データベースシステム 109－52
（1996．7．26）

OODBによるSGML文書データベースの設計

波内 みさ
NEC C&C研究所

文書中に構造情報を持ったSGML文書に関し、これをオブジェクト指向データベース(OODB)を使って格納・管理するために提供するクラス・ライブラリについて、その構成を考察する。

ここでは、文書の構造情報や文書部品としてのテキストの保持方法など、SGML文書をDB中に格納する際に決定すべきいくつかの事項について分析する。特に文書の構造情報については、文書構造のDBクラスへの反映方法に関し3種類の手法を議論し、各方式の有効な使い分けと連携方法について考察する。

A Design of a SGML Document Database System using an OODB

Misa Namiuchi
C&C Research Laboratories, NEC Corporation

We discuss a class library architecture for an Object-Oriented Database (OODB) to store and manage SGML documents which have structure information in themselves.

We examine several ways to store document structure and to hold text fragments. Especially for the former, we compare three ways and discuss their application suitable for each case.

1 はじめに

文書情報の形態が装置やシステムに依存せず、その文書の論理構造が形式的に記述された文書を**構造化文書 (structured documents)** とよぶ。構造化文書は個々のシステムから独立しているだけでなく、その構造が厳密に定義されるため、電子出版、文書データベースの作成・保守、文書の交換流通などに効果的である。このため、企業内・企業間の文書を構造化文書の形式で作成することにより、文書交換、参照、保存を効率化させようとする動きが活発になっている。

構造化文書規格の代表的なものには、**ODA** と **SGML**[1] がある。両者とも ISO の規格として制定されているが、CALS 推進の流れを受けた後者に関し、利用環境全般に渡った支援システムが盛んに発表されている。

構造化文書を活用する上で、それらを格納・管理するデータベース (DB) には次の機能が要求される。

1. 題名、著者、発行日などの書誌情報によって文書を検索できること
2. 利用者による任意 / 特定のキーワードによって、文書を検索できること
3. 文書要素の構成に基づく検索が可能なこと
4. 文書全体のみばかりでなく、文書の構成要素を部品として管理し、それらを再利用するために検索・提供できること

このうち 1,2 は、非構造化文書を含めた文書 DB 一般に要求される機能であるが、3, 4 は、構造化文書の持つ構造情報を利用することによって実現可能な機能である。

これらの機能を実現するために、オブジェクト指向データベース (OODB) を利用した構造化文書 DB システムの研究開発が行われ [2][3]、テキスト本体とその構造情報の格納管理方式に関し、いくつかの方法が提案されている [5]。また、現在では、SGML 文書管理 DB 製品も発表されている。

OODB を利用する場合、OODB の次のような特徴により、上記システム機能 3,4 の効率の良い実現を試みることができる。

- 文書の構成要素をオブジェクトとして管理することにより、部品としての柔軟性を実現
- 集合型やオブジェクト間の関連により、文書の木構造あるいはグラフ構造をそのまま DB のデータ構造として格納可能
- クラス階層と継承を利用し、文書部品の複数の実現方法を提供可能

SGML 規格に則った文書 (SGML 文書) を OODB に格納する場合には、以下の 2 つの課題に関し、方式を決定しなければならない。

1. 文書の構造情報をどういう方式 (クラス構造) で格納・管理するか
2. 各文書の構成要素に対応するテキストをどのように格納するか

本稿では、C++ をデータモデルとする OODB を利用し、そのクラス・ライブラリとして構造化文書 DB の機能を提供するときの、これらの課題へのアプローチを検討する。

2 文書構造の格納方式

2.1 検討を行う 3 種類の格納方式

SGML では、文書型定義 (Document TypeDefinition: DTD) によって文書の構成要素 (文書要素) とそれらの間の関係、すなわち文書構造が規定されている。この DTD に則った実際の SGML 文書 (文書インスタンス) 中では、文書要素の境界をタグで区切ることにより、文書要素とテキストを対応付けている。

SGML 文書インスタンスを OODB に格納する場合、文書の構造情報を DB 構造に反映させるために文書要素ごとにオブジェクトを生成し、そのオブジェクトに文書の属性、対応するテキストなど、文書要素の情報を格納することが一般的である。

ここでは、文書要素を格納するクラス群として OODB 側にどのようなものを用意するかにより、次の 3つの手法を検討する。

(1) 文書要素対応クラスを生成する方法

… 文書要素に対応したクラスを DTD 毎に生成して提供

(2) 文書部品クラスのみを用意する方法

… 文書部品を実現するクラスのみを用意し、各文書要素の情報をその属性値として管理

(3) 文書部品と文書要素対応クラスを併用する方法

… 文書部品クラスとそれを基底クラスとした各文書要素に対応したクラスを生成して提供

```
<!DOCTYPE a_doc [
<!ENTITY   %cont     "figure|para" >
<!ELEMENT  doc        - -  (title, author+, section+) +(footnote)>
<!ELEMENT  title      - -  (#PCDATA)>
<!ELEMENT  author     - -  (#PCDATA)>
<!ELEMENT  section    - -  (title, (%cont;)*, section*)+>
<!ELEMENT  figure     - -  (CDATA, title)>
<!ELEMENT  para       - -  (#PCDATA)>
<!ELEMENT  footnote   - -  (#PCDATA) -(footnote)>
<!ATTLIST  figure     height   NUMBER  #REQUIRED
                      width    NUMBER  #REQUIRED >
]>
```

図 1: DTD の例

次節以降では、これらの方式に対し、スキーマ設計、DB 中のクラス数、文書 DB システム実現の容易さ、クラス・ライブラリとしての運用面、利用面について検討する。比較を行うための例として、以下では図 1の DTD に従う SGML 文書を考える。

2.2 文書要素対応クラスを生成する方法

本方式では、文書要素ごとにクラスを生成し [3]、文書要素の構造および属性は、クラスのメンバとして定義する。この方式により図 1の DTD に対して生成されるクラス構造 (以下、OMT 記法を用いる) とインスタンスのイメージを、図 2に示す。

[1] スキーマ設計

- 基本的に DTD の element 定義に忠実にクラスを定義する
- C++ のクラスでは表現できない文法 ("|", inclusion/exclusion など) を取り扱うために、補助的なクラスやメンバ関数を導入する必要がある

[2] クラスの数

文書要素の数、すなわち DTD 中に定義されたタグの数だけクラスが生成される。また、"|" (or) を実現する方法として、union 型を導入する方法 [2] と、図 2-(a) の「cont」のような基底クラスを導入する方法が提案されているが、後者の場合、その分のクラス数が増える。

[3] システム実現

任意の DTD から DB スキーマを自動生成する処理が複雑で困難。

[4] クラス・ライブラリとしての運用、利用

- 文書要素名 (タグ名) がクラス名に、属性・構造がメンバ変数として実現されるため、直観的な AP 開発が可能
- 各文書要素に特化した操作手続きを定義できるので、データ操作が容易
- 文書部品 (タグ名) を指定した検索時に、クラスの Extent を利用して高速処理が可能
- DTD が変更されると大幅なスキーマ更新が必要な場合がある

[5] 効果的な対象文書の性質

- DTD が変更される可能性のないもの (業務用定型文書や、既出版文書を SGML 文書化したものなど)

2.3 文書部品クラスのみを用意する方法

本方式では、文書要素を部品として管理するクラス (文書部品クラス) を 1 つだけ用意する [4]。attribute(属性) を管理するクラスをこれに追加する場合もある。この方式により図 1の DTD に対して生成されるクラス構造とインスタンスのイメージを、図 3に示す。

[1] スキーマ設計

- 文書部品クラスは、自分の構成要素 (文書部品オブジェクト) を保持する配列と属性オブジェクトの配列から構成される。文書要素名 (タグ名) は、メンバ変数値として格納される。

- DTD ごとに新しくクラスを定義する必要はなく、また、DTD の異なる SGML 文書でも共通に利用可能。
- 各文書要素に特化した操作手続きではなく、検索対象部品および条件を引数とする汎用操作手続きを用意

[2] クラスの数

最低限、"element" クラスのみを用意することによって対応可能。

[3] システム実現

部品クラスとそのメンバ変数にアクセスする汎用検索関数を用意するだけなので、容易。

[4] クラス・ライブラリとしての運用、利用

- DB スキーマは、DTD の変更に影響を受けない
- 要素名、属性、構造を操作手続きの引数として扱わなくてはならないため、AP 開発が直観的にできない
- タグ名による文書部品のカテゴライズが DB データのレベルで行われていないため、特定の部品に対する検索に際し、全件検索を行うか、予めタグ名ごとの集合を生成しておかなければならない
- 文書の構成情報を得るためには、DTD 情報の参照あるいは構成オブジェクトへのアクセスが必要

[5] 効果的な対象文書の性質

- DTD 変更が予想されるもの
- DTD の定める文書規則を守っていない作成途中のドラフト

2.4 文書部品と文書要素対応クラスを併用する方法

ここで、2.2と 2.3の方法の併用を考え、文書構造を格納する部品クラスと、それを基底クラスとして各文書要素に対応したクラスを生成する手法を提案する。この方式により図1の DTD に対して生成されるクラス構造とインスタンスのイメージを、図 4に示す。

本方式では、文書構成は基底クラスの部品クラスにおいてオブジェクト・ポインタ配列として実現する。その派生クラスである文書要素対応クラスには、各文書構造名 (タグ名) を持つメンバ関数を用意し、それを使って構成要素にアクセスできるようにする。この「文書要素対応オブジェクト」として文書部品を扱うことにより、DB レベルでの文書要素の管理と、文書構造を意識した AP 開発が可能となる。またその一方で、「部品オブジェクト」として DTD の制約を受けずに自由に文書を格納することが可能となり、文書部品を 2 つの性質を持つオブジェクトとして扱うことができる。

[1] スキーマ設計

- DTD ごとに新しいクラス群が生成されるが、新規に定義するメンバ変数はタグの attribute(属性) 程度
- 構成要素配列に対して、文書要素名でアクセスするためのメンバ関数定義が必要

[2] クラスの数

文書要素の数だけクラスが生成されるが、"|" を表現するための補助的なクラスは不要。

[3] システム実現

クラス構成は単純だが、構成要素配列にアクセスするためのメンバ関数定義が繁雑。

[4] クラス・ライブラリとしての運用、利用

- 文書要素の追加 / 削除などによる DTD 変更の場合は DB スキーマの更新が必要だが、文書要素出現の順序、回数の変更などに関しては、スキーマ変更の必要はない
- タグ名による文書部品のカテゴライズが実現されるので、部品単位の検索が高速
- 文書要素対応クラスからの構成要素アクセスがメンバ関数を介し間接的になるため、処理効率が 2.2節の方式に比べて落ちる

[5] 効果的な対象文書の性質

2.3節に同じ。

2.5 考察

以上の 3 種類の方法をシステム構築面、利用・運用面から比較すると、それぞれが効果的な適用対象文書が異なることがわかる。

社内業務用帳票など、各文書要素の形式が明確に規定され、それが変更される可能性が低い場合に

は、2.2節の「文書要素対応クラス」を利用することにより、効率の良いAP開発や高速アクセスが可能となる。また、確定途上のDTDを使った文書や、作成途中の文書などをDBに格納したい場合には、2.3節の「文書部品クラス」を使って、構造を柔軟に変更可能とすることができる。さらに、2.4節の「文書部品」オブジェクトと「文書要素対応」オブジェクトを併用する方式により、柔軟な部品管理に加え、「文書要素対応クラス」を一種のビューとして用いた直観的なAP開発を行うことが可能となる。

2.4節の方式により、利用面に関してはほぼすべてのSGML文書を対象とできると考えられるが、DTDに合致していない文書要素を「文書要素対応クラス」から扱う方法や、十分な性能を達成するシステムの構築には、さらに検討が必要である。

3 SGML文書格納のためのその他の課題

SGML文書をDBに格納する際に決定しなければならない事項の一つに、テキスト本体の格納方法がある。文書全体のみを格納すると、ある文書部品に対応するテキストを、ある場合には巨大なテキスト中から探索して切り出さなくてはならず、「部品」として柔軟に利用することが困難である。このため、何らかの方法で文書要素とそれに対応するテキストとの対応付けが必要である。

まず、最小文書要素に対応するテキストのみをオブジェクトに格納する方法がある。この場合、元のテキストのサイズと同じ記憶容量で格納できるが、文書を最小単位よりも大きな単位(特定の章、節など)で取り出そうとすると、多くの最小部品を再合成する必要があるため、一般に非常に処理コストがかかる。また、文書部品すべてに対応するテキストを持たせる方法が考えられるが、この場合は、上位部品が下位部品のテキストを重複して持つため、元のテキスト・サイズを遥かに越えた記憶容量が必要となる。

テキストの格納方法は、このように、部品としての利用可能性と記憶容量のトレード・オフになるため、システムで一意に決定することは困難である。

また、SGML文書中のタグ中に出現するattribute(属性)の値をDBに格納する際に、スキーマ上でその型を決められないという問題がある。これは、DTDで指定される属性のデータ型が、文書中に記される文字の種類を中心に規定されているため、特にそれが数値の場合、桁数、精度が不明であるためである。DBスキーマを最適に生成するためには、DTDの持つ意味を知っている利用者に各属性の値に関する情報を要求しなければならない。

このように、あるSGML文書を格納するために最適なスキーマを構築するためには、利用者との相互情報交換が必要であり、文書管理DB側には扱う文書によって可変な要因を柔軟に受け入れる枠組を用意する必要がある。

4 おわりに

本稿では、OODBを使ってSGML文書を格納・管理するDBシステムを構築する際に提供するクラス・ライブラリついて検討した。

文書の構造情報を管理するスキーマ生成の複雑さとAP開発、運用の容易さはトレード・オフの関係にあり、格納対象のSGML文書の性質によって、適/不適がある。また、文書要素の属性として実際に入力される値やテキストの分解・格納の単位など、システムだけで一意に解決できない課題も多い。SGML文書DBを構築する際には、このような点を考慮して、多分に利用者との相互情報交換が可能なものを作る必要があると思われる。

なお、本稿で検討した文書構造の格納方法の特徴を検証するために、現在、弊社のOODBMS "PERCIO" 上に、プロトタイプを作成している。本プロトタイプの評価を足掛かりとし、実運用システムの設計を行う予定である。

[謝辞]

本研究に関し、貴重な御助言を下さったNEC C&C研究所 鶴岡 邦敏 担当部長に深謝いたします。

参考文献

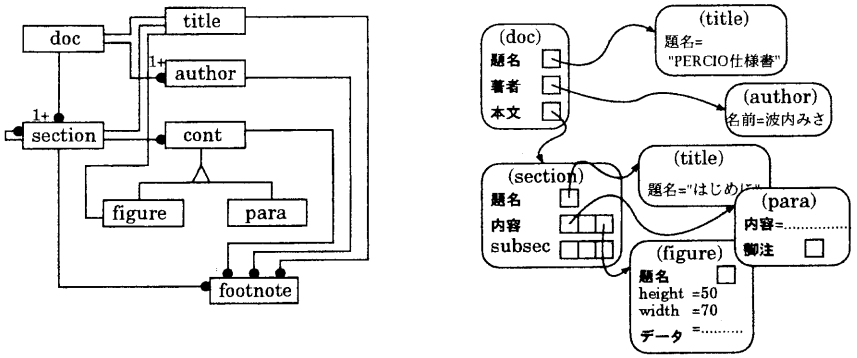
[1] ISO 8879, "1986. Information Processing - Text and Office System - Standard Generalized Markup Language (SGML)," 1986.

[2] Christophides, V., Abiteboul, S., Cluet, S. and Scholl, M., "From Structured Documents to Novel Query Facilities," Proc. on SIGMOD Intl. Conf. on the Management of Data, pp. 313-324, 1994.

[3] Sacks-Davis, R., Arnold-Moore, T. and Zobel, J., "Database Systems for Structured Documents," Proc. of 1st Intl. Conf. on the Application of Database Technologies & their Integration, 1994.

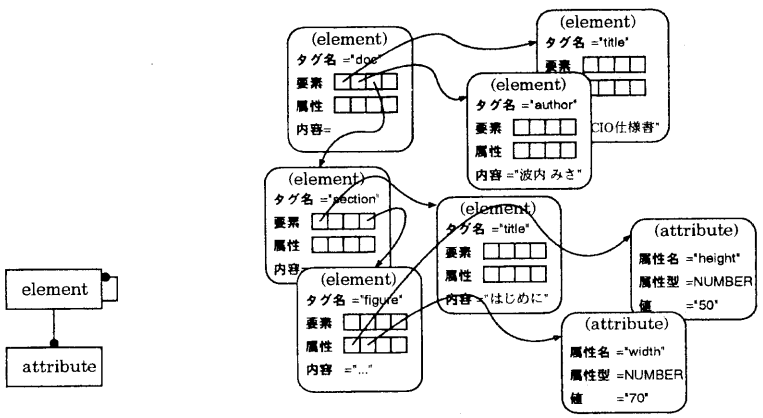
[4] 岩崎 雅二郎, 「OODBによる構造化文書データベースの実装」, DBS研究会会報 94-DBS-99, Vol. 94, No. 62, 1994.

[5] 吉川 正俊, 「構造化文書とデータベース」, Proceedings of Advanced Database System Symposium '95, pp. 49-57, 情報処理学会, 1995.



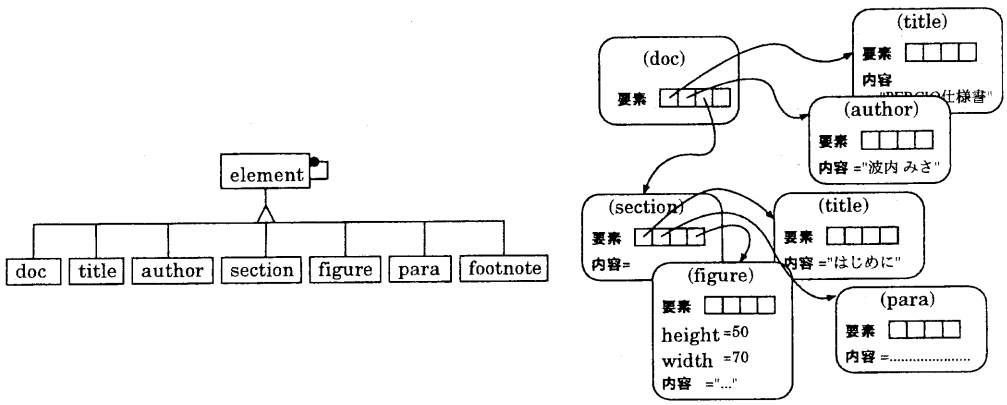
(a) クラス図　　(b) インスタンス例

図 2: 文書要素に対応したクラス



(a) クラス図　　(b) インスタンス例

図 3: 文書部品を表現するクラス



(a) クラス図　　(b) インスタンス例

図 4: 文書部品と文書要素対応クラスの併用