

集合シグネチャによるコレクションオブジェクトの問合せ処理

石川 佳治* 北川 博之†

* 奈良先端科学技術大学院大学 情報科学研究科

† 筑波大学 電子・情報工学系

複合オブジェクトを対象としたデータベースでは、集合、マルチセット、リストなどのコレクションオブジェクトを効率的、かつ柔軟に扱えるような枠組みが求められている。本稿では、筆者らが提案し研究を進めてきた集合シグネチャファイルの手法を一般化し、マルチセット、リストなどのさまざまなコレクションオブジェクトの検索について適用可能な索引手法とそれを用いた問合せ処理の枠組みを提案する。集合シグネチャファイルのファイル構造自体をコレクション型として記述し、集合シグネチャファイルを用いた問合せ処理をコレクション型の枠組みの中でとらえ、さまざまなコレクションオブジェクトの検索を統一のかつフォーマルに記述する。

A Framework for Query Processing of Collection Objects
with Set-based Signature Files

Yoshiharu Ishikawa* Hiroyuki Kitagawa†

* Graduate Institute of Information Science, Nara Institute of Science and Technology

† Institute of Information Sciences and Electronics, University of Tsukuba

Database systems supporting complex objects require efficient and flexible manipulation facilities for collection objects (e.g., set, bag, list). In this paper, we generalize the notion of *set-based signature files* – retrieval methods for set-valued objects –, and propose a unified framework to retrieve various kinds of collection objects. Set-based signature files themselves are specified as collection types, and queries concerning collection objects are translated into queries over set-based signature files as collection objects. Query processing for collection objects is formally described using a collection-oriented algebra and calculus.

1 はじめに

オブジェクト指向データベースなどの複合オブジェクトを対象としたデータベースでは、集合、マルチセット、リストなどのコレクションオブジェクトを効率的、かつ柔軟に扱えるような枠組みが求められている。筆者らはこれまで、テキスト検索の分野で一般的な索引手法であるシグネチャファイル [Fal92] の手法を集合データの検索に利用することを提案し、シグネチャファイルの物理的構成法や性能の評価などを行ってきた [IKO93, IK94, KFIO93]。本稿では、集合値検索のためのシグネチャファイル (集合シグネチャファイル) をより一般化し、集合以外のマルチセット、リストなどのさまざまなコレクションオブジェクトの検索に適用可能な枠組みを提案する。

近年、データベースプログラミング言語の研究分野において、コレクション型 (collection type)、あるいはバルクデータ型 (bulk data type) などと呼ばれる、コレクションオブジェクトのための言語の構成手法に関する研究が進められている。そのような研究には計算能力に焦点をあてた研究 [BTBN91, BTS91] も見られるが、一方で、並列処理可能な構成子をベースとして言語を構成することにより、効率的な実行を意図した研究も現れてきている [PSV92]。そのようなコレクション型に関する研究の一つとして、Fegaras らによる、モノイドに基づくコレクション型の構成法および問合せ言語の提案がある [Feg94, FM95]。

本稿では、集合シグネチャファイルの構造と集合に関する検索 (集合値検索) を Fegaras らのコレクション型の枠組みの中でとらえる。集合シグネチャファイル自体をコレクション型として記述し、集合値検索をコレクション型として定義された集合シグネチャファイルへの問合せとして帰着させる。さらに、集合値検索を一般化し、他のコレクションオブジェクトに対する問合せを支援することを試みる。モノイドに基づくコレクション型が内包する並列性により、本稿の枠組みで記述された問合せは容易に並列処理可能であると考えられる。

2 集合値検索と集合シグネチャファイル

まず、集合値検索の概念について説明する。集合値検索の例として、ODMG-93 standard [Cat94] の OQL 言語に基づく、以下の問合せを考える：

```
select distinct d.name
from d in Department
where d.orders is_superset? set("a", "b")
```

この問合せは、商品 "a"、"b" の両方とも注文した部門の名前を求める問合せである。set("a", "b") のように

問合せ条件中に現れる集合を問合せ集合 (query set) と呼び、データベース中に格納され比較対象となる集合をターゲット集合 (target set) と呼ぶ。上のような問合せを $T \supseteq Q$ (has-subset) の問合せと呼ぶ。 $T \ni q$ (has-element) は $T \supseteq Q$ の特殊な場合である。この他に、包含関係が逆の場合の $T \subseteq Q$ (is-subset)、ターゲット集合と問合せ集合が共通要素をもつかどうかを判定する $T \cap Q$ (has-intersection)、集合が等価であるか判定する $T \equiv Q$ (is-equal) などの問合せの種類が考えられる。

集合シグネチャファイル (set-based signature file) [IKO93, KFIO93] は、従来テキスト検索の分野で主に用いられてきたシグネチャファイル [Fal92] の手法を集合値検索に適用したものである。シグネチャ (signature) とは、個々のデータオブジェクトから生成される固定長のビット列であり、シグネチャを対応するデータオブジェクトの識別子とともに格納したものがシグネチャファイルである。シグネチャの生成には、スーパーインポーズドコーディング (super-imposed coding) と呼ばれるコーディング手法が用いられるのが一般的である。

集合シグネチャは以下のように生成される。集合データが与えられると、集合のそれぞれの要素 e について、 F ビットのビット列である要素シグネチャ (element signature) が生成される。要素シグネチャでは、 F ビットのうちの m ビットに "1" が設定される。この m 個のビット位置は、 $h(F, m, e) = \{b_1, \dots, b_m\}$ ($0 \leq b_i \leq F-1$) なるハッシュ関数 h により決定されるものとする。次に、これらの要素シグネチャのビットごとの論理和をとり (スーパーインポーズ処理)、与えられた集合に対する集合シグネチャ (set signature) が生成される。図 1 に、集合データ set("a", "b") に対する集合シグネチャの生成の例を示す。

集合の要素	要素シグネチャ
"a"	0001001000000100
"b"	0100000001000100
集合シグネチャ → 0101001001000100	

図 1: シグネチャの生成 ($F = 16, m = 3$)

作成された集合シグネチャは、対応するデータオブジェクトの識別子とともにシグネチャファイルに格納される。シグネチャファイルに格納されたそれぞれのシグネチャをターゲットシグネチャ (target signature) と呼ぶ。

先に示したような集合値に関する問合せが与えられると、まず問合せ集合より問合せシグネチャ (query signature) と呼ばれる集合シグネチャを生成する。次に、シグネチャファイル中のそれぞれのターゲットシグネチャについて問合せシグネチャとのパターンマッチを行ない、条件が満たされた場合には対応するデータオブジェクトを問合せを満たす候補とする。このようなオブジェクト

をドロップ (drop) と呼ぶ。パターンマッチの条件は問い合わせの種類に依存して定まり、

- $T \supseteq Q: S_Q \wedge S_T = S_Q$
- $T \subseteq Q: S_Q \wedge S_T = S_T$
- $T \sqcap Q: \text{weight}(S_Q \wedge S_T) \geq m$
- $T \equiv Q: S_Q \equiv S_T$

となる [KFIO93]。ただし、 S_Q, S_T はそれぞれ問合せシグネチャとターゲットシグネチャを表し、 $\wedge$ はビットごとの論理和を表す。また、 $\text{weight}(S)$ はウェイト (weight)、すなわち、シグネチャ S 中の “1” の数を返す関数である。

シグネチャファイルを用いた検索では、ハッシングによる要素シグネチャの作成とスーパーインボーズドコーディングにより、ターゲットシグネチャが誤ってパターンマッチしてしまうことがある。誤ってパターンマッチしたターゲットシグネチャに対応するデータオブジェクトをフォルスドロップ (false drop) と呼ぶ。シグネチャファイルを用いた検索では、最終的な結果からフォルスドロップの影響を取り除くフォルスドロップレゾリューション (false drop resolution) の処理が必要となる。

本稿では集合シグネチャファイル自体をコレクションオブジェクトとしてとらえ、ファイル構造や問合せ処理をフォーマルに記述する。記述のために、次節で述べる Fegaras らのコレクション型の枠組み [Feg94, FM95] を用いる。Fegaras らのコレクション型の研究は、コレクション型に分割統治に基づく並列性を導入しようという試みの一つでもある。シグネチャファイルはその構成から並列処理にも適していると考えられ、並列処理のためのファイル構成手法も数多く提案されている [GTZ92, Lin93, PF94]。本研究のもう一つの目的は、このような並列処理可能なシグネチャファイルの手法をコレクション型の検索手法として活用し、問合せ処理の効率化を実現することである。

3 モノイドに基づくコレクション型

コレクション型やバルクデータ型に関する研究には、単一の枠組みによりコレクション型を統一的に扱うことを目的の一つとしたものが見られる。統一的な枠組みに基づくことにより、最適化などで用いられる変換規則の数を減らすことができ、効率の面で有利となる。また、新たなコレクション型の導入も比較的容易となる。コレクション型の枠組みを構成する際に、分割統治 (divide-and-conquer) に基づく並列・分散処理が容易になるよう、分割統治処理に適した構成要素で言語を構成することが提案されている。そのような例としては、SVP [PSV92] や structural recursion [BTBN91, BTS91, BTBW92] がある。

Fegaras らは、数学的なモノイドの概念に基づいて、さまざまなコレクション型を統一的に扱い、分割統治処理に適したコレクション型の構成手法を提案した。彼らは、コレクション間の準同型写像に基づく汎用オペレータ monoid homomorphism により問合せを表現できる言語 (モノイド代数) を定義し、この言語を基にしてオブジェクト指向データベースの問合せ処理フェーズをフォーマルに記述することを研究の目的の一つとした [Feg94, FM95]。また、monoid comprehension と呼ばれる形式で表される、モノイド代数と等価なモノイド論理の提案も行った。

まずは、[Feg94] の記法に従ってモノイドの定義を与える。

定義 3.1 (モノイド) T を型とし、 $\text{zero}[T]$ を型 T のある値、 $\text{merge}[T]$ を型 $(T \times T) \rightarrow T$ の関数とする。 $\text{merge}[T]$ が結合律を満たし、 $\text{zero}[T]$ が単位元となるとき、三つ組み $(T, \text{zero}[T], \text{merge}[T])$ は型 T のモノイド (monoid) である。□

集合、リストなどのコレクション型は、以下の自由モノイドにより表現される。

定義 3.2 (自由モノイド) $T(\alpha)$ を型パラメータ α により決定される型とし、 $(T(\alpha), \text{zero}[T], \text{merge}[T])$ をモノイドとする。 $\text{unit}[T]$ が型 $\alpha \rightarrow T(\alpha)$ の関数であるとき、四つ組

$$(T(\alpha), \text{zero}[T], \text{unit}[T], \text{merge}[T])$$

は自由モノイド (free monoid) である。□

なお、以下では、 $T\{e_1, \dots, e_n\} = \text{merge}[T](\text{unit}[T](e_1), \dots, \text{merge}[T](\text{unit}[T](e_n), \text{zero}[T]))$ という略記法を用いる。たとえば $\text{set}\{1, 2\} = \text{merge}[T](\text{unit}[\text{set}](1), \text{merge}[\text{set}](\text{unit}[\text{set}](2), \text{zero}[\text{set}]))$ である。

[Feg94] に挙げられている自由モノイドの例を表 1 に示す。ただし、ordered set は重複を含まないリストであり、その merge 演算は $x \cup y = \text{append}(x, x - y)$ と定義される。C はモノイド上で交換律 (commutative law) が成立することを示し、I はべき等律 (idempotent law) が成立することを意味する。 $\{C, I\}$ のうち、モノイド T で成立する性質がモノイド S で成立する性質を包含するとき、 $S \preceq T$ と記述する。たとえば、set では C, I が成立し、bag では C のみ成立するので $\text{bag} \preceq \text{set}$ である。なお、表 1 では、さらに $[e_1, \dots, e_n] = \text{list}\{e_1, \dots, e_n\}$ などの略記法が用いられている。

並列処理の立場から見ると、モノイドでは結合律が成り立つため、左から右にデータをまとめる限りにおいて処理をどのように並列プロセスに分散するかは結果には影響しない。加えて、交換律が成り立てば処理結果はどのような順番で統合してもよいため、より並列性が向上することになる。

表 1: 自由モノイドの例

T	$\text{zero}[T]$	$\text{unit}[T](a)$	$\text{merge}[T]$	C/I
list	$[\]$	$\{a\}$	append	
set	$\{\}$	$\{a\}$	$\cup$	CI
bag	$\{\{\}\}$	$\{\{a\}\}$	$\cup$	C
ordered-set	$\{\}\}$	$\{a\}$	$\cup$	I
string	$""$	$"a"$	concat	

組み込みの型や演算に相当するモノイドは、単純モノイドと呼ばれ、以下のように定義される。

定義 3.3 (単純モノイド) $(T, \text{zero}[T], \text{merge}[T])$ がモノイドで、 $\text{unit}[T]$ が型 $T \rightarrow T$ の恒等関数 $\text{id} = \lambda x.x$ であるとき、四つ組み $(T, \text{zero}[T], \text{unit}[T], \text{merge}[T])$ は単純モノイド (simple monoid) である。□

単純モノイドの例としては表 2 のようなものが挙げられている [Feg94]。

表 2: 単純モノイドの例

T	type	$\text{zero}[T]$	$\text{unit}[T](a)$	$\text{merge}[T]$	C/I
sum	int	0	a	+	C
some	bool	false	a	$\vee$	CI
all	bool	true	a	$\wedge$	CI

操作の対象となるモノイドの構造は、以下のモノイド型の概念により定義される。

定義 3.4 (モノイド型) モノイド型は、以下のいずれかの形式をもつ。

- T (T は単純モノイド)
- $T(\text{prtype})$ (T は自由モノイド)
- $T(\text{type})$ (T は自由モノイド)
- $\langle a_1 : t_1, \dots, a_n : t_n \rangle$ (レコード型)

ただし、 prtype は int, bool, char であり、 $t_1, \dots, t_n$ はモノイド型である。レコードの構成要素を取り出すのに、 $a_i(x)$ または $x.a_i$ という記法が用いられる。□

なお、 $t_1 \times t_2 = \langle \pi_1 : t_1, \pi_2 : t_2 \rangle$ という略記法で順序対 (ordered pair) を表現する。

モノイド間の準同型写像 monoid homomorphism は、問合せを記述する上で中心的な役割を果たす。

定義 3.5 (Homomorphism) 自由モノイド $(T(\alpha), \text{zero}[T], \text{unit}[T], \text{merge}[T])$ からモノイド $(S, \text{zero}[S], \text{merge}[S])$ への準同型写像 (homomorphism) $\text{hom}[T, S]$ は型 $(\alpha \rightarrow S) \rightarrow T(\alpha) \rightarrow S$ をもち、以下のように定義される：

$$\begin{aligned}
\text{hom}[T, S](f) \text{zero}[T] &= \text{zero}[S] \\
\text{hom}[T, S](f)(\text{unit}[T](a)) &= f(a) \\
\text{hom}[T, S](f)(\text{merge}[T](x, y)) &= \\
&\quad \text{merge}[S](\text{hom}[T, S](f) x, \text{hom}[T, S](f) y)
\end{aligned}$$

ただし、 $T \preceq S$ である。□

monoid homomorphism の例としては、以下のようなものがある：

$$\begin{aligned}
a \in x &= \text{hom}[\text{set}, \text{some}](\lambda e.(e = a)) x \\
\text{length}(x) &= \text{hom}[\text{list}, \text{sum}](\lambda e.1) x
\end{aligned}$$

[Feg94] では、問合せ言語としてモノイド項 (monoid term) 上の代数系であるモノイド代数 (monoid algebra) が提案されている。 $\text{hom}[T, S](\lambda v.e_1) e_2$ の形式の項はモノイド項の一種である。モノイド項の詳しい定義は省略する。

定義 3.6 (Monoid Comprehension) モノイド $(T, \text{zero}[T], \text{unit}[T], \text{merge}[T])$ 上の comprehension は以下のように定義される：

$$\begin{aligned}
T\{e\} &= \text{unit}[T](e) \\
T\{e \mid x \leftarrow u, \bar{\tau}\} &= \text{hom}[S, T](\lambda x.T\{e \mid \bar{\tau}\}) u \\
T\{e \mid \text{pred}, \bar{\tau}\} &= \text{if pred then } T\{e \mid \bar{\tau}\} \text{ else } \text{zero}[T]
\end{aligned}$$

ただし、 u は $S \preceq T$ を満たす自由モノイド S により $S(\alpha)$ と表される型を有している。 $x \leftarrow u$ の形式の項はジェネレータ (generator) と呼ばれる。pred は述語であり、ガード (guard) あるいはフィルタ (filter) と呼ばれる。 $\bar{\tau}$ はジェネレータまたはガードをコンマで区切ったシーケンス (空も可) である。□

monoid comprehension の例を示す。

$$\begin{aligned}
a \in x &= \text{some}\{a = e \mid e \leftarrow x\} \\
\text{filter}(p)(x) &= \text{set}\{e \mid e \leftarrow x, p(e)\} \\
\text{length}(x) &= \text{sum}\{1 \mid e \leftarrow x\} \\
\text{count}(x, a) &= \text{sum}\{1 \mid e \leftarrow x, e = a\}
\end{aligned}$$

[Feg94] では、モノイド代数と等価なモノイド論理 (monoid calculus) が与えられている。モノイド論理の項はモノイド代数の項と同様に定義されるが、monoid homomorphism の形式の項の代わりに monoid comprehension の形式の項が含まれる。

4 集合値検索を含む問合せの表現

4.1 集合値検索条件の表現

以下では、集合値検索条件をモノイド論理により表現することを試みる。集合に関する比較条件のうち、メンバシップは前節の例で示されている。包含関係に基づく検索条件は、some と all の組合せにより表現できる。

$$x \supseteq y = \text{all}\{e \in x \mid e \leftarrow y\}$$

たとえば、 $a = \{a_1, a_2, \dots, a_n\}$ としたとき、上の式は

$$(a_1 \in x) \wedge (a_2 \in x) \wedge \dots \wedge (a_n \in x)$$

となり、メンバシップ検索に帰着される。集合どうしに共通要素があるかどうかの判定は、

$$x \sqcap y = \text{some}\{e \in x \mid e \leftarrow y\}$$

で表せる。集合の等価性は、たとえば

$$x \equiv y = \text{all}\{(x \supseteq b) \wedge (y \supseteq a) \mid a \leftarrow x, b \leftarrow y\}$$

や、

$$x \equiv y = \text{all}\{(x \supseteq y) \wedge (y \supseteq x) \mid\}$$

で表せる。真部分集合であるかどうかは、

$$x \sqsubset y = \text{set}\{(x \supseteq y) \wedge \text{not } o(x \equiv y) \mid\}$$

で与えられる。

4.2 問合せ処理の例

4.1節で導入した $x \supseteq y$ を用いると、2節冒頭で示した OQL による問合せは、モノイド論理により

$$\text{set}\{d.\text{name} \mid d \leftarrow D, d.\text{orders} \supseteq Q\}$$

と表現できる。ただし、 D はクラス `Department`、 Q は集合 `set("a", "b")` を表すものとする。この問合せは直接的にモノイド代数の表現に変換できる。

$$\text{hom}[\text{set}, \text{set}](\lambda d.\text{set}\{d.\text{name} \mid d.\text{orders} \supseteq Q\}) D$$

この表現は、さらに [Feg94] のモノイド論理からモノイド代数への変換規則を用いて以下のように変形できる。

$$\begin{aligned} & \text{hom}[\text{set}, \text{set}](\lambda d.\text{set}\{d.\text{name} \mid d.\text{orders} \supseteq Q\}) D \\ &= \text{hom}[\text{set}, \text{set}](\lambda d.\text{if all}\{d.\text{orders} \ni q \mid q \leftarrow Q\} \\ & \quad \text{then } \{d.\text{name}\} \text{ else } \{\}) D \\ &= \dots \\ &= \text{hom}[\text{set}, \text{set}](\\ & \quad \lambda d.\text{if hom}[\text{set}, \text{all}](\\ & \quad \quad \lambda q.\text{hom}[\text{set}, \text{some}](\lambda o.(q = o)) d.\text{orders}) Q \\ & \quad \text{then } \{d.\text{name}\} \\ & \quad \text{else } \{\}) D \end{aligned}$$

つまり、集合値に関する検索条件が組み込み型上の等価性の判定条件に帰着されたことになる¹。このように、[Feg94, FM95] の枠組みでは、コレクション型に対する包含関係、メンバシップなどの検索条件は、等価性の判定などの基本的演算に細分化される。分割統治に基づくこのような処理の細分化は並列・分散処理における原則であるが、集合シグネチャファイルのようなコレクション型に対する索引を有効に生かせない。

¹[Feg94] で提案された正規化アルゴリズムではこれより先、さらに多少の変換が行われる。

5 コレクション型としての集合シグネチャファイル

この節では、集合シグネチャファイルをコレクション型として記述し、集合シグネチャを用いた検索をコレクションオブジェクト上の問合せとして表現することを考える。コレクション型による問合せ処理の記述は一種の仕様と考えられ、下層のシステム構成や実装方式に応じて並列性を抽出し、問合せ処理の効率化を図れると考えられる。

5.1 集合シグネチャの記述

まず、集合シグネチャを一般的に表現する。このために、集合 $\{0, 1\}$ なる単純モノイド *bit* が存在するものとする。モノイド *bit* の zero 演算は 0、merge 演算はビットの論理和である。次に、集合シグネチャを表す自由モノイド *ssig* を定義する。ただし、与えられた型 α に関わらず、集合シグネチャの型 $T(\alpha) = \text{ssig}[F, m, h](\alpha)$ は F ビットのビットベクタとなる。*ssig* の zero 演算は次のように表される。

$$\text{zero}[\text{ssig}[F, m, h](T)] = \langle \text{zero}[\text{bit}], \dots, \text{zero}[\text{bit}] \rangle \quad (F \text{ times})$$

つまり、 F 個の “0” (= $\text{zero}[\text{bit}]$) からなるビットベクタ ($\langle \dots \rangle$ で表す) が生成される。unit 演算は次のようになる。

$$\text{unit}[\text{ssig}[F, m, h](T)](a) = \langle b_0, \dots, b_{F-1} \rangle$$

ただし、

$$b_i = \begin{cases} 1 & \text{if } i \in h(F, m, a) \\ 0 & \text{otherwise} \end{cases}$$

であり、 F ビットのうちの m ビットに “1” の値をもつビットベクタがハッシュ関数 h により作成される。最後に merge 演算を示す。

$$\begin{aligned} & \text{merge}[\text{ssig}[F, m, h](T)](\langle a_0, \dots, a_{F-1} \rangle, \langle b_0, \dots, b_{F-1} \rangle) \\ &= \langle \text{merge}[\text{bit}](a_0, b_0), \dots, \text{merge}[\text{bit}](a_{F-1}, b_{F-1}) \rangle \end{aligned}$$

merge は、スーパーインポーズドコーディングの操作を表していることになる。以上により、集合 x が与えられたとき、ビット長 F 、要素シグネチャのウェイト m でハッシュ関数 h により作成される集合シグネチャは、以下のモノイド論理式で表現できる。

$$\text{make_ssig}(x, F, m, h) = \text{ssig}[F, m, h]\{e \mid e \leftarrow x\}$$

5.2 SSF 方式の集合シグネチャファイルの記述

最も基本的なシグネチャファイルの構成方式はシーケンシャルシグネチャファイル (sequential signature file, SSF) [Fal92] である。SSF は、各オブジェクトに対し生成されたシグネチャと、オブジェクトより与えられた識

別子 (id) のペアをリストにして格納したものである。id を整数値で表すことにすると、パラメータ F, m, h で作成された SSF のファイル構造は

$$SSF[F, m, h] = [(ssig[F, m, h], int)]$$

というモノイド型で表される。しかし、シグネチャファイルの並列性をさらに上げるには、リストではなく交換律が成り立つ bag として

$$SSF[F, m, h] = \{[(ssig[F, m, h], int)]\}$$

と扱ったほうが有利である²。

よって、クラス C の集合値属性 A について型 $SSF[F, m, h]$ の集合シグネチャファイルを作成する処理は、

$$\begin{aligned} & \text{make_SSF}(C, A, F, m, h) \\ &= \text{bag}\{(\text{make_ssig}(c.A, F, m, h), c.oid) \mid c \leftarrow C\} \end{aligned}$$

と表せる³。このような集合シグネチャファイルの定義により、複数のマシンにシグネチャファイルを分散して持たせ、分割統治に基づいた検索処理を行うことも可能となる。

5.3 SSF 方式の集合シグネチャファイルによる検索の表現

$T \supseteq Q$ の検索をまず考える。2 節で述べた検索時のパターンマッチの条件は、問合せシグネチャのそれぞれの“1”のビット位置に対応するターゲットシグネチャのビット位置の値が“1”となることと等価である。よって、 t, q を、それぞれターゲットシグネチャ、問合せシグネチャを表す $ssig[F, m, h]$ 型の集合シグネチャとしたとき、 $T \supseteq Q$ に対する集合シグネチャファイルのパターンマッチ条件は

$$\begin{aligned} & \text{has-subset}(t, q) = \\ & \text{all}\{a[i] = 1 \mid a[i] \leftarrow t, b[j] \leftarrow q, i = j, b[j] = 1\} \end{aligned}$$

となる。ただし、 $a[i]$ は (a, i) の略記である。このモノイド項は、 t と q が $T \supseteq Q$ のパターンマッチ条件を満たした場合に true になる。同様に、 $T \subseteq Q$ については、

$$\begin{aligned} & \text{is-subset}(t, q) = \\ & \text{all}\{a[i] = 0 \mid a[i] \leftarrow t, b[j] \leftarrow q, i = j, b[j] = 0\} \end{aligned}$$

と表される。 $T \cap Q$ については、少なくとも m 個の同じビット位置に“1”がセットされるという条件から、

$$\begin{aligned} & \text{has-intersection}(t, q, m) = \\ & \text{length}(\text{set}\{i \mid a[i] \leftarrow t, b[j] \leftarrow q, \\ & \quad i = j, a[i] = 1, b[j] = 1\}) \geq m \end{aligned}$$

²シグネチャファイル中での id の一意性が保証されているなら、さらに set として扱うことも考えられる。

³ $\text{make_ssig}(x, F, m, h)$ はモノイド項ではないが、式を簡単化するためこのような表記を行う。実際には先に示した make_ssig に対応するモノイド項の内容に置き換えられる。

となる。 $T \equiv Q$ については

$$\begin{aligned} & \text{is-equal}(t, q) = \text{all}\{a[i] = b[j] \mid a[i] \leftarrow t, b[j] \leftarrow q, i = j\} \\ & \text{と与えられる。} \end{aligned}$$

問合せ集合 Q が与えられたとき、 $\text{make_ssig}(Q, F, m, h)$ により問合せシグネチャが作られる。よって、 $SSF[F, m, h]$ 型の集合シグネチャファイル S_{SSF} に対し、

$$\begin{aligned} & \text{ssig-scan_SSF_has-subset}(S_{SSF}, Q, F, m, h) \\ &= \text{set}\{s.\pi_2 \mid s \leftarrow S_{SSF}, \\ & \quad \text{has-subset}(s.\pi_1, \text{make_ssig}(Q, F, m, h))\} \end{aligned}$$

により $T \supseteq Q$ のパターンマッチ条件を満たす集合データに対応する id の集合が検索されることになる。これを一般化し、集合値検索を実行する問合せを

$$\text{ssig-scan_SSF}(S_{SSF}, Q, qtype, F, m, h)$$

のように表せる。 ssig-scan_SSF は、引数 $qtype$ の値 (has-subset , is-subset , ...) に従って、対応するパターンマッチ条件で SSF 方式に基づく集合シグネチャファイルをスキャンし、パターンマッチした id の集合を返す。

上の例では SSF 方式を考慮したが、集合シグネチャファイルの物理的な構成方式としては他にもさまざまなものが考えられる [IKO93, Fal92]。X 方式に基づく集合シグネチャファイルに対する検索は、X 方式の集合シグネチャファイルを S_X として、同様に

$$\text{ssig-scan}_X(S_X, Q, qtype, \dots)$$

と表現できる (第 4 引数以降には X 方式のシグネチャファイルに特有のパラメータが入る)。そこで、さまざまな集合シグネチャファイルを抽象化した論理的な集合シグネチャファイル S を考え、その検索を

$$\text{ssig-scan}(S, Q, qtype)$$

と表すことができる。これにより、モノイド論理による問合せの表現の中に、

$$\text{set}\{f(c) \mid c \leftarrow C, c.A \supseteq Q\}$$

などというパターンが存在した場合、これを集合値検索ととらえ $\text{ssig-scan}(S, Q, \text{has-subset})$ と置き換え、集合シグネチャファイル S の物理構成にしたがったたとえば ssig-scan_SSF を呼ぶなどの段階的な問合せ処理の詳細化が考えられる。

5.4 フォルドロップレゾリューションと問合せ実行

オブジェクトの id i に対し、 $\text{ref}(i)$ で参照されたオブジェクトが与えられるとする。集合シグネチャファイルの検索により得られた id の集合を I として、

$$\text{set}\{\text{ref}(i) \mid i \leftarrow I\}$$

により、対応するオブジェクトの集合が得られる。しかし、シグネチャファイルの検索では、一般にフォルスドロップレゾリューションの処理が必要である。クラス C の集合値属性 A に対し問合せ集合 Q を与えて $T \supseteq Q$ の問合せを行った場合、集合シグネチャファイルの検索結果が I ならば、

$$\text{FDR}(I, A, Q) = \text{set}\{o \mid o \leftarrow \text{set}\{\text{ref}(i) \mid i \leftarrow I\}, o.A \supseteq Q\}$$

によりフォルスドロップレゾリューションの処理が実現できる。この場合も問合せの種類を引数とした一般化が可能である。これを $\text{FDR}(I, A, Q, \text{qtype})$ と表す。

以上の結果により、4.2 節で集合シグネチャファイルが存在しない場合の問合せ処理を示したモノイド論理式

$$\text{set}\{d.\text{name} \mid d \leftarrow D, d.\text{orders} \supseteq Q\}$$

は以下のように変形される。

$$\text{set}\{d.\text{name} \mid d \leftarrow \text{FDR}(\text{ssig-scan}(S, Q, \text{has-subset}), \text{orders}, Q, \text{has-subset})\}$$

ただし、 S はクラス C の属性 A 上の集合シグネチャファイルである。

属性 A 上の集合シグネチャファイル S の型が $\text{SSF}[F, m, h]$ であるとき、上の問合せは 4.2 節の問合せ例と同様、[Feg94] の変換規則により canonical form に変換され、最終的には

$$\begin{aligned} &\text{hom}[\text{bag}, \text{set}](\text{hom}[\text{set}, \text{set}](\lambda d.\{d.\text{name}\})o \\ &\quad (\text{hom}[\text{set}, \text{set}](\lambda o.\text{set}\{o \mid o.\text{orders} \supseteq Q\})o \\ &\quad (\text{hom}[\text{set}, \text{set}](\lambda i.\{\text{ref}(i)\})o \\ &\quad (\lambda s.\text{if has-subset}(s.\pi_1, \\ &\quad \quad \quad \text{make_ssig}(Q, F, m, h)) \\ &\quad \text{then } \{s.\pi_2\} \\ &\quad \text{else } \{\})\})\}) S \end{aligned}$$

というモノイド項が得られる。ただし、式の簡単化のため has-subset , make_ssig については展開は行っていない。このモノイド項は、

$$\text{hom}[\text{bag}, \text{set}](f) S$$

という形式をもち、シグネチャファイル S が実際には n 台のマシン上の $\text{SSF}[F, m, h]$ 型の集合シグネチャファイル $S_1, \dots, S_n$ として分散配置されていた場合、

$$\text{hom}[\text{bag}, \text{set}](f) S_1 \cup \dots \cup \text{hom}[\text{bag}, \text{set}](f) S_n$$

と実行される。すなわち、各マシンにおいて並列に計算した結果を $\cup$ により集計したものが最終結果となる。

6 他のコレクション型への適用

集合以外のマルチセット (bag), リストなどのコレクション型についても、集合シグネチャファイルを利用し検索処理を効率化することが可能である。これは、コレクションオブジェクトに対する比較演算を集合値の比較演算に帰着することにより行える。

例として、クラス C の属性 A がマルチセットであるときに、この属性に関するマルチセットの包含関係に基づく検索を考える。ただし、マルチセットの包含関係を、ここでは

$$x \supseteq y = \text{all}\{\text{count}(x, a) \geq \text{count}(y, a) \mid a \leftarrow y\}$$

と定義する。属性 A について、 $\text{make_SSF}(C, A, F, m, h)$ などのように集合シグネチャファイルを作成する問合せを発行した場合、モノイドに基づくコレクション型の枠組みの一様性により集合シグネチャファイルを実際に作成することが可能である。ただし、集合シグネチャに対応するモノイド型 $\text{ssig}[F, m, h]$ の定義がスーパーインボーズドコーディングに基づくことより、マルチセット中の要素の多重度 (multiplicity) に関する情報は集合シグネチャでは失われてしまう。たとえば、 $\{\{1, 2\}\}$ に関するシグネチャと $\{\{1, 2, 2\}\}$ に関するシグネチャは等しくなり、 $\{\{1, 2, 2\}\}$ で作成したシグネチャを与えてパターンマッチを行った場合 $\{\{1, 2\}\}$ に対するシグネチャともマッチしてしまうことになる。

しかし、シグネチャファイルを用いた検索では最終的にフォルスドロップレゾリューションを行うことから、集合シグネチャファイルを検索して得られる id の集合には検索条件を満たさないものが含まれてもよいことに注意する。すなわち、シグネチャファイルはできるだけ検索結果の候補を絞り込むために用いられるのであり、検索結果には余分なミスマッチが含まれていてもよい。マルチセットやリストに対するシグネチャファイルでは、要素の多重度や順序などの情報は失われてしまうが、あくまでも絞り込みの手段としてシグネチャファイルを利用し、フォルスドロップレゾリューションにより最終的にふり落としをすることで効率的な検索が可能となる。たとえば、5.4 節の例で示した問合せにおいて orders 属性がマルチセットであり、 $\supseteq$ に関する問合せ (has-subbag) がなされた場合、

$$\text{set}\{d.\text{name} \mid d \leftarrow \text{FDR}(\text{ssig-scan}(S, Q, \text{has-subbag}), \text{orders}, Q, \text{has-subset})\}$$

という問合せにより検索処理が実行できる。これを一般化すると、

$$\text{set}\{g(c) \mid c \leftarrow \text{FDR}(\text{ssig-scan}(S, Q, \text{qtype}), A, Q, \text{qtype}')\}$$

という形式でコレクション型の検索が一般的に表現できることがわかる。ただし, *qtype* はマルチセット, リストなど, 検索対象となっているコレクション型における検索条件 ($T \supseteq Q$ など) で, *qtype'* は *qtype* に対応する, 集合のセマンティクスでの検索条件 ($T \supseteq Q$ など) である。

7 おわりに

本稿では, 集合シグネチャファイルのファイル構造や検索をコレクション型の枠組みの中でとらえ, コレクション型の検索への一般化を図った。また, 並列性を内包するコレクション型の枠組みにより, シグネチャファイルに存在する並列性を活かせる形での抽象的な問合せ処理の記述をはかった。今後は, この枠組みのもとで最適化などの問合せ処理について, また, 並列処理に基づくシグネチャファイルのファイル構成方式との統合などを行っていく。

謝辞

日頃から御支援いただいている奈良先端科学技術大学院大学 マルチメディア統合システム講座 (植村研究室) ならびに筑波大学データベース研究室の皆様に感謝いたします。

参考文献

- [BTBN91] V. Breazu-Tannen, P. Buneman, and S. Naqvi: "Structural Recursion as a Query Language", in *Proc. 3rd Intl. Workshop on DBPL*, pp. 9-19, Nafplion, Greece, 1991.
- [BTBW92] V. Breazu-Tannen, P. Buneman, and L. Wong: "Naturally Embedded Query Languages", in *ICDT '92*, pp. 140-154, Springer-Verlag, 1992 (LNCS 646).
- [BTS91] V. Breazu-Tannen and R. Subrahmanyam: "Logical and Computational Aspects of Programming with Sets/Bags/Lists", in *Automata, Languages and Programming*, pp. 60-75, Springer-Verlag, 1991, (LNCS 510).
- [Cat94] R. Cattell ed.: *The Object Database Standard: ODMG-93*, Morgan Kaufmann, San Francisco, California, 1994, Release 1.1.
- [Fal92] C. Faloutsos: "Signature Files", in W. B. Frakes and R. Baeza-Yates eds., *Information Retrieval - Data Structures and Algorithms*, pp. 44-65, Prentice-Hall, Englewood Cliffs, NJ, 1992.
- [Feg94] L. Fegaras: "A Uniform Calculus for Collection Types", Technical Report 94-030, Oregon Graduate Institute, 1994.
- [FM95] L. Fegaras and D. Maier: "Towards an Effective Calculus for Object Query Languages", in *Proc. of ACM SIGMOD*, San Jose, California, May 1995.
- [GTZ92] F. Grandi, P. Tiberio, and P. Zezula: "Frame-Sliced Partitioned Parallel Signature Files", in *Proc. of 15th ACM SIGIR Conf.*, pp. 286-297, Copenhagen, Denmark, June 1992.
- [IK94] Y. Ishikawa and H. Kitagawa: "Analysis of Indexing Schemes to Support Set Retrieval of Nested Objects", in *Proceedings of International Symposium on Advanced Database Technologies and Their Integration (ADTI'94)*, pp. 55-62, Nara, Japan, Oct. 1994.
- [IKO93] Y. Ishikawa, H. Kitagawa, and N. Ohbo: "Evaluation of Signature Files as Set Access Facilities in OODBs", in *Proc. ACM SIGMOD Conf.*, pp. 247-256, Washington, D.C., May 1993.
- [KFIO93] H. Kitagawa, Y. Fukushima, Y. Ishikawa, and N. Ohbo: "Estimation of False Drops in Set-valued Object Retrieval with Signature Files", *Proc. of the 4th Intl. Conf. on Foundations of Data Organization and Algorithms (FODO)*, pp. 146-163, Springer-Verlag, Oct. 1993, (LNCS 730).
- [Lin93] Z. Lin: "Concurrent Frame Signature File", *Distributed and Parallel Databases*, 1(3):231-249, July 1993.
- [PF94] G. Panagopoulos and C. Faloutsos: "Bit-Sliced Signature Files for Very Large Text Databases on a Parallel Machine Architecture", in *Proc. of 4th Intl. Conf. on EDBT*, pp. 378-392, Cambridge, UK, Mar. 1994.
- [PSV92] D. S. Parker, E. Simon, and P. Valduriez: "SVP - A Model Capturing Sets, Streams, and Parallelism", in *Proc. of VLDB*, pp. 115-126, Vancouver, Canada, Aug. 1992.