まるので，$E(y)$ を持つグループに yを挿入する．グループ $(c,n,x_1,\cdots,x_i,x_{i+1},\cdots,x_n)$ に yを挿入する場合，$x_i < y < x_{i+1}$なら挿入後のグループは $(c,n+1,x_1,\cdots,x_i,y,x_{i+1},\cdots,x_n)$ となる．索引部に $E(y)$ が存在しない場合は，挿入すべき位置がまだ決まっていないので次のように行う．yの検索が終了したページを P(Pは索引部の葉のページ)，Pにある類名表記を $c_1,c_2,\cdots,c_\ell$とする．$E(y) < c_1$ ならポインタ d_1，$c_{i-1} < E(y) < c_i (2 \leq i \leq \ell)$ なら d_iが指すデータ部のページに新しいグループ $(E(y),1,y)$ を挿入する．更に，索引部のページ Pに $E(y)$ と新しいグループが挿入されたページを指すポインタを挿入すればよい．

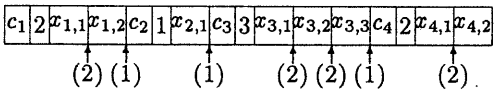


図 5: データ部のページの分割可能な位置

キーやグループを挿入したことによりページの使用領域サイズがその最大値 $U_{\max}$を超えた場合，そのページを分割しなければならない．データ部のページを分割する方法を次に示す．最初にデータ部のページの分割可能な位置を定義する．

SP1: グループの境界位置．

SP2: グループが n 個 $(n \geq 2)$ のキーを持つ場合，i 番目のキーと $i+1$ 番目のキーの間．ただし，$1 \leq i \leq n$．グループが 1 個しかキーを持たない場合は，グループ内に分割可能な位置はない．

例えば，図5に示す使用領域を持つページでは，矢印で示す位置が分割可能な位置である．(1) は SP1 に，(2) は SP2 に属する分割点であることを示す．

実際のページの分割点は，ページの使用領域の中央 Mからσ_kの範囲内にある分割可能な位置から選ぶ．分割可能な位置が複数ある場合，SP1 に属する最も Mに近い分割点を選ぶ．SP1 に属する分割点がない場合，SP2 の中の最も右側の分割点を選ぶ．これは，グループが 2 つのページに分けられた場合，2 つ目のページにあるそのグループのキーを検索するには，データ部のページを 2 つ読み込む必要があるので，最初のページにできる限り多くキーを持たせるためである．

図6は，データ部のページ Pが 2 つのページ Pと P'に分割される例である．Mからσ_kの範囲には分割可能な位置は 2 つあるが，そのうち SP1 に属する分割点 ($x_{2,2}$と c_3の間) が選ばれる．また，データ部のページが分割されると，索引部の葉のページのポインタを正しいデータ部のページを指すように更新する必要がある．

既にいっぱいである索引部の葉のページに新たな類名表記を挿入する場合，ページの分割が必要となる．索引部は B^+木として構成するので，ページの分割も基本的には B^+木と同様であるが，連結リストに対する操作も必要である．

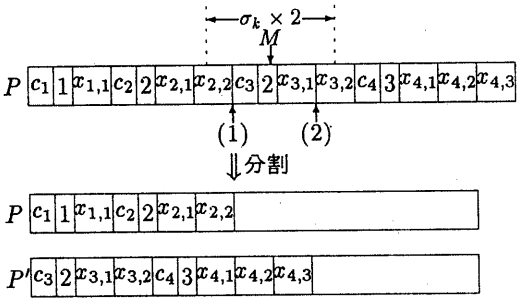


図 6: データ部のページの分割の例

索引部の葉のページの連続する 2 つのエントリを (c_l, d_l), (c_r, d_r) とし，この間に分割点があるとする．ページの分割は，ページのちょうど中央 M ではなく，M から σ_i の範囲内で行う．索引部のページの最適な分割点は次のように決める．$d_l \neq d_r$ となる分割点の中で最も M に近いものを最適な分割点とする．$d_l \neq d_r$ となる分割点がない場合は，最も右側の分割点を最適な分割点とする．これは，分割後の d_l が指すページは，後で述べるように連結やアンダーフローの処理を行えないので，d_l の指すページにできる限り多くのグループを格納させるためである．

最適な分割点を決めた後にそのページを分割する．そして，分割された 2 つのページを分岐する類名表記を 1 つ上のレベルに置くため，分割後の左のページの中の最も右側の類名表記を 1 つ上のレベルのページに挿入する．葉より上のレベルへの挿入は B^+ 木の場合とまったく同じである．

索引部のページが 2 つに分割されたので，そのページを親としていたデータ部のページからなる連結リストも 2 つに分ける必要がある．c_l, c_r を持つグループをそれぞれ G_l, G_r とする．連結リストの分割方法は，以下の 2 つの場合に分類できる．

(1) G_l と G_r が同じページにない場合．

(2) G_l と G_r が同じページにある場合．

場合 (1) は，図 7(a) のような場合であり，必ず $d_l \neq d_r$ となる．この場合は簡単であり，G_l を持つページと G_r を持つページ間のポインタを削除するだけでよい．

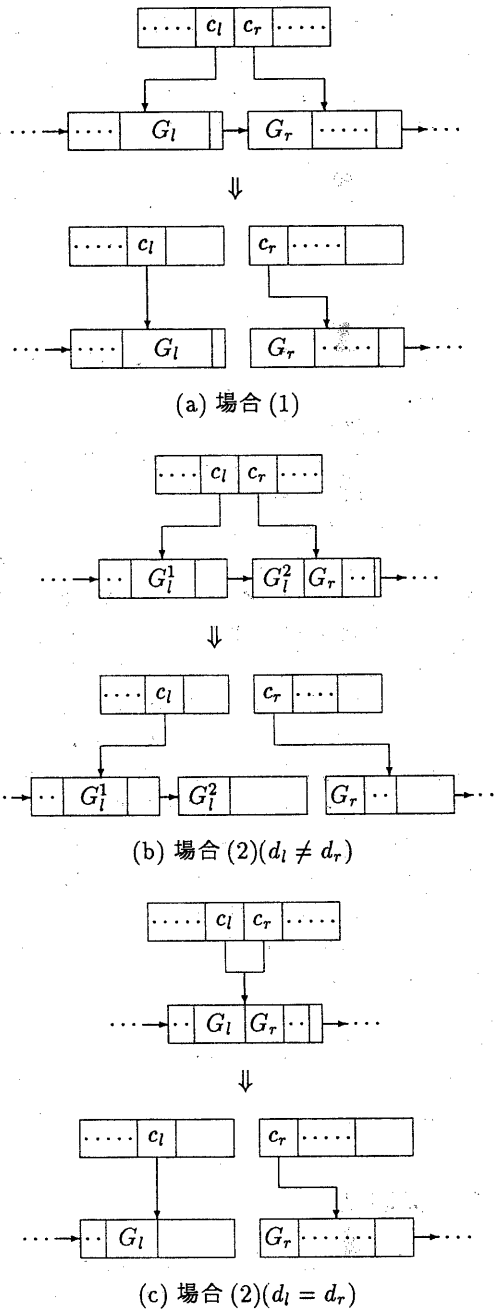


図 7: 連結リストの分割の例

場合 (2) では，$d_l \neq d_r$となる場合と $d_l = d_r$となる場合があるが，どちらの場合も同じ処理を行う．つまり，d_rの指すページをG_lとG_rの間で分割する．そして必要ならば，分割後の d_rが指すページで後で述べるような連結やアンダーフローの処理を行う．図 7(b) と (c) は，場合 (2) の例であり，(b) は $d_l \neq d_r$となる場合，(c) は $d_l = d_r$となる場合である．

4.4 削除

ファイルにあるキーを削除するには，キーを検索した後に見つかったキーを削除すればよい．このとき，削除すべきキーを持つグループが 1 つのキー (削除すべきキー) しか持たない場合，そのグループを削除し，索引部から削除したグループの類名表記を削除すればよい．

キーやグループが削除され，ページがアンダーフロー状態になれば，そのページは連結やアンダーフローの処理が必要である．ただし，連結リストの最後の要素であるページがアンダーフロー状態のときは，1 つ手前の要素のページを簡単には知ることができないのでこのような処理は行わない．(双方向リストなら簡単に手前のページが分かる．) アンダーフロー状態のページ Pと隣のページ P'の使用領域の合計サイズが $U_{\max}$以下なら，その 2 つのページを連結できる．ページの連結は，P'のページの使用領域のすべてのグループを Pのページに移動し，索引部の葉のページのポインタを正しいデータ部のページを指すように更新すればよい．アンダーフローの処理は，2 つのページを連結した後に再び分割すればよい．

索引部のページの連結やアンダーフローの処理は，B^+木の場合と同じである．

5 実験結果

B 木の評価には，検索や更新の操作に必要な 2 次記憶へのアクセス回数や記憶利用率が用いられる．ここでも，提案するファイルについてこの 2 点を調べる．また，誤ったキーの検索では，検索率も合わせて調べる．

実験で使用したキー集合には，UNIX の OS 下

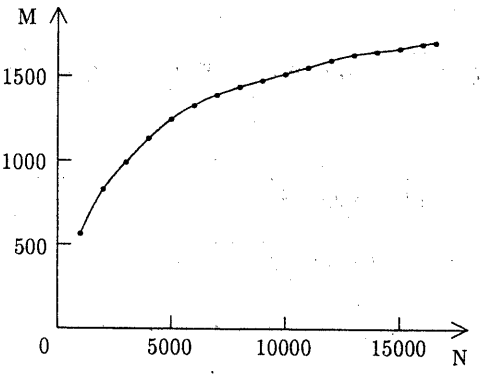


図 8: キーの数と類名表記の数の関係

にある約 2.5 万語の英単語辞書のうち長さ 6~10 の単語 16,561 語を用いた．また，以下の分類を用いてファイルを構成した．

$$S_0 = \{a, b, c, d, e, f, g, h, i, j, k, l, m\},$$
$$S_1 = \{n, o, p, q, r, s, t, u, v, w, x, y, z\}.$$

ファイルを構成するためのパラメータとして，ページサイズを 1,024byte，$k = 169$，$\sigma_i = 5$，$\sigma_k = 50$byte とした．

実験の手順を以下に示す．16,561 個のキー集合からランダムにキーを選び出し，順にファイルに挿入する．1,000 個のキーが挿入されるごとに，その処理の間に行われた 2 次記憶へのアクセスの回数を調べ，1 つのキーあたりの平均値を求め，それを挿入コストとする．また，その時点でファイルに存在するキーをすべて検索することにより検索コストを得る．同時に記憶利用率も調べる．以下同様にすべてのキーを挿入し終わるまで 1,000 個のキーを単位に処理を続ける．削除の場合も同様に 1,000 個のキーを単位に削除コストを調べる．

キーの数 Nと類名表記の数 Mの関係を図 8に示す．また，検索・挿入・削除のそれぞれのコストを図 9，10，11に示す．R，W はそれぞれページの読み込み回数，書き込み回数を示す．索引部の木の高さは，キーの数が 205 までは 1 であるが，206 以上 16,651 までは常に 2 であり，そのため検索コストは約 3 回 (索引部で 2 回，データ部で 1 回) となった．N が 7,000 のあたりから検索コストが 3 より若干多くなった．これは，複数のページに分

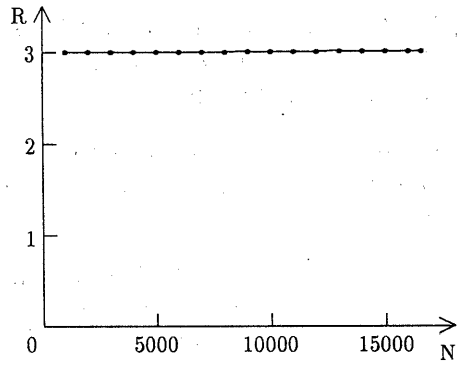


図 9: 検索コスト

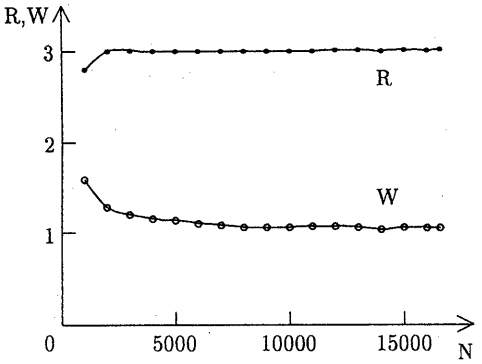


図 10: 挿入コスト

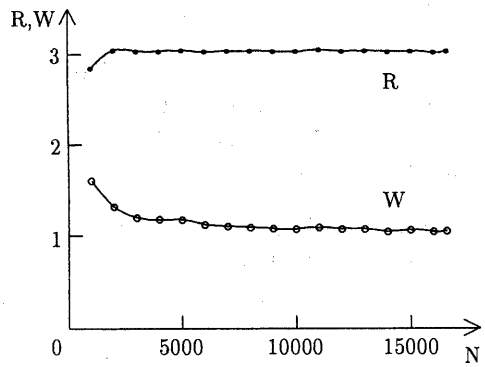


図 11: 削除コスト

表 1: 誤ったキーによる検索
N_{key} : 距離を計算したキーの数

長さ	誤りの種類	$\|S\|$	N_{key}	R	検索率 (%)
7	置換	21.1	681	22.0	91.7
8	置換	23.3	376	24.7	97.0
9	置換	23.5	179	25.2	97.0
6	挿入	20.9	675	22.0	95.7
7	挿入	23.3	367	24.6	98.2
8	挿入	23.4	176	25.2	99.3
8	脱落	21.0	681	22.0	84.9
9	脱落	23.4	376	24.6	90.8
10	脱落	23.8	179	25.4	94.3

割されたグループが現れたために，データ部での読み込み回数が増加したからである．挿入・削除のそれぞれの 2 次記憶への書き込み回数が，キーの数が少ないときに多くなっている．これは，図 8からも分かるように，キーの数が少ないときは類名表記数の増減が激しいため，索引部のページへの書き込みが頻繁に起こるからである．

表 1に誤ったキーによる検索結果を示す．誤ったキーは以下のようにして作った．キー集合からランダムにキーを選び，1 つのキーにつき 1 つの誤りをランダムに発生させる．誤りの種類は，置換・挿入・脱落とし，それぞれの誤りについて誤ったキーを発生させる．このとき，誤ったキーがキー集合にある他のキーに変化したものは除いておく．誤りの種類，および，キーの長さごとにそれぞれ 1,000 個のキーを作る．これらのキーを 16,651 個のキーが挿入されたファイルから検索して表 1の結果を得た．誤ったキーによる検索では，読み込んだページを主記憶上に留めておけば，1 つのキーを検索するために，同じページを 2 度以上読み込むことはない．ページの読み込み回数は，正しいキーを検索する場合と比較すると，約 7.3~8.5 倍であることが分かる．

最後に，ファイルの記憶利用率 SU を図 12に示す．記憶利用率はキーの数に関係なく約 70%になった．なお，記憶利用率は，以下の式を用いて計算

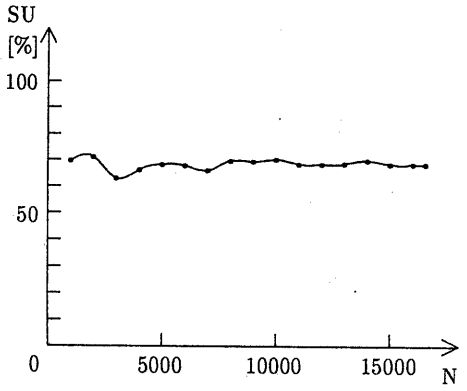


図 12: 記憶利用率

を行った．

$$SU = \frac{\text{全ページの使用領域サイズ}}{\text{ページサイズ} \times \text{ページ数}}$$

6 むすび

類名表記を用いた階層ファイルの構成にB木を用いることにより，キーの挿入・削除が容易に行えるファイルの構成法および操作法を述べた．実験による結果では，キーの検索や更新に必要なコストは，ほぼ木の高さと同じになった．また，記憶利用率はキーの数に関係なく約70%になった．

今後の課題としては，提案したファイルが，より多くのキーを用いた場合でも有用であることを確かめること，また，ファイル構成を再検討し，一部複雑な挿入処理をより簡単にすることなどである．

参考文献

[1] R. Bayer and E. McCreight, "Organization and Maintenance of Large Ordered Indexes", *Acta Inf.*, vol.1, pp.173-189, 1972.

[2] H. Wedekind, "On the Selection of Access Paths in a Data Base System", *Proc. IFIP Working Conf. on Data Base Management*, pp.385-397, 1974.

[3] R. Bayer and K. Unterauer, "Prefix B-Trees", *ACM Trans. on Database Systems*, vol.2, no.1, pp.11-26, 1977.

[4] D. Comer, "The Ubiquitous B-Tree", *Computing Surveys*, vol.11, no.2, pp.121-137, 1979.

[5] D. J. Taylor and D. E. Morgan and J. P. Black, "Redundancy in Data Structures: Improving Software Fault Tolerance", *IEEE Trans. Software Eng.*, vol.6, no.6, pp.585-594, 1980.

[6] D. J. Taylor and D. E. Morgan and J. P. Black, "Redundancy in Data Structures: Some Theoretical Results", *IEEE Trans. Software Eng.*, vol.6, no.6, pp.595-602, 1980.

[7] K. Kant and A. Ravichandran, "Synthesizing Robust Data Structures – An Introduction", *IEEE Trans. Comp.*, vol.39, no.2, pp.161-173, 1990.

[8] S. N. Srihari, "Computer Text Recognition and Error Correction", *IEEE Computer Society Press*, 1985.

[9] E. Tanaka and T. Toyama and S. Kawai, "High Speed Error Correction of Phoneme Sequences", *Pattern Recognition*, vol.19, no.5, pp.407-412, 1986.

[10] E. Tanaka and Y. Kojima, "A High Speed String Correction Method Using Hierarchical File", *IEEE Trans. PAMI*, vol.9, no.6, pp.806-815, 1987.

[11] 沼倉, 田中, 青木, 矢野目, 矢吹, "誤ったキーでも検索できる情報検索システム", 情報処理学会論文誌, vol.30, no.11, pp.1468-1478, 1989.