データベースシステム 104－1
（1995．7．18）

ビットスライス方式に基づく分割シグネチャファイル構成法の提案と評価

渡辺 悟康 * 北川 博之 †

* 筑波大学 工学研究科
† 筑波大学 電子・情報工学系

概要

集合は複雑なデータ構造を支援するデータベース中において、頻繁に現れる基本的なデータ構造である。そのため、複雑なデータを扱う先進的なデータベースシステムでは、集合値を効率良く支援する索引機構が必要となる。我々は、従来テキスト検索に用いられてきたシグネチャファイルを集合値検索機構として取り上げ、検索コストなど各種のコストの評価を行なってきた。

本論文では、シグネチャファイルの検索コストの低減を目的とし、分割シグネチャファイルとビットスライストシグネチャファイルの考え方を組み合わせた、新しいシグネチャファイル構成法である *Partitioned Bit-Sliced Signature File* (P-BSSF) と *Bit-Sliced Quick Filter* (BSQF) の提案を行なう。また、提案する方法の検索コスト、更新コスト及び格納コストについての見積りを行ない、その有効性を検討する。

Design and Evaluation of Partitioned Bit-Sliced Signature Files

Noriyasu Watanabe* Hiroyuki Kitagawa†

*Doctoral Degree Program in Engineering, Univ. of Tsukuba
†Institute of Information Sciences and Electronics, Univ. of Tsukuba

abstract

Sets are primitive data objects and often appear in advanced databases which support complex data structures. Therefore, it is necessary for the advanced database system to have access facilities which support set-valued object retrieval efficiently. We have proposed the use of signature files as set-valued retrieval facilities and evaluated retrieval, update and storage costs.

In this paper, we propose new partitioned signature file organizations, named *Partitioned Bit-Sliced Signature File* (P-BSSF) and *Bit-Sliced Quick Filter* (BSQF), based on the bit-sliced scheme to decrease the retrieval cost. We estimate retrieval, update and storage costs for the proposed organizations, and evaluate their effectiveness.

1 はじめに

近年データベース適用分野の拡大に伴い、単純なレコードだけでなく階層構造などを持つような複雑な対象をデータベース化する要求が高まっている。その要求に応えるために、入れ子型リレーショナルモデルや、オブジェクト指向モデル等の研究が進められてきた。そのような複雑な構造の基本構成子としては、しばしば集合が含まれる。したがって、複雑な対象構造を扱う先進的なデータベースシステムでは、集合データ固有の演算を効率良く扱うことが要求される。これまで、入れ子型索引等、階層構造を意識した検索機構がいくつか提案されているが、それらは様々な集合値検索を意図したものではない。

我々は、集合値に対する索引機構として従来主にテキスト検索を対象としてきたシグネチャファイル[1]の適用を考え、研究を行なってきた。

シグネチャファイルの物理的構成法としては、*Sequential Signature File*(SSF)、*Bit-Sliced Signature File*(BSSF)、*Frame-Sliced Signature File*(FSSF)[2]、*Quick Filter*[3]等がこれまでに提案されている。我々はそれらの内、ある種の条件を満たすBSSFが集合値検索において特に有効であることを示した[4]。

本論文では検索対象オブジェクト数の多い大規模データベースにおいて、シグネチャファイルによる検索コストの低減を目的とし、分割シグネチャファイルの概念とビットスライス方式の概念を組合わせた、シグネチャファイル構成法を提案する。そして、集合値検索を対象として提案するシグネチャ構成法による検索コスト、更新コスト、及び格納コストについて見積りを行ない、その有効性を検討する。

本研究に関連する研究としては、extendible hashingとFSSFを組み合わせた、*Two-dimensional Dynamic Signature File*(TDSF)[5]の提案がある。しかし、これは集合値検索を意図したものではなく、またシグネチャの構成法に制限があるため集合値検索の効率化には不十分であると思われる。

本論文の構成は以下のとおりである。2章では、シグネチャファイルの手法の集合値検索への応用と、これまでに提案されているシグネチャファイル構成法の内、代表的なものについて述べる。3章では、本論文で提案する*Partitioned Bit-Sliced Signature File*(P-BSSF)の構成と各処理について述べる。4章では、性能評価のためのコストモデルについて述べる。5章では、検索コストの評価を行ない、6章ではP-BSSFを改良した*Bit-Sliced Quick Filter*(BSQF)の提案を行なう。7章で更新コスト、8章で格納コストの面からBSSF、P-BSSF、BSQFの比較評価を行なう。9章は本論文のまとめである。

2 シグネチャファイルの概要

2.1 シグネチャファイルによる集合値検索

シグネチャ(*signature*)とは、個々のデータオブジェクトから作成される固定長のビット列のことである。本研究で対象とする集合値検索のためのシグネチャの作成法は以下による。

1. 集合データオブジェクトの各要素オブジェクトから、長さがFビットで、その内mビットが"1"にセットされた**要素シグネチャ**(*element signature*)を作成する。またこの時mをウエイトと呼ぶ。
2. すべての要素シグネチャのビットごとの論理和をとる**スーパーインポーズドコーディング**(*superimposed coding*)を行ない、**集合シグネチャ**(*set signature*)を作成する。

このようにして作成されたシグネチャと、各集合データオブジェクトの識別子(OID)の組を格納したものを**シグネチャファイル**(*signature file*)とする。図1に集合{"DBMS", "OS", "画像処理"}から集合シグネチャが作成される例を示す。

集合の要素		要素シグネチャ
"DBMS"	⟶	00010001
"OS"	⟶	01001000
"画像処理"	⟶	00000101
		⇓ 論理和
集合シグネチャ	⟶	01011101

図 1: 集合シグネチャの作成

問合せが与えられた際に、問合せ条件中に現れる集合を**問合せ集合**(*query set*, Q)、データベース中の検索対象の集合を**ターゲット集合**(*target set*, T)と呼ぶ。また、それぞれから作成される集合シグネチャを、**問合せシグネチャ**(*query signature*, S_Q)、**ターゲットシグネチャ**(*target signature*, S_T)と呼ぶ。

集合値検索の問合せ条件には様々なものが考えられるが、本論文では、TがQを包含する場合($T \supseteq Q$)と、逆にQがTを包含する場合($T \subseteq Q$)を対象とする。それぞれの問合せ条件について、以下の条件を満たすオブジェクトが問合せ条件を満たす候補となる。

$T \supseteq Q$: $S_Q \wedge S_T \equiv S_Q$

$T \subseteq Q$: $S_Q \wedge S_T \equiv S_T$

ここで、"∧"はビット列の論理積を、"≡"はビット列が等しいことを示している。

しかし、これらの候補の中には、実際に問合せ条件を満たす**アクチュアルドロップ**(*actual drop*)と、実際には条件を満たさない**フォルスドロップ**(*false drop*)があるので、該当するオブジェクト自身をを取り出してその判定を行なう必要がある。この操作を**フォ**

ルスドロップレゾリューション (*false drop resolution*) と呼ぶ。図 2に $T \supseteq Q$ に対する問合せ処理例を示す。

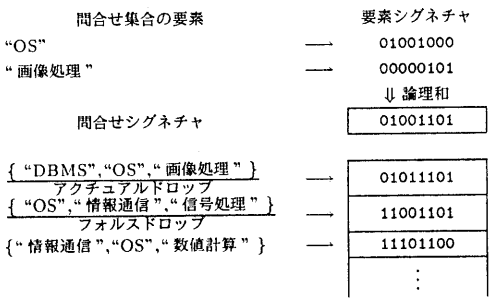

図 2: 問合せ処理 ($T \supseteq Q$)

2.2 検索効率を考慮したシグネチャファイル構成法

もっとも単純なシグネチャファイル構成法である SSF では、シグネチャファイル全体をスキャンする必要があり、そのコストが検索コストに大きな影響を与える。そのため検索効率を考慮したシグネチャファイル構成法がこれまでにも提案されている。ここでは *Bit-Sliced Signature File*(BSSF) と分割シグネチャファイル構成法の一つである Fixed Prefix 法 [6] の二種類について述べる。

2.2.1 BSSF

BSSF では、集合シグネチャを 1 ビットずつ**ビットスライスファイル** (*bit-slice file*) と呼ばれるファイルに格納する。検索は問合せ条件によって次のように処理される。

1. 与えられた問合せから問合せシグネチャを作成する。
2. $T \supseteq Q$: 問合せシグネチャで "1" の立っているビット位置に対応するビットスライスファイルのみをスキャンする。

   $T \subseteq Q$: 問合せシグネチャで "0" になっているビット位置に対応するビットスライスファイルのみをスキャンする。
3. フォルスドロップレゾリューションを行なう。

長所は、問合せシグネチャのウエイトによっては、検索時のコストがかなり低減できることがあげられる。問題点としては、一般に SSF と比較して更新コストが大きくなってしまうことがあげられる。

2.2.2 Fixed Prefix 法

Fixed Prefix 法は、シグネチャの先頭数ビットをキーとして、シグネチャを複数のパーティションに分割格納する、分割シグネチャファイル構成法の一手法である。検索の時は、検索対象のパーティションの絞り込みが可能なためすべてのパーティションをスキャンする必要がないので、検索処理が高速になる。しかし問合せ集合の要素数が、$T \supseteq Q$ では小さいと、$T \subseteq Q$ では大きいとパーティションの絞り込みが十分できず、ほぼすべてのシグネチャをスキャンする必要が生じる。

3 Partitioned Bit-Sliced Signature File (P-BSSF)

以下では、前述の BSSF と Fixed Prefix 法を組合せた、*Partitioned Bit-Sliced Signature File* (P-BSSF) の提案を行ない、その評価を行なう。

P-BSSF の構成を図 3に示す。

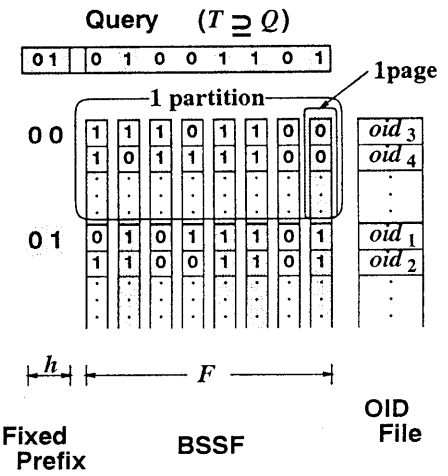

図 3: P-BSSF の構成

BSSF 部の各ビットスライスファイルの第 i 物理ページからなる F ページの集まりを**パーティション**と呼ぶ。P-BSSF はあらかじめ決めた $n(=2^h)$ 個のパーティションからなる。各オブジェクトのターゲットシグネチャをどのパーティションに格納するかは以下に述べる。

3.1 P-BSSF の作成法

P-BSSF は次のように作成される。

1. 長さ f ビット、ウエイトをターゲットシグネチャ中の "0" と "1" の生じる確率がそれぞれ 0.5 となる $m_{\mathrm{opt}}(= \frac{f}{D_t} \ln 2)$ ビットとして、ターゲットシグネチャを作成し、その先頭 h ビットのビッ

トパターン (Prefix, $S_{T\{FP\}}$) によりシグネチャを格納するパーティションを決定する。ここでウエイトを m_{opt} とするのは、各パーティションへの割り当ての均衡を図るためである。

2. 長さ F ビット、ウエイト m ビットでターゲットシグネチャ $S_{T\{BSSF\}}$ を作成し、対応する BSSF 部に格納する。集合値検索の検索効率の向上を図るためには、$m = 2 \sim 3$ とするのが有効である [4]。

3. 対応する OID を OID ファイルに格納する。

一つの集合値に対して Fixed Prefix 部用と BSSF 部用の二種類のターゲットシグネチャを作成するのが一つの特徴である。

3.2 検索処理

P-BSSF における検索処理として通常の検索手法と、BSSF における検索効率化の手法である**スマート検索** (*smart retrieval*)[4] を応用した検索処理手法について考察を行なう。

3.2.1 通常の検索処理

通常の検索処理では、問合せ集合が与えられると次の処理を行なう。

1. ターゲットシグネチャと同様に、BSSF 部用シグネチャ ($S_{Q\{BSSF\}}$) と Prefix($S_{Q\{FP\}}$) 用のシグネチャの二種類の問合せシグネチャを作成する。

2. $S_{Q\{FP\}}$ を用いて、パーティションの絞り込みを行なう。次の条件を満たすパーティションが検索対象となる。

$$T \supseteq Q : S_{T\{FP\}} \wedge S_{Q\{FP\}} \equiv S_{Q\{FP\}}$$

$$T \subseteq Q : S_{T\{FP\}} \wedge S_{Q\{FP\}} \equiv S_{T\{FP\}}$$

3. 検索しなければならないパーティション中の BSSF 部のターゲットシグネチャと、BSSF 部用の問合せシグネチャとのマッチングを行ない、問合せ条件を満たす候補を取り出す。その時に候補となるシグネチャ $S_{T\{BSSF\}}$ は、問合せ条件によって次のようになる。

$$T \supseteq Q : S_{T\{BSSF\}} \wedge S_{Q\{BSSF\}} \equiv S_{Q\{BSSF\}}$$

$$T \subseteq Q : S_{T\{BSSF\}} \wedge S_{Q\{BSSF\}} \equiv S_{Q\{BSSF\}}$$

4. フォルスドロップレゾリューションを行ない、アクチュアルドロップを返す。

3.2.2 スマート検索による検索処理

スマート検索は、[4] において提案した BSSF における検索効率化の一手法である。

シグネチャファイルでは、候補となったオブジェクトが問合せ条件を満たすかどうかという、最終的な判定はフォルスドロップレゾリューションの段階で行なわれる。よって、フォルスドロップが多少増加しても、スキャンするビットスライスファイルの数を減らした方が、検索コスト全体からみれば有利になることがある。この性質を利用した手法が、スマート検索方式である。

P-BSSF の中には BSSF 部が存在するので、P-BSSF でも同様にスマート検索を行なうことが可能である。詳しい処理については後述する。

3.3 更新処理

更新処理として、挿入処理、削除処理について述べる。その際、更新するオブジェクトのターゲットシグネチャ、$S_{T\{BSSF\}}$ 及び Prefix $S_{T\{FP\}}$ と OID が分かっているものとする。また、パーティションのオーバーフローは生じないものとする。

3.3.1 挿入処理

P-BSSF に新しいターゲットシグネチャを挿入する場合、以下の手順で処理を行なう。

1. Prefix $S_{T\{FP\}}$ により、挿入すべきパーティションを決定する。

2. そのパーティションの OID ファイルをスキャンし、削除ビット[1]が立っている場所を見つけ、そこにオブジェクトの OID を挿入する。

3. 対応する削除ビットをリセットする。

4. $S_{T\{BSSF\}}$ で "1" の立っている位置について、パーティション中の BSSF 部に "1" を立てる。

3.3.2 削除処理

P-BSSF からオブジェクトを削除する場合、以下の手順で処理を行なう。

1. Prefix $S_{T\{FP\}}$ により、そのシグネチャが含まれているパーティションを決定する。

2. そのパーティション中の対応する OID ファイルをスキャンし該当する OID があれば削除ビットを立てる。

3. BSSF 部の立っているビットを "0" にリセットする。

[1] OID ファイル中にあるビットで、その位置に有効なオブジェクトが存在するかしないかの情報を持っている。

4 コストモデル

P-BSSF の検索コスト RC、更新コスト UC、格納コスト SC のコストモデルを作成する。検索コストと更新コストは物理ページの I/O の回数を、格納コストは格納するのに必要な物理ページ数をコストとする。コストモデルの作成の際に使う記号の定義を表 1に示す。

記号	定義
N	オブジェクトの総数
P	1 ページのバイト数 (= 4096)
b	1 バイトのビット数 (= 8)
oid	OID のバイト長 (= 8)
D_t	ターゲット集合の要素数
D_q	問合せ集合の要素数
pl	各パーティションの充足率
F, f	シグネチャのビット長
m	BSSF 部の要素シグネチャのウエイト (= 2)
m_{opt}	Fixed Prefix 部の要素シグネチャのウエイト $(= \frac{f}{D_t} \ln 2)$
h	Fixed Prefix 部のキーのビット長
n	Fixed Prefix 部によって分割されるパーティションの数 $(2^{h-1} < n \leq 2^h)$
A	アクチュアルドロップ数
Fd	フォルスドロップ確率
P_o	1 オブジェクトを取り出すためのページアクセス数 (= 1)

表 1: 記号の定義

コストモデルを作成する際、以下を仮定する。

- D_t はすべてのオブジェクトで一定
- データベース中には各集合値は均等に出現

4.1 検索コストモデル

総検索コスト RC は、Prefix で絞り込まれたパーティション中の BSSF 部の検索コスト、OID ファイルにアクセスするコスト (LC_{OID})、フォルスドロップレゾリューションを行なうコストとの和になり、(1) 式で表される。

$$RC = n \cdot PAR \cdot LC_{BSSF} + LC_{OID} + P_o(A + PAR \cdot Fd(N - A)) \quad (1)$$

ここで、PAR は検索する必要のあるパーティションの割合 (*Partition Activation Ratio*) を表し、m_q を問合せシグネチャのウエイトとすると、(2) 式 [7] 及び (3) 式で表される。

$$PAR_{\{T \supseteq Q\}} \approx \left(1 - \frac{m_q}{2f}\right)^h \quad (2)$$

$$PAR_{\{T \subseteq Q\}} \approx \left(\frac{f + m_q}{2f}\right)^h \quad (3)$$

また、LC_{BSSF} は 1 パーティション中の BSSF の検索コストを表し、次の (4) 式及び (5) 式で表される。

$$LC_{BSSF\{T \supseteq Q\}} = m_q \quad (4)$$

$$LC_{BSSF\{T \subseteq Q\}} = F - m_q \quad (5)$$

LC_{OID} は、SC_{OID} を OID ファイルの格納総ページ数とすると、Yao の関数 [8] を用いて (6) 式で表される。

$$LC_{OID} = SC_{OID} \cdot \left[1 - \prod_{i=1}^{A+Fd(N-A)} \frac{N \cdot (1 - 1/SC_{OID}) - i + 1}{N - i + 1}\right] \quad (6)$$

Fd はフォルスドロップの生じる確率で、(7) 式及び (8) 式で表される [4]。

$$Fd_{\{T \supseteq Q\}} \approx (1 - e^{-\frac{m}{F} D_t})^{m D_q} \quad (7)$$

$$Fd_{\{T \subseteq Q\}} \approx (1 - e^{-\frac{m}{F} D_q})^{m D_t} \quad (8)$$

4.2 更新コストモデル

更新は挿入、削除ともにまず Prefix を用いて、更新するパーティションを決定する。次に、そのパーティションに対応する OID ファイルを探索し、最後に BSSF 部にビットをセット、またはリセットする。よって、更新コストはターゲットシグネチャのウエイトを m_t とすると、(9) 式で表される。

$$UC = \frac{b \cdot oid}{2} + m_t \quad (9)$$

4.3 格納コストモデル

総格納コスト SC は、BSSF 部の格納コスト SC_{BSSF} と OID ファイルの格納コスト SC_{OID} の和になり、(10) 式で表される。

$$SC = SC_{BSSF} + SC_{OID} \quad (10)$$

$$SC_{BSSF} = F \left\lceil \frac{N}{Pb \cdot pl} \right\rceil$$

$$SC_{OID} = \left\lceil \frac{N}{\lceil \frac{P}{oid} \rceil \cdot pl} \right\rceil$$

5 検索コストの解析

本章では、P-BSSF と BSSF の検索コストを比較する。ここでは $D_t = 100, F = 1024$ として解析を行なう。検索コストに影響を与える要因は数多くあるが、パーティションの充足率はコストに影響を与えると考えられる。通常、P-BSSF は Fixed Prefix 法を用いているためパーティションの充足率は 100% にはならない。そこで、充足率が 100%,80%,60% の

場合を検討する。充足率 pl によって、オブジェクトの総数 N が変化するので、検索コスト RC を N で割った値 RC/N を比較する。P-BSSF における充足率とオブジェクトの総数の関係を表 2に示す。また、比較する BSSF のオブジェクトの総数は 800,000 とする。

充足率 (pl)	オブジェクトの総数 (N)
60%	629,146
80%	838,861
100%	1,048,576

表 2: 充足率とオブジェクトの総数

5.1 通常の検索処理によるコスト

通常の検索処理における検索コストを、図 4,5に示す。

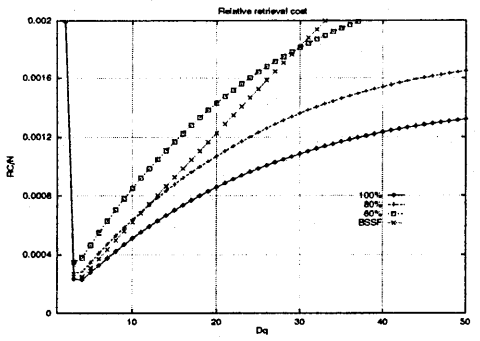

図 4: $T \supseteq Q$ の検索コスト

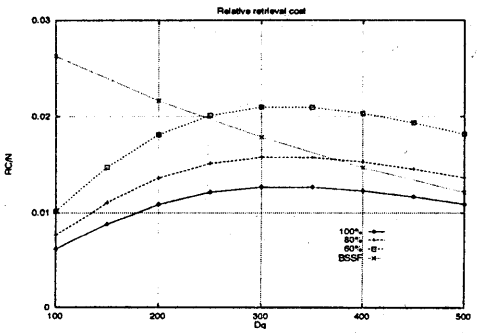

図 5: $T \subseteq Q$ の検索コスト

図 4,5より、充足率が検索コストに影響を与えることが分かる。充足率 100% の場合が最も効率良く、充足率の低下に伴い検索コストは増加する。充足率が低下した場合には、BSSF の方が検索コストで有利である領域が生じることが分かる。しかし、$T \supseteq Q$ では D_q が大きい場合、$T \subseteq Q$ では D_q が小さい場合、BSSF と比較して P-BSSF が有効であることが分かる。

5.2 スマート検索処理によるコスト

図 4より、BSSF の検索コストは $D_q = 4$ の時に最小となることが分かる。この場合の検索コストは、$D_q \leq 4$ ではフォルスドロップレゾリューションのコスト、$D_q > 4$ ではシグネチャをスキャンするコストが中心となる。特に $D_q > 4$ では、フォルスドロップはほとんど発生せず、フォルスドロップレゾリューションのコストは無視できる。したがって、$D_q > 4$ でのコストの増加は、無駄なシグネチャのスキャンを削減することで低減できる。

この場合、スマート検索の処理は以下のようになる。

1. 実際の問合せが出された時 $D_q \leq 4$ であれば、通常通りの検索を行なう。
2. $D_q > 4$ ならば、問合せ集合中の任意の 4 つの要素を選んで、問合せシグネチャを作成し、検索を行なう。

このスマート検索を P-BSSF に応用すると次のようになる。

1. $D_q \leq 4$ の時は通常通りの検索を行なう。
2. $D_q > 4$ の時は、全要素を用いた問合せシグネチャから Prefix を作成し、パーティションを決定する。続けて任意の 4 つの要素を選んで BSSF 部のシグネチャを作成し、検索を行なう。

スマート検索方式を適用した時の検索コストを、図 6,7に示す。

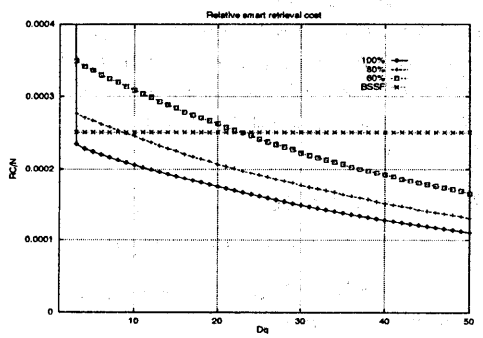

図 6: $T \supseteq Q$ の検索コスト

スマート検索によるコストは、図 6,7のように BSSF で一定になっている範囲についても、P-BSSF では

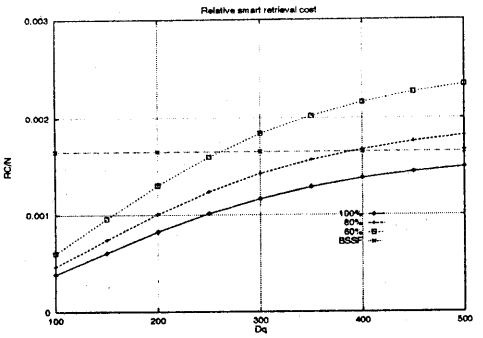

図 7: $T \subseteq Q$ の検索コスト

減少を続けており、$T \supseteq Q$ では D_q が大きいほど、$T \subseteq Q$ では D_q が小さいほど P-BSSF の方が有利になることが分かる。しかし通常の検索同様、充足率が影響を与え、充足率が低くなるとコストが悪化する領域が生じる。

6 Bit-Sliced Quick Filter (BSQF)

前章で述べたように P-BSSF では、充足率が検索コストに影響を与える。したがってオブジェクトの追加や削除が比較的多い動的な環境ではこの点が問題となる。そこでこのような場合でも、ある程度の充足率を保てるように、Fixed Prefix 法の代わりに linear hashing[9] を用いた Quick Filter の方式と組み合わせた、*Bit-Sliced Quick Filter* (BSQF) を提案する。linear hashing では、パーティションの充足率をコントロールする *load control* の手法 [9] が適用可能である。load control によって、充足率 75% とした時はオーバーフローが 5% 起こることが報告されている。

linear hashing では、パーティションの分割、結合が動的に生じる。BSQF ではこの際、各シグネチャのパーティションを決定するために用いる Prefix の再計算を避けるため、あらかじめ h_{MAX} ビットの Prefix 用のターゲットシグネチャを格納しておくことにする。

図 8に BSQF の構成を示す。

検索処理、更新処理の方法は、更新時にパーティションの分割、結合が生じる可能性があることを除いては、ほぼ P-BSSF と同様である。図 9,10に BSQF と BSSF のスマート検索による検索コストを示す。($D_t = 100, F = 1024, N = 800,000, pl \approx 75\%$)

7 更新コスト

これまでに述べた各シグネチャファイル構成法の更新コストを表 3に示す。ただし、BSQF については はパーティションの分割、結合が生じる確率は非常に低いためそのコストは考慮していない。

P-BSSF、BSQF の方が BSSF に比べかなり低い

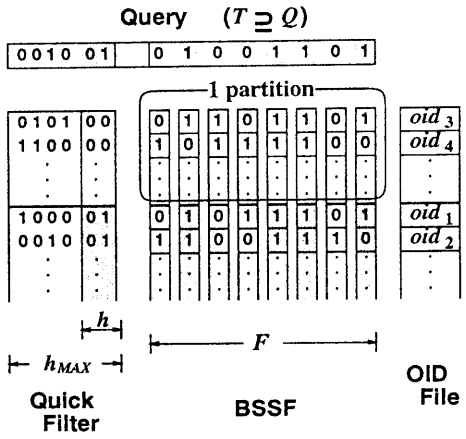

図 8: BSQF の構成

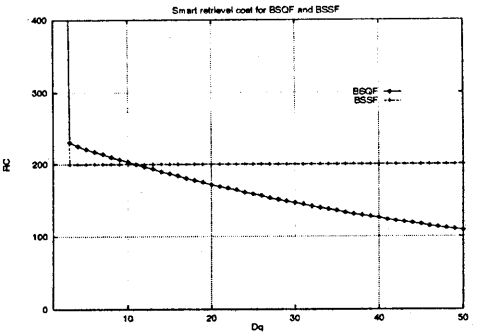

図 9: BSQF と BSSF の検索コスト ($T \supseteq Q$)

コストとなっていることが分かる。これは、P-BSSF、BSQF が分割シグネチャ法を用いることによって更新するシグネチャの存在するパーティションを特定できることによる。

8 格納コスト

$F = 1024, N = 800,00, pl \approx 75\%$ の時の、格納コストを表 4に示す。BSQF の格納コストは、P-BSSF のコストに Quick Filter 部のコスト (SC_{QF}、今回のパラメタでは、160) を加えたものとなる。表 4より、P-BSSF と BSQF の格納コストはほぼ同等といえる。また、BSSF は他の 2 つの pl 倍になっている。

ファイル	UC
P-BSSF, BSQF	214
BSSF	963

表 3: 更新コスト

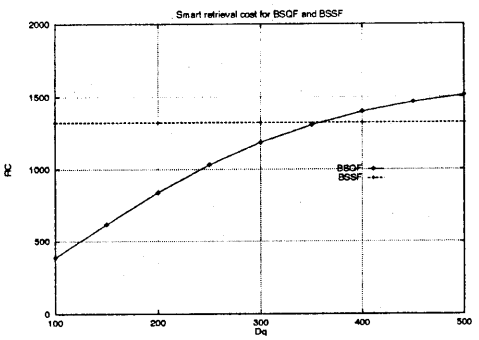

図 10: BSQF と BSSF の検索コスト ($T \subseteq Q$)

ファイル	SC_{QF}	SC_{BSSF}	SC_{OID}	SC
P-BSSF		32,768	2,048	34,816
BSQF	160	32,768	2,048	34,976
BSSF		25,600	1,563	27,163

表 4: 格納コスト

9 まとめ

本論文では検索対象の多い大規模データベースにおいて、シグネチャファイルによる検索コストの低減を目的とし、分割シグネチャファイルのアプローチとビットスライス方式のアプローチを組み合わせたシグネチャファイル構成法を提案し、その検索、更新、格納の各コストの見積りと評価を行なった。特に検索については、集合値検索を対象として、$T \supseteq Q$ と $T \subseteq Q$ の場合の 2 通りについて解析を行なった。

検索コストについては、特にパーティションの充足率の影響を考慮して検討を行なった。その結果、充足率が一定以上の場合及び $T \supseteq Q$ では D_q が大きい時、$T \subseteq Q$ では D_q が小さい時は、P-BSSF は有効であることが分かった。一方、充足率が低くかつ、$T \supseteq Q$ では D_q が小さい時、$T \subseteq Q$ では D_q が大きい時、P-BSSF の効率が BSSF に比べ悪くなることがあることが分かった。スマート検索を応用することによって、通常の検索コストを大きく削減可能で、スマート検索を行なった BSSF よりもスマート検索による効果が大きいことが示された。しかし、通常の検索処理の場合と同様、充足率や D_q の値によっては効率が悪くなる場合があることが分かった。この問題を緩和する手法として充足率をある程度コントロールできる linear hashing を用いた BSQF の提案を行ない、その有効性を検討した。

更新コストについては、一般的に BSSF は不利であるといわれている。これは、集合シグネチャを作成する際に "0" と "1" が同じ確率で出現するように、m_{opt} を用いる場合が多いからであり、今回のように比較的小さい m を用いることによって、ある程度は更新コストを抑えることができる。さらに、P-BSSF と BSQF では分割シグネチャファイル法を用いることにより、更新を行なうパーティションを特定することができ、BSSF と比較して更新コストを大きく削減可能である。本論文で提案した P-BSSF、BSQF の更新コストは今回設定したパラメタの下では BSSF の $\frac{1}{5} \sim \frac{1}{4}$ である。

格納コストは BSSF がもっとも小さく、P-BSSF、BSQF のコストに充足率をかけたコストになる。

以上より大規模データベースにおける集合値検索を対象として、BSSF に比較した P-BSSF 及び BSQF の有効性を示すことができた。

今後の課題としては、BSQF においてパーティションの分割、結合が行なわれた時の処理方法とそのコスト見積り、シグネチャファイルを構成する際の各パラメタの設定法等があげられる。

謝辞

シグネチャファイルに関して共に御検討いただいていた奈良先端科学技術大学院大学の石川佳治助手、ならびに筑波大学データベース研究室の諸氏に感謝致します。

参考文献

[1] C. Faloutsos and S. Christodoulakis. "Signature Files: An Access Method for Documents and Its Analytical Performance Evaluation". *ACM Trans. Off. Inf. Syst.*, Vol. 2, No. 4, pp. 267–288, 1984.

[2] Z. Lin and C. Faloutsos. "Frame-Sliced Signature Files". *IEEE Trans. Knowl. and Data Eng.*, Vol. 4, No. 3, pp. 281–289, 1992.

[3] P. Zezula, F. Rabitti, and P. Tiberio. "Dynamic Partitioning of Signature Files". *ACM Trans. Inf. Syst.*, Vol. 9, No. 4, pp. 336–369, 1991.

[4] Y. Ishikawa, H. Kitagawa, and N. Ohbo. "Evaluation of Signature Files as Set Access Facilities in OODBs". In *Proc. ACM SIGMOD*, pp. 247–256, 1993.

[5] J. K. Kim and J. W. Chang. "A Two-dimensional Dynamic Signature File Method". In *International Symposium on Advanced Database Technologies and Their Integration (ADTI'94)*, pp. 63–70, 1994.

[6] D. L. Lee and C. Leng. "A Partitioned Signature File Structure for Multiattribute and Text Retrieval". In *IEEE Trans. Knowl. and Data Eng.*, pp. 389–397, 1990.

[7] P. Ciaccia and P. Zezula. "Estimating Accesses in Partitioned Signature File Organizations". *ACM Trans. Inf. Syst.*, Vol. 11, No. 2, pp. 133–142, 1993.

[8] S.B. Yao. "Approximating Block Accesses in Database Organizations". *Communications of ACM*, Vol. 20, No. 4, pp. 260–261, 1977.

[9] W. Litwin. "Linear Hashing : A New Tool for File and Table Addressing". In *In Proceedings of 6th International Conference on VLDB*, pp. 212–223, 1980.