

構造化文書とデータベースの融合について

吉川正俊 市川 修 植村俊亮

奈良先端科学技術大学院大学 情報科学研究科

SGML のような構造化文書では、文書中にその構造に関する情報を記述しているために、従来の内容に基づく検索に加えて、文書の論理的な構造に基づく検索処理を行うことが可能である。本論文では、SGML 文書における章、節などのマーク付けに加え、単語レベルでのマーク付けをも考慮し、単語が表す実世界オブジェクトの情報をデータベースで一元管理するための機構を提案する。本機構のもとでは、通常 SGML 属性に格納される情報がデータベース内で一元管理される。しかも、データベースによる属性管理は、利用者には透明である。さらに、このようにデータベースと融合された SGML 文書を検索するための問い合わせ言語を与える。この問い合わせ言語は、特定のデータベーススキーマを前提としておらず、DTD が表す文書要素間の構造を表現した DTD グラフを仮想スキーマとみなす。

Amalgamating Structured Documents and Databases

Masatoshi YOSHIKAWA Osamu ICHIKAWA Shunsuke UEMURA

Graduate School of Information Science
Nara Institute of Science and Technology

8916-5 Takayama, Ikoma, Nara 630-01, Japan

SGML describes informations on the structure of documents. By making use of these informations, documents can be retrieved not only by content-based queries but also by structure-based queries. Typical document structures are chapters, sections and paragraphs. In this paper, we also take into account the mark-up of words in documents, and propose a mechanism to uniformly manage the information on real-world objects which the marked-up words represent. In our mechanism, data normally stored in SGML attributes are transparently stored in ordinary databases. Also, we present a query language to retrieve SGML documents which are coupled with databases in this manner. The query language does not assume a particular database schema; instead, it utilizes DTD graphs, representing element structures of DTDs, as virtual schemas.

1 はじめに

SGML は、共通符号化を基本とする文書記述言語であり、ISO[3] および JIS[4] で標準化されている。近年、出版系での利用とともに、WWW(World Wide Web) のようなハイパーテキストへの応用など広く普及し始めている。各 SGML 文書は、基本的に

- 文書型定義 (DTD: Document Type Definition): 生成される文書の論理構造を明記した文法規則
- マーク付き文書: タグ付きで情報内容を含む文書インスタンス

から成る。マーク付き文書は、DTD に従って文書中にタグを付けることにより、文書の内容に加えて論理構造を表現したものである。SGML は、このように文書を論理的に構造化することにより、各種の文書処理の合理化、文書の交換および再利用を促進することを目的としている。この目的を実現するためにデータベースは本質的な役割を果たす。そのため、SGML 文書データベースに関する研究が盛んになりつつある [5]。

SGML 文書をデータベースに格納することにより文書の内容のみならず論理構造に基づいた検索が可能となる。たとえば、DTD で宣言された文法に基づいてデータベーススキーマを構築する例として、Christophides ら [2] は、O₂モデルを拡張した OODB スキーマとして DTD を表現し、O₂SQL を拡張した問い合わせ言語によりマーク付き文書を検索することを提案している。また、Blake ら [1] は、個々の DTD とは独立した構造を持つ仮想関係をもとに SQL によるマーク付き文書の検索法を与えている。

このような文書の論理構造に基づく検索は重要な機能である。しかし、SGML のマーク付けの用途は非常に広範囲に及び、上記のような章、節などのいくつかの文章をマーク付けの対象とする「巨視的な」利用法のみならず、文書中のある単語が表現する実世界のオブジェクトに関する情報を記述するためにその単語をマーク付けするという「微視的な」利用も可能である。本研究では、SGML データベースに関する他の多くの研究が前提としてきた「巨視的な」マーク付けに加え、単語レベルの「微視的な」マーク付けをも考慮する。SGML の枠組では、文字列という表層によって参照される実世界オブジェクトに関する情報は通常 SGML 属性で表現される。その多くのものは実際にはデータベースにより一元管理を行なうことが望ましい。本研究では、このような一元管理を実現し、しかもこのように管理されたデータベース属性を SGML 文書内では仮想的に通常の SGML 属性と区別なく扱えるようにするための機構を与える。また、このような機構を通常の SGML 体系に最小の機能追加により実現するための方法について述べる。この管理機構により、構造化文書の中にデータベースに格納されたデータを取り込んで、それらを統一的に扱うことが可能となる。

さらに、このようにデータベースと融合された SGML 文書を検索するための問い合わせ言語について述べる。本研究で考える問い合わせは、以下のような特徴を持つ。

- 複数の DTD が存在し、それらに準拠する大量の文書を統一的に扱う。
- SGML に必須の要素である順序関係および SGML 属性も扱う。
- SGML 文書中の文書要素とデータベースを上述の意味で融合して、文書に対して意味的な問合せも行う。
- 検索のために特別なスキーマ定義を行わず、個々に作成された DTD および マーク付き文書をそのまま検索対象とする。

- SGML の論理的な構造記述能力を利用するために、個々の DTD から導出可能な DTD グラフを仮想スキーマとして扱い SQL により問合せを行う。

以下、2 章では、SGML について述べ、3 章で SGML 属性を利用した構造化文書とデータベースとの情報の相互利用について述べる。また、4 章では、SGML で作成された文書に対して SQL によって行われる問合せを述べ、5 章で、今後の課題について述べる。

2 SGML

SGML は、国際規格 ISO 8879 で定められた国際標準であり、文書の構造および属性の記述を表現するものである。大別すると以下のようなになる。

- 文書型定義 DTD (図 1)
文書が準拠しなければならない文法を表したものであり、以下のものから構成される。
 - 要素宣言 (ELEMENT): マーク付き文書を構成するためのエレメント名を宣言し、文書構造を定義する部分であり、要素の名前、出現回数、出現順序等が定義される。
 - 属性定義並び宣言 (ATTLIST): その要素に指定できる SGML 属性とそのとり得る値を定める。
 - 実体宣言 (ENTITY): その型の文書の中で参照できる実体を定め、実際の文書中での文字列の置き換え等が定義される。
- マーク付き文書
文書にタグが付いていることによって、“章”や“節”というような論理的なまとまりの開始位置および終了位置が特定可能である。

```

<!DOCTYPE article [
  <ELEMENT article -- (title, author+, abstract, section+) >
  <ELEMENT section -- (title, p*, subsec*)>
  <ELEMENT subsec -- (title, p+, subsec*)>
  <ELEMENT title -- (#PCDATA)>
  <ELEMENT author -- (#PCDATA)>
  <ELEMENT abstract -- (#PCDATA)>
  <ELEMENT p -- (#PCDATA)>
]>

```

図 1: 文書型定義の例

3 文書要素とデータベースの融合方式

SGML のような構造化文書では、文書中にその構造に関する情報を記述しているために、従来の内容に基づく検索に加えて、文書の論理的な構造に基づく検索処理を行うことが可能である。また、文書中のある文書要素に関してのデータベース情報が存在すれば、それらの情報を利用して、更に多様な問合せを行うことが可能になる。

ここでは、SGML 属性を利用して、構造化文書中の文書要素にデータベースに格納されたデータを持たせることにより、文書要素とデータベースの融合を図ることについて考える (図 2)。

3.1 基本概念

3.1.1 SGML 属性

一般に、SGML で作成される文書には SGML 属性が含まれ、文書の作成時に開始エレメントとともに入力される。実際に使用される SGML 属性としては、以下の様なものが考えられる。

1. 文書の構造に関するもの

- 特定のエレメント出現を一意に識別する。
<figure figid='piechart1'>
- 特定のエレメントの型を識別する。
<div1 type='section'>

2. 文書の内容に関するもの

- <person age=20> 田中 </person>

SGML 属性の用途は数多く存在するが、例えば、2. に関する属性については、典型的にデータベースに格納されたデータを表している。そこで、このような SGML 属性について、実際に存在するデータベースに格納されたデータを参照させ、構造化文書中にデータベース情報を取り込み構造化文書とデータベースとの相互利用をはかる。

3.1.2 SGML 属性によるデータベースオブジェクトの参照

構造化文書中の文書要素をデータベースと融合させるためには、新たに文書要素を表現するエレメントおよび文書要素を特定するための SGML 属性が必要となる。データベースと融合させる文書要素が表すオブジェクトは、人であったり都市であったりするために、それらの要素を表す名前をエレメント名として定義しなければならない。また、文書要素とデータベース中に存在する実体に対応付けるために、SGML 属性として oid を持たせる。

```

<person oid = 1234> 田中 </person>
<city oid = 2345> 奈良 </city>

```

これらの文書要素について、oid は、データベース中に存在する実体に対応する識別子を表し、エレメント名が文書要素の型を表すので、各々のエレメントから以下のような SQL が作成される。

SELECT *	SELECT *
FROM person	FROM city
WHERE oid=1234	WHERE oid=2345

問合せ結果を仮想的に、各々のエレメントの SGML 属性として持たすことによって、エレメント person および city は、間接的にデータベースに格納された属性および属性値を SGML 属性として持つことになる。このようにして、文書要素とデータベースを融合させることによって以下のことが可能となる。

- 特定の文書要素 (実体) に対する情報を幅広く持つ
- 実体の変化への対応
- データベースの情報を利用した多様な問合せ

SGML 属性に格納されるデータとして、実際にデータベース中のデータを使用するためには、構造化文書に対するアクセスとともに、そのデータベースに対するサポートも可能でなければならない。このような相互操作

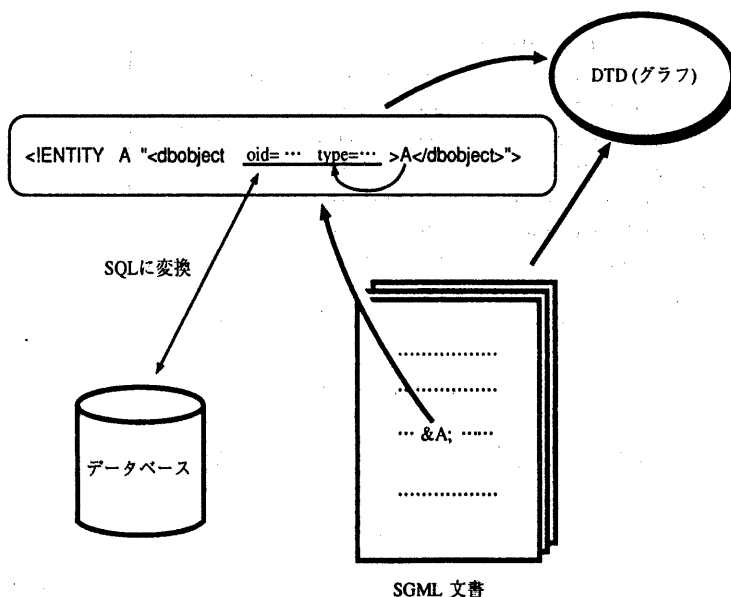


図 2: 文書要素とデータベースの融合 (概念図)

は、なるべく共通のインタフェースで両方にアクセスできることが必要である。本研究では、構造化文書がデータベースと融合されているかどうかに関わらず、同一の方法で構造化文書にアクセスし、その中の文書要素がデータベース情報を利用できるものであれば、更にデータベース中の情報を利用して問合せを行う。

“oid” は文書要素とデータベース中のインスタンスとを結び付けるための SGML 属性であり, “type” は要素の型を特定する SGML 属性である。エレメント “dbobject” は “person” や “city” 等を抽象化したものであって、エレメント “dbobject” を導入することで、あらゆる型に対する文書要素に対しても同様の方法でタグ付けすることが可能となる。

3.2 融合機構の洗練化

3.1 節で述べたように、文書要素とデータベースを融合するためには、エレメントおよび SGML 属性を新たに宣言し、それらのエレメントを文書中に付加しなければならない。ここでは、具体的な融合方式について述べる。

3.2.1 エレメントの抽象化

3.1.2 節に示したように、データベースと融合させる文書要素の型ごとにエレメントを宣言すると、新たに定義するエレメントが増大し、文書が複雑になるだけでなく、それに伴う DTD の変更も大きくなる。そこで、融合させるエレメントを一般化して、次のように定義する。

```
<dbobject oid = 1234 type = person>
    田中 </dbobject>
<dbobject oid = 2345 type = city>
    奈良 </dbobject>
```

3.2.2 エンティティによるエレメント管理

SGML において、エンティティは、長い文や文字列を短い名前で参照したり、別のファイルに蓄えてある文書を取り込むために利用される。一般に、エンティティを使用することで、以下のことが可能になる。

- 文書中で何回も出現する要素の入力が楽になる。
- 宣言されたエンティティ内容を一括して管理することが可能となる。
- 共有して利用されるエンティティについては、それらをまとめて一つのファイルに格納し、パラメータエンティティを使用して複数の DTD に参照されることが可能となる。

そこで、各 dbobject エレメントについて以下のようにエンティティ宣言し、同じ文書要素に対して一括して管理を行う。

```
<!ENTITY tanaka " <dbobject oid = 1234
    type = person> 田中 </dbobject>">
```

マーク付き文書に現れる特定の文書要素には &tanaka; と参照させるだけで済むようになる。

3.2.3 エンティティの生成

データベースとの融合を簡略化するためにエンティティの宣言を行なった。データベース中で異なる実体を表す文書要素に対しては、別のエンティティを宣言しなければならないが、データベース中で同一の実体を表す文書要素に対しても、文書要素の表記が異なれば、エンティティの内容も異なるために同一の実体を表すものについても、その表記法の数だけエンティティを定義しなければならない。

また、エレメント dbobject については、融合によってデータベースの情報を利用することが可能であるが、データベース情報を利用しない通常の SGML 属性を持つる必要がある。

そこで、これらを考慮して、エンティティの定義をできるだけ自動的に行うことを考える。エンティティの生成について以下に述べる。

1. 参照名は、文書中での実際の表記およびデータベースを利用しない SGML 属性の指定の並びを引数とした、文書要素の oid を示すものとする。

$\&ename(A, t_1 = T_1, \dots, t_n = T_n);$

ここで、A は実際の文書に現れる文字列であり、 $t_1, \dots, t_n$ は SGML 属性名の指定で、 $T_1, \dots, T_n$ は SGML 属性値の指定である。また、ename は、この文書要素を一意に識別することが可能な参照名 (oid) である。

2. 上のように定義したエンティティ参照について、その参照名の oid およびその型は、実際のデータベース中で表現されているものであり、(参照名, oid, 型) の組をローカルな辞書に持ち、参照名からその文書要素の oid が特定できる。

辞書

参照名	oid	型
kato	1234	person
⋮	⋮	⋮

3. データベースとの融合のために、エンティティ宣言されるものは、一般に次のような形をとる。

```
<!ENTITY ename " <dbobject oid = ...
type = ... sgm = ... > A </dbobject> ">
```

ここで、sgm は、通常の SGML 属性の指定を表すための、エレメント dbobject の SGML 属性である。

そこで、1. および 2. の情報から、ename, oid の値、type の値、sgm の値 および文書要素 (A) の値の定まった特定のエンティティを作成し、それぞれをファイルに格納する。oid が同じものでも、複数のエンティティが

定義されることがあるので、エンティティ名はシステムで適当に付ける。

例えば、&kato(加藤); という入力に対しては、oid および type の値は、2. から、1234 および person となり、1. から文書要素は“加藤”であることが判断でき、出力として、&kato1; を返す。ただし、&kato1; は、実際には以下のように宣言されている。

```
<!ENTITY kato1 " <dbobject oid = 1234
type = person > 加藤 </dbobject> ">
```

以上のように、データベースと融合させたい文書要素について、ローカルな辞書を用いることで、エンティティを (半) 自動的に作成することができ、生成されたエンティティ一つのファイルに格納することで、複数の DTD から参照されることができ、各マーク付き文書中で共有して利用することが可能となる。また、データベース情報を利用しない、通常の SGML 属性にも対応することが可能となる。

3.2.4 融合に伴う DTD の変更

構造化文書をデータベースと融合させるために、エレメント dbobject を使用した。融合を実現するためには、DTD を次のようにしなければならない。

- データベースとの融合に伴う dbobject エレメントの導入のため、SGML で規定されている“例外”を用いる。
- dbobject エレメントに対する SGML 属性の付加。
- 作成されたエンティティの呼び出し。

データベースとの融合のための DTD の修正は少ないことが望ましい。そこで、データベースとの融合のために使用するエレメント dbobject に関しては、文書中の任意の場所に現れ得るために、包含 (inclusion) を利用して、存在する DTD の最上位のエレメントに付加させなければならない。DTD を直接変更する必要があるのは、この部分だけであり、dbobject のエレメント宣言、SGML 属性宣言およびその他のエンティティ宣言に関しては、どの DTD に対しても共通のものを付加させるだけなので、これらの宣言すべてを一つのファイルにして格納し、各 DTD からパラメータエンティティで呼び出せば良い。

```
<!ELEMENT root --
( ..., ... ) +(dbobject)> (1)
...
```

```
<!ENTITY % sample SYSTEM 'sample'> (2)
%sample;
```

したがって DTD には、以下の二つを行なえば、データベースとの融合のための処理が完了する。

1. dbobject の包含の定義
エレメント定義中の +(dbobject)

2. 融合のための宣言を持つファイルを DTD に呼び出すためのエンティティ宣言

sample ファイルの内容は以下のようである。

- (a) 融合のために使用されるエンティティ宣言
(b) dbobject のエレメント宣言および属性宣言

```
<!ELEMENT dbobject -- #PCDATA -(dbobject)>
<!ATTLIST dbobject oid CDATA #REQUIRED
                  type CDATA #REQUIRED
                  sgm CDATA #IMPLIED>
```

SGML 属性 oid, type の値は必須であるが, sgm の値は任意である。

3.3 スキーマ表示

SGML 属性を利用して文書要素をデータベースと融合させ、データベースに格納されたデータを利用できるようになった。しかし、実際の文章からこれらの文書要素に対して問合せが行われる場合、ユーザはどの文書要素あるいはエレメントがデータベースと融合されているのか、また、どのようなデータベース属性を持つのかを知らなければ、融合させた事を十分に活用できない。そのために、どのような情報が実際の文書に付加されているのか、その情報はどの文書要素に付加されているのか、ということをユーザに知らせる必要がある。

3.3.1 DTD グラフによるスキーマ表示

融合した構造化文書およびデータベースに対するアクセスは、3.1 節でも述べたように、ともに構造化文書を通して行う。構造化文書に対するスキーマについては、4.1 節で述べる DTD グラフが、仮想的にその役割を果たすため DTD グラフを構造化文書に対するスキーマとして表示する。

また、構造化文書に融合されたデータベース情報については、問合せ時に、文書構造も利用するために、DTD グラフと独立に表すのではなく DTD グラフに関連付けて、各エレメントに対してどの関係が付加されたのかを示す。

4 問合せ

4.1 DTD グラフ

DTD 中に保持されている構造情報を利用した問合せの表現および処理のために、DTD に対して、各々のエレメントをノードとして表した有向グラフを作成する。導出された有向グラフを DTD グラフと呼ぶ(図 3)。

DTD グラフにおいて、あるエレメント A から、有向枝を辿って到達できるエレメントを、A の下位のエレメントと定義する。逆に、DTD グラフにおいて、あるエレメント X から、有向枝を遡って到達できるエレメントを X の上位のエレメントと定義する。

DTD グラフには、一つの DTD が与えられると一意に決まり、各々の文書構造とは独立である基本 DTD グラフと、各々の文書構造に依存する文書依存 DTD グラフに大別することができる。これらの DTD グラフを構造化文書に対する仮想スキーマとして、DTD グラフを通して問合せが行われる。

4.1.1 基本 DTD グラフ

基本 DTD グラフは、DTD 中の各々のエレメントをノードとして表した有向グラフであり、それぞれの DTD に対してその構造が一意に決定される。基本 DTD グラフは、文書の基本となる構造(型)を表すものであり、仮想的なデータベーススキーマとみなし、文書に対して問合せを行う。基本 DTD グラフ中の各々のエレメントの識別および出現順序に対応するために、グラフ中のノードに対して以下のようにラベル付けを行う。

1. ルートからの距離が 1 のエレメント群に対して左から順に番号をつける。その番号がエレメントのノード番号となる。
2. ルートからの距離が n ($n \geq 2$) で、同一の親を持つエレメント群に対してそれぞれ左から順に番号を付ける。“ (各エレメントの直接の親のノード番号). (各エレメントに付けられた番号) ” がそのエレメントのノード番号となる。したがって、ルートからの距離が n であるエレメントのノード番号は n 次で表される。
3. 順序が決まらないエレメント群は、同一の番号と各々のエレメントを識別する値を持ち、個々の文書インスタンスのレベルで、ノード番号が確定される。

一般に、DTD グラフはサイクルを含むが、サイクル(再帰)になっているものについては、仮想的に何回も再帰が行われているものとして考える¹。

4.1.2 文書依存 DTD グラフ

DTD が与えられると、基本 DTD グラフは一意に特定できるが、実際の文書インスタンスレベルでは、繰り返しのため、基本 DTD グラフの構造と実際の文書の構造が必ずしも一致しない。そこで、個々の文書に対しても DTD グラフで、問合せに対応することを可能にするため、基本 DTD グラフを型とする文書依存 DTD グラフ(解析木)を文書ごとに作成する。

4.2 問合せ言語

SGML のような構造化文書および 3 章で述べたような、構造化文書に融合されたデータベース情報にアクセスするためには、構造情報を利用したような通常のデータベースシステムの持つ問合せ能力に加えて、以下のような能力が必要である。

¹SGML の規定によりエレメント間の入れ子の深さは有限である。

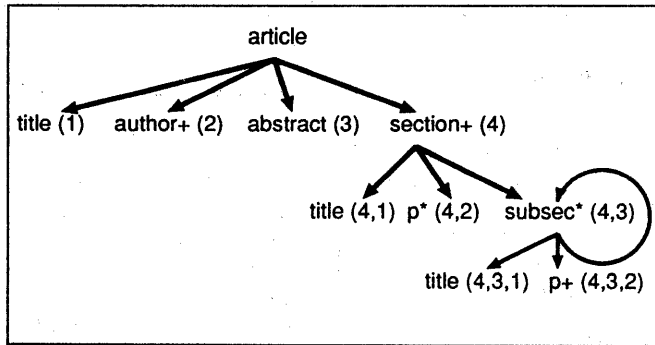


図 3: DTD グラフの例

1. 文書中の要素間の順序関係を扱える。
2. 構造化文書に融合されたデータベース情報に対しても同様の手法で検索を行える。

1. に答えるために、DTD グラフの各々のエレメント (ノード) に対する関数 num() を導入する。2. に関しては、構造化文書がデータベースと融合されていても、そうでなくても、同一の方法で構造化文書にアクセスし、その中の文書要素がデータベース情報を利用できるものであれば、更にデータベース中の情報を利用して問合せを行う。

これらの要求に答えるために、以下の章で実際の問合せについて例を基にして述べる。

4.3 問合せの例

4.3.1 マーク付き文書に対する問合せ

Q 1 アブストラクトの中に 'SGML' と 'HyTime' を含む <article> の <title> および 第一 <author> を見つけよ。

```

SELECT <title>, <author>[1]
FROM <article>
WHERE <abstract> contains
      'SGML' and 'HyTime'
  
```

Q1. のように、問合せにはタグを伴ったエレメント名を明記する。繰り返し記号 ("*" および "+") によって定義されるエレメントについては、繰り返された回数を [] 中に指定することで、繰り返しを伴う問合せも可能となる。また、テキストデータベースのように文字列一致 contains を用いて問合せを行う。ただし、すべての範囲に全文検索を行うのではなく、その検索範囲を <article>.<abstract> の下位に限定している。

Q 2 <abstract> に 'database' を含む <article> の <title> から第一 <section> までと、最終の <section> を示せ。

```

SELECT <title>→<section>[1]
      ,<section>[-1]
FROM <article>
WHERE <abstract> contains 'database'
  
```

Q2. のように SELECT 句にユーザの欲しい文書の特定の範囲を指定することが可能である。問合せ中の A → B は、エレメント A からエレメント B までの内容を示す。また文書中で、A は B よりも前に定義してあるものでなければならない。繰り返しを伴うエレメントに関して、elements[-n] はそのエレメントの最後の繰り返しからの順序を示す。

Q 3 <abstract> に 'HyTime' を含む <article> の <section> について、<subsec> の <title> を見つけよ。

```

SELECT s(<subsec>)+.<title>
FROM <article>a, a.<section> s
WHERE <abstract> contains 'HyTime'
  
```

Q3. では、ObaseSQL[6] に従い、(<subsec>)+ のように正規表現で指定することが可能である。このため、再帰的に定義された構造から任意の深さまでの情報を引き出すことが可能である。

Q 4 <article> に含まれるすべての <title> を見つけよ。

```

SELECT a..<title>
FROM <article> a
  
```

Q4. のように、".." 表記を用いることで、完全な経路を指定しなくても <article> から到達可能なすべての <title> に対して探索が行われる。DTD グラフにおい

ては、到達可能なすべての経路を辿れば良い。

4.3.2 文書の構造に対する問合せ

マーク付き文書に対する問合せは、4.3.1節で考えた。文書の論理構造を利用して、文書に問い合わせることは重要であるが、ここでは、それとともに重要である文書の構造に対する問合せについて考える。

Q 5 <title> に 'SGML' を含む <article> について、そのエレメントだけを取り出せ。

```
SELECT element*(a)
FROM <article> a
WHERE <title> contains 'SGML'
```

Q5. において element*(a) は <article> に関して、文書中に現れるすべてのエレメントを示す。関数 element*() を指定することで実際の文書の内容を除いたタグの部分だけを取り出すことができる。

Q 6 <title> が 'Database Systems' である <article> の <section> の子であるエレメントを出現する順に示せ。

```
SELECT element(s.<*>)
FROM <article> a, a.<section> s
WHERE a.<title> = 'Database Systems'
ORDER BY num(s.<*>)
```

Q6. のように、SGML ではエレメントの出現に関して順序を持つため、順序関係についての問合せが考えられる。順序関係については、(1) 繰り返しを伴うもの(2) 親子関係によるもの(3) 兄弟関係によるものが考えられるが(1) および(2) については前の問合せで考えた。(3). の関係を扱うために関数 num() を導入する。関数 num() は、引数にエレメントを指定し、そのノード番号を返す。エレメントに対してラベル付けを行っているために、同じ親を持つエレメントどうしでは、番号の小さいものほど文書中に出現する順序が早いことになる。

4.3.3 SGML 属性を利用した問合せ

3章で述べたように、構造化文書とデータベースを融合させることによって、データベース情報を利用した多様な問合せが可能となる。

Q 7 "database" を含む <article> の <author> とその年齢を示せ。

```
SELECT <title>, <author>, <author>.person.age
FROM <article>
WHERE <abstract> contains 'database'
```

Q7. では、<author> エレメントに対して、データベース中の情報が付加されているため、エレメントに続く

ドット表記によって、構造化文書中からデータベースに格納されたデータにアクセスすることが可能になる。

Q 8 <article> に現れる大都市を見つけよ。

```
SELECT <title>, a..<*>.city.name
FROM <article> a
WHERE city.population > 500,000
```

Q8. のように、".." 表記を使用することも可能である。SELECT 句では、データベース city に融合された文書中の文書要素を見つけ出すメカニズムが必要となる。WHERE 句については、文書中には、データベース city に融合された文書要素はあらゆるところに存在する可能性があるが、city に関するデータは、ただデータベースに格納されているだけなので、融合されたデータベースにアクセスをすれば良い。

5 おわりに

今後の課題としては、文書間のつながりを考え、ハイパーテキスト的な操作とシステムの実現に関して検討を行う。

謝辞: 日頃から数々の有益な御討論をいただく植村研究室の皆様へ感謝いたします。

参考文献

- [1] G. E. Blake, M. P. Consens, P. Kilpeläinen, P. -Å. Larson, T. Snider, and F. W. Tompa. Text / Relational Database Management Systems: Harmonizing SQL and SGML. In *Proc. of First International Conference on Applications of Databases*, Vol. 819 of *Lecture Notes in Computer Science*, pp. 267-280. Springer-Verlag, June 1994.
- [2] V. Christophides, S. Abiteboul, S. Cluet, and M. Scholl. From Structured Documents to Novel Query Facilities. In *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, pp. 313-324, May 1994.
- [3] ISO 8879 : 1986. *Information Processing - Text and Office System - Standard Generalized Markup Language (SGML)*, Oct. 15 1986.
- [4] JIS X 4151 : 文書記述言語 SGML (Standard Generalized Markup Language), 日本規格協会, 1992.
- [5] Ron Sacks-Davis, Timothy Arnold-Moore, and Justin Zobel. Database Systems for Structured Documents. In *Proc. of the International Symposium on Advanced Database Technologies and Their Integration*, pp. 272-283, October 1994.
- [6] 吉川正俊, 田中克己, 上善恒夫, 田中康暁, 蛭井潤, 堀田光次郎. ObaseSQL: 拡張経路式と継承演算子を持つオブジェクトベース言語. In *Proceedings of Advanced Database Symposium '93*, pp. 63-72, 1993.