データベースシステム 114－13
（1998. 1. 20）

オブジェクト指向モデリングによる文書部品管理方式

波内 みさ
NEC C&Cメディア研究所
nami@ccm.cl.nec.co.jp

SGMLなどの構造化文書を格納・管理する構造化文書データベースは、構造化文書自身の書誌情報とテキスト本体の他、文書中に示される論理構造情報を管理することによって、文書を構成する文書要素を文書部品として利用することが可能となる。一つの文書部品が管理する文書要素の数や文書部品を操作するための機能は、アプリケーションや利用する局面によって異なるため、構造化文書データベースは、それ自身が文書部品に関して必要な機能を柔軟に取り込む枠組みを持つことが望ましい。

本稿では、オブジェクト指向モデリングを利用し、部品オブジェクトの生成・操作機能に関する拡張性を実現するフレームワークを持つ構造化文書データベース・スキーマを提案する。

An Object-Oriented Modeling for Document Parts Management

Misa Namiuchi
C&C Media Research Laboratories, NEC Corporation

A structured-document database that stores and manages structured documents such as SGML, enables users to utilize document-composing elements as document parts by managing not only document bibliographic information and their contents but logical structure information of them. As number of elements which a document part will manage and functions for managing the part are different between applications or situations, a structured-document database is expected to have flexibility that can extend its functionality necessary for document parts management.

In this paper, the author proposes a schema for structured-document databases that have a framework to extend functionality of creation and management of parts objects, using object-oriented modeling.

1 はじめに

構造化文書の規格の一つであるSGML (Standard Generalized Markup Language)[1] は、文書データの交換を容易にすることを目的として、文書処理系固有の情報を含んだレイアウト情報を取り除き、文書の論理構造をテキストとして明示するものである。1986年のISO化 (ISO 8879) に続き、日本でも1992年にJIS化 (JIS X 4151) されている。現在、WWWで広く利用されているHTML (HyperText Markup Language) 、および、HTMLに代わる言語として注目されているXML (eXtensible Markup Language)[17] も、SGMLの一つのアプリケーションである。

構造化文書を格納・管理し、共有・流通させるデータベースを、構造化文書データベース (DB) とよぶ。構造化文書DBは、書誌情報検索、全文検索機能などの通常の文書DBの機能に加え、構造化文書の持つ文書の論理構造情報を利用した検索機能を特長とする [16]。さらに、構造化文書を構成する個々の文書要素 (element) あるいはその集合を文書部品として管理することにより、これらを参照・編集操作やテキスト再利用の単位として利用することができる。

一つの文書部品が管理する文書要素の数や文書部品を操作するための機能は、アプリケーションや利用する局面によって異なるため、構造化文書DBは、それ自身が文書部品に関して必要な機能を柔軟に取り込む枠組みを持つことが望ましい。

我々は、特にSGML文書に関し、その様々な利用方法に柔軟に対応するために、システムに要求される機能を容易に取り込むことが可能なSGML文書DBの構築を検討している。本稿では、SGML規格に則る文書およびそれを構成する文書部品を格納・管理し、ユーザが自由に文書部品化単位を指定したり、特定の部品操作処理を追加することが可能なSGML文書管理フレームワークを提案する。

2 拡張可能なSGML文書DBの要件

2.1 構造化文書の管理方式

構造化文書を管理する場合、文書自身の情報である書誌情報（題名、著者、発行年月日、管理部門など）の他、テキスト・データ（構造化文書本体）そして構造化文書が持つ論理構造 (文書要素) が管理対象となる。構造化文書DBを構築する際には、このような情報をどのように管理し、それぞれに対してどのような機能をサービスするかを決定する必要がある。

構造化文書DBにおける文書本体と論理構造情報、そして文書部品の管理方法に関しては、現在までに以下のような研究あるいは選択が行われている [6]。

(1) テキスト本体の管理

1. DB外のファイル・システムに管理。DBでは文書id, ファイル名などを管理する。DBインタフェースから文書内容の参照はできない。
2. DB中の一データとして管理。テキスト操作はDBシステムの文字列操作機能を利用。
3. DBと既存の文書検索専用システムとを緩やかに統合し、高度な文字列検索機能をDBインタフェースから利用 [15][4]。

(2) 論理構造情報の管理

1. 論理構造をDBスキーマに直接反映 [3][12]。
2. 論理構造を管理するための汎用型 (クラス) をDBスキーマに用意し、構造情報はデータ (インスタンス) に埋め込む。
3. 論理構造を管理する抽象データ型を導入。DBスキーマに論理構造は現れない [2]。

(3) 文書部品の管理

1. 論理構造に従って抽出された文書要素を一つの文書部品単位として管理。
2. 論理構造に従って抽出された文書要素のうち、指定されたものを文書部品として管理。
3. 文書要素を部品化せず、要求の都度、取り出す。

このような管理手法を実装する場合、ベースとなるデータ管理システムに何を選択するかにより、基盤の実装モデルと機能の特徴に相違が現れる。現在までに提案されているDBシステムをベースとしたシステムの特徴を以下に示す。

- リレーショナル・データベース (RDB) を利用

  … 個々の文書要素をフィールド値として管理する例は少なく、文書全体を一フィールドあるいは外部の文書検索システムに格納し、SQLインタフェースによりアクセス可能なシステムを構築。

- オブジェクト指向データベース (OODB) を利用

  … 文書要素をオブジェクトで管理し、それを文書部品として利用。文書検索システムと連携する研究 [4] もある。

- 独自モデルによる DB を利用

  … 管理対象をマルチメディア・データ全体とするなど、実現目標とする機能仕様が大きい。

本稿では特に、SGML 文書を文書部品の集合として積極的に利用することを目的とし、OODB の利用を前提として考える。

2.2 OODB を利用した SGML 文書 DB

OODB を利用して構造化文書 DB を構築するとき、その文書要素をオブジェクトとして管理することにより、文書インスタンス毎に異なる文書の論理構造を容易に管理することができる。また、文書要素を単位としたテキスト参照、編集が容易となる上、文書部品としての文書全体 / 文書要素の再利用を促進することができる。

このような利点により、OODB をベースとした SGML 文書 DB は現在までに多数提案され [7][8][9]、またいくつかの製品としても実現されている [18][19]。

OODB に文書要素を格納する手法としては、以下の 2 種類の方法がある。

(1) DTD (Document Type Definition) の各 element 宣言に対応したクラスを定義し、element 構造はクラス構造 (メンバ変数) として定義

(2) element 情報を格納する汎用クラスを定義し、element 構造は属性値としてインスタンス・レベルで管理

しかし、単に論理構造を格納するクラス群を定義しただけでは、システム構築時に以下のような課題が残ることがある。

- 部品化する文書要素を自由に選択できない。DTD 単位の部品化指定は可能でも、文書インスタンス毎の指定変更は困難。
- 部品を管理するクラス (部品クラス) を処理モジュール (プログラム) 中にハード・コーディングすると、ユーザが部品クラスをカスタマイズした派生クラスを定義した場合、既存処理モジュールがその派生クラスを処理対象にできない。
- 文書インスタンス毎、あるいは、文書要素毎に動的に操作を変更することができない。

つまり、柔軟な SGML 文書 DB を構築するためには、オブジェクトの生成とその振る舞いに関し、ユーザの要求機能を柔軟に取り込めるようなクラス構成 (スキーマ) を考慮することが重要である。

以下の節では、このような機能拡張の柔軟性を持った SGML 文書 DB スキーマについて議論する。

3 オブジェクト指向フレームワークを用いたスキーマ設計

本節では、OODB を利用して SGML 文書 DB を構築する場合の前節で示したような課題に対し、オブジェクト指向フレームワークを利用した DB スキーマによる解決法を提案する。本フレームワークでは、システムの機能の拡張性、柔軟性、再利用性を高めるデザイン・パターン [5] を利用する。

以下では、下記の機能項目ごとに、目的とする機能の詳細、設計したクラス構造、それによって得られた利点・欠点、問題点・課題について議論する。

[1] SGML 文書管理の基本クラス

… 文書要素、テキストの同時管理。

[2] 文書部品機能の拡張

…文書部品の持つ情報、機能を変更。任意の文書部品管理用クラスの生成。文書部品生成時のユーザ処理追加。

[3] 動的な格納方法指定

…文書の格納方法の動的変更など。

なお、以下に示すクラス構造は、OMT 記法 [11] に類似した記法による。

3.1 SGML 文書管理の基本クラス

ここでは、書誌情報と文書本体、および、論理構造情報を保持する SGML 文書管理の基本クラスを設計する。

3.1.1 目的機能

以下の機能を有すクラス群を考える。

(a) 書誌情報の管理とテキスト、文書要素の管理機能が独立に変更・拡張可能

(b) 文書要素の包含関係を表現する文書要素管理機能

(c) 各文書要素に対応したテキストを保持可能

(d) 部品として管理する文書要素の数を動的に変更可能

3.1.2 クラス構造

基本クラスの構造を図1に、インスタンス例を図2に示す。StructuredDoc クラスが書誌情報を、Contents クラスとその派生クラスが文書の構造情報、テキストを管理する。これにより、書誌情報管理と文書本体の管理構造が分離されるため、それぞれの派生クラスを定義することにより、独立に機能拡張が可能となる。

StructuredDoc クラスは、書誌情報を管理する他、文書本体にアクセスするインタフェースを提供し、文書構造の詳細を隠蔽する。すなわち、Contents クラスの派生クラスに対する wrapper クラスとして機能する。

Contents クラスは文書要素を管理する抽象クラスで、文書部品として利用する単位となる。基本的な派生クラスとして、子オブジェクトの集合により文書要素の包含関係を表現する Element クラスと、それ以上包含要素を持たないテキストを管理する Term クラスを持つ。Element クラスは、部品として有用な文書要素のみを選択的に子オブジェクトとして保持し、必要な場合には部品化していない包含文書要素を動的に分解して利用する。

これらのクラスを基底クラスとして、文書部品の基本機能を持ったユーザ定義クラスを導入することができる。図1には、例として、ある文書要素のテキストに対応する DB データとのリンクを保持するようなユーザ定義クラス *LinkedElem* を示している。

なお、本クラス構造は、部分 - 全体階層を表現するための典型的なデザイン・パターンである Composite パターンを形成している。

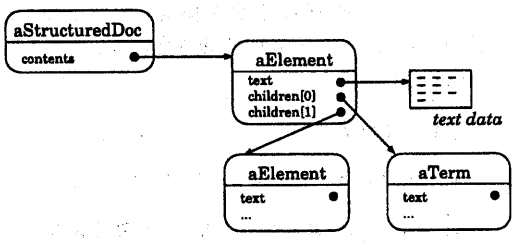

図 2: SGML 文書管理インスタンス例

3.1.3 クラス構造の利点・欠点、問題点・課題

本クラス構造により、新たな Contents クラスを容易に追加でき、その構成、構造を StructuredDoc により隠蔽することができるが、その一方で以下の様な課題が生じる。

- StructuredDoc として生成されるオブジェクトの構成、構造を制御することが困難になる。動的にオブジェクト生成処理を制御する仕組みが必要。

本課題については、以下で議論するオブジェクトの生成方法で対応方法を検討する。

3.2 文書部品機能の拡張

文書部品の持つ情報と機能は、前述の Contents クラスあるいは Element, Term クラスを基底クラスとした派生クラスを定義することにより、拡張することができる。

これらをはじめとする任意の文書部品の生成処理は、まず、StructuredDoc クラスあるいは Contents クラスの各派生クラスのメンバ関数として実装する方法が考えられる。この場合、どの Contents クラスの派生クラスを生成してどの文書要素を格納するかという格納規則が予め決まっている場合には、それをメンバ関数として実装することは容易だが、新しい格納規則を追加したい場合には、既存のこれらのクラスに新しいメンバ関数定義を追加して、クラス定義を更新しなければならない。また、ユーザが独自の Contents クラスの派生クラスを定義した場合にも、既存クラスのインスタンス生成方法と同じ方法でオブジェクト生成を行いたい。

3.2.1 目的機能

ここでは、以下の機能を有すクラス群を考える。

(a) 格納処理の追加、およびユーザによるカスタマイズを可能とし、既存クラス構造への影響を最小にする

(b) 格納処理のインタフェースを共通化

3.2.2 クラス構造

文書部品生成のためのクラス構造を図3に示す。DocFactory クラスは、SGML 文書を格納するためのインタフェースとしてのメンバ関数 Store() を持つ。さらに、Contents オブジェクトを生成するためのメンバ関数 CreateConts() と、StructuredDoc を生成して Contents オブジェクトをセットす

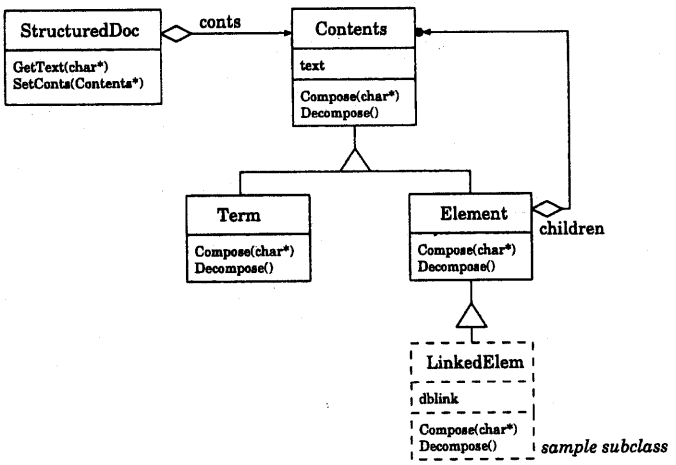

図 1: 基本クラス構造

るメンバ関数 CreateDoc() を持ち、Store() の中からこれらを call する。

実際に生成する StructuredDoc および Contents オブジェクトは、DocFactory クラスの派生クラスで定義される。例えば、TreeFactory クラスでは、すべての文書要素に対して Element オブジェクトを生成する CreateConts() を持ち、DocFactory の CreateConts() をオーバーライドする。一方、LinkedElemFactory クラスはユーザ定義クラスの例で、その CreateConts() は 3.1.2節の LinkedElem オブジェクトを生成する。またここで同時に、テキスト内容による他 DB データ更新など、生成する部品に関連した独自ユーザ処理を追加することもできる。

StructuredDoc オブジェクトも、メンバ関数 CreateDoc() を DocFactory の派生クラスで再定義することによって、生成する派生クラスを変更することが可能である。

なお、本クラス構造は、オブジェクト生成に関する Abstract Factory パターン、Factory Method パターン、Template Method パターンを形成している。

3.2.3 クラス構造の利点・欠点、課題・問題点

本クラス構造は、文書部品を生成するための文書要素の格納規則を変更・拡張したい場合に有用であるが、ユーザが利用する際には以下の点が重要である。

- フレームワークの典型的な作法「我々を呼び出すな。必要なときに我々が呼ぶ。」[13] を利用する。このため、処理を拡張する際、ユーザがフレームワークについて確実に理解していなければならない。

3.3 動的な格納方法指定

3.2節で検討したクラス構造は、文書要素の格納規則を静的に変更・拡張する場合に有効である。これに対し、ここでは実行時に動的に格納方法を変更するためのクラス構造を考える。これは、ユーザがアプリケーションと対話しながら文書要素の格納方法を変更する場合などに利用できる。

3.3.1 目的機能

ここでは、以下の機能を有すクラス群を考える。

(a) 動的に文書要素の格納方法を変更可能

(b) クラス構造への影響を最小限にして新しい格納方法を導入

3.3.2 クラス構造

文書部品を動的に生成するためのクラス構造を図 4に、オブジェクト間の協調関係を示すインタラクション・ダイアグラムを図 5に示す。まず、3.1節で導入した StructuredDoc クラスにメンバ関数 Store() を設け、引数で与えられた文書 (テキスト) を格納するものとする。

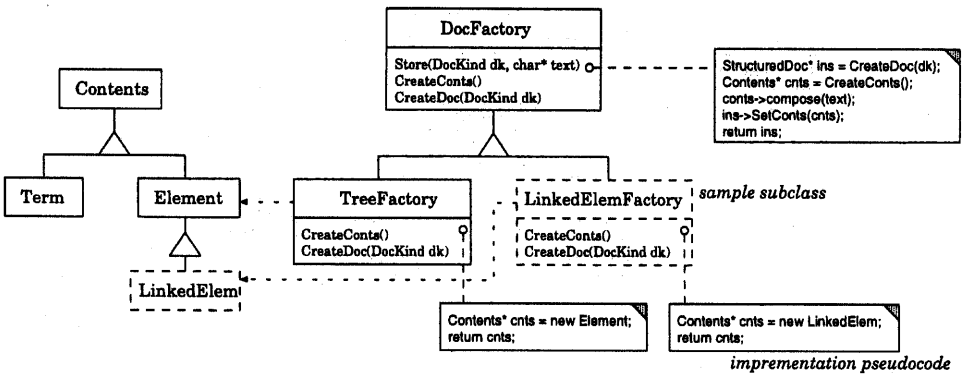

図 3: DocFactory による文書部品生成

Contents クラスには、格納処理を実現するオブジェクトとその処理対象となるテキストを引数に取るメンバ関数 Accept() を設定する。Accept() 内では、引数で受け取った文書要素格納用オブジェクトに対し、自身を引数として格納処理を行うメンバ関数を起動する。

図 4では、3.2節で導入した DocFactory を文書要素格納用オブジェクトとし、その各派生クラスに、それぞれのクラスの格納方針に従って Element オブジェクト、Term オブジェクトを生成するメンバ関数 CreateElem(), CreateTerm() を設定している。Contents クラスの派生クラスでは、その Accept() 内で対応する DocFactory のメンバ関数を呼び出し、それぞれの格納方針に応じたオブジェクトを得ることができる。

Contents::Store() と DocFactory::CreateElem(), CreateTerm() に、Accept() に与える DocFactory オブジェクトの指定を GUI と連携して動的に取得する処理 (図 5の GetFactory 関数など) を入れることにより、各文書要素毎にユーザが動的に DocFactory オブジェクトを指定することが可能となる。

新しい格納方法を導入する場合には DocFactory クラスの派生クラスが新しく定義されるが、これが既存の StructuredDoc, Contents オブジェクトに影響を与えることはない。

なお、本クラス構造は、処理を加えるオブジェクトのクラスに変更を加えずに、新しい処理を実現する Visitor パターンを形成している。

3.3.3 クラス構造の利点・欠点、問題点・課題

本クラス構造を利用して動的な格納処理変更を行う場合にも、ユーザがフレームワークについて確実に理解していなければならない。

3.4 その他の機能の実現

以上で議論したクラス構造に関し、これらと連携して実現可能な機能とその実現方法について、以下に簡単に示す。

- テキストの格納場所変更

  格納場所を表現する抽象クラスと、DB 内格納、ファイル格納など実際の格納場所に対応する派生クラスを定義。これを Contents オブジェクトのメンバ変数 text からポイントし、格納したい場所に応じて適宜変更して利用。

- テキストの内容変更を他の DB データに通知

  トリガの働きをするクラスを定義し、そのインスタンスを Contents オブジェクトとそれに対応する DB オブジェクトとに関連付ける。

- SGML を HTML/XML に変換

  変換のためのインタフェースを持つクラスと、実際の変換規則と変換処理を持つ派生クラスを設定し、StructuredDoc クラスに関連付ける。

4 おわりに

本稿では、SGML 文書の管理方式を柔軟に拡張し、様々な機能を持つ文書部品を利用可能とすることを目的として、オブジェクト指向フレームワークを用いた構造化文書 DB スキーマを提案した。

本スキーマを利用することによって、文書の書誌情報とテキスト本体、文書要素を同時に管理し、か

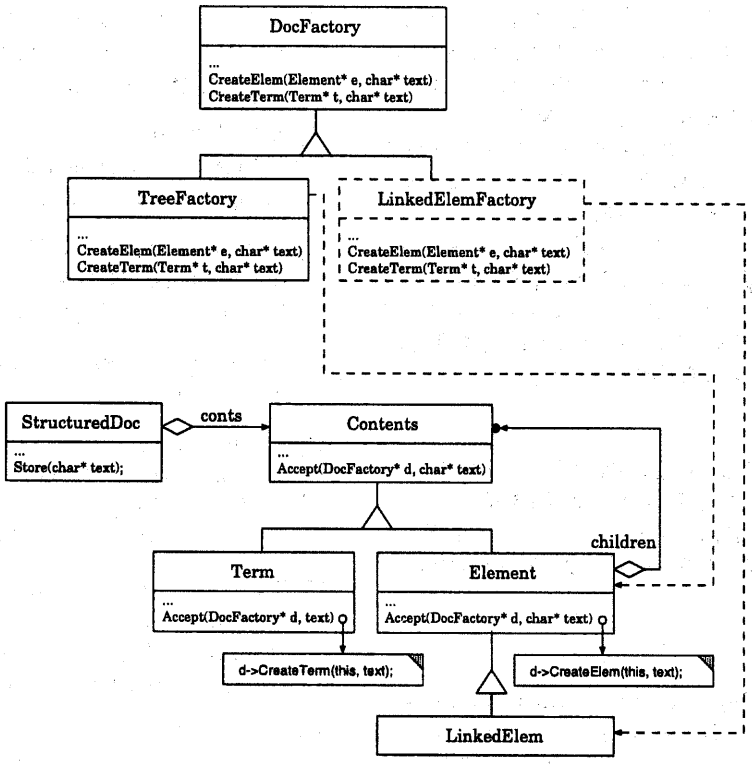

図 4: 文書部品の動的生成

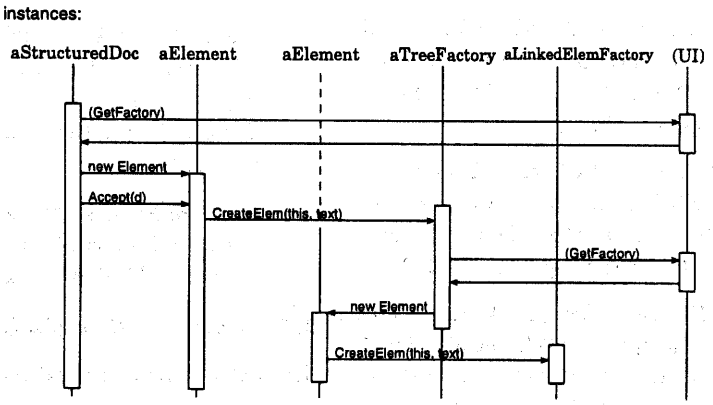

図 5: 文書部品の動的生成におけるオブジェクトの相互作用

つ、文書部品の持つ機能の拡張、文書部品生成時のユーザ処理追加、文書の格納方法の動的変更などの機能を実現することができる。例えば、ここで提案したクラス構造をHTML/XML管理に適用することによって、それぞれの文書に対して特有の処理をユーザが自由に組み込み、柔軟なシステムを構築するために利用することが可能である。

今後は、本スキーマ上で実際にクラス、処理の追加を行った場合にどのような波及効果あるいはトレードオフがあるかを検証する予定である。また同時に、弊社OODB PERCIO[14][10]に実装し、拡張性と性能などについて定量的評価を行いたい。

[謝辞]

本研究に関し、貴重な御助言を下さったNEC C&Cメディア研究所 鶴岡 邦敏研究部長、および、オブジェクト指向フレームワークについて数多くの示唆を与えてくださった同研究所 佐伯 剛幸氏に深謝いたします。

参考文献

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

[2] Blake, G.E., et al., "Text/Relational Database Management Systems: Harmonizing SQL and SGML," Proc. of Intl. Conf. on Applications of Database (Lecture Notes in Computer Science No. 819), pp. 267-280, 1994.

[3] Christophides, V., et al., "From Structured Documents to Novel Query Facilities," Proc. on SIGMOD Intl. Conf. on the Management of Data, pp. 313-324, 1994.

[4] Croft, W.B., et al. "A Loosely-Coupled Integration of a Text Retrieval System and an Object-Oriented Database System," Proc. of ACM SIGIR Conf., pp. 223-231, 1992.

[5] Gamma, E., et al., "Design Patterns," Addison-Wesley, 1995. (邦訳: 本位田, et al., 「オブジェクト指向における再利用のためのデザインパターン」, ソフトバンク, 1995.)

[6] 加藤, 吉川、「データベースオブジェクトリンクを制約として有する構造化文書ビューの構築」, Proc. of Advanced Database System Symposium '97, pp. -, 情報処理学会, Dec. 1997.

[7] 中津山, et al., 「文書データベース管理システム Xebec の概要」, 情報処理学会 DBS 研究会、95-DBS-101, pp. 97-104, 1995.

[8] 波内, 「OODB による SGML 文書データベースの設計」, 情報処理学会 DBS 研究会、109-52, pp. 311-316, July 1996.

[9] 波内, 鶴岡, 「SGML/HTML 文書 DB におけるテキスト格納方式の提案」, 情報処理学会第54回全国大会, 3-205, March 1997.

[10] http://www.ace.comp.nec.co.jp/product/db/percio/p_main.htm

[11] Rumbaugh, J., et al., "Object-Oriented Modeling and Design," Prentice-Hall, 1991. (邦訳: 羽生田, 「オブジェクト指向方法論 OMT - モデル化と設計 -」, トッパン, 1992.)

[12] Sacks-Davis, et al., "Database Systems for Structured Documents," Proc. of 1st Intl. Conf. on the Application of Database Technologies & their Integration, 1994.

[13] Sweet, R. E., "The Mesa programming environment," SIGPLAN Notice, 20(7), pp.216-229, July 1985.

[14] 鶴岡, et al., 「オブジェクト指向データベース管理システム PERCIO の開発と今後の課題」, 電子情報通信学会論文誌 D-I, Vol.J79-D-I No.10, pp.587-596, Oct. 1996.

[15] Yan, T.W. and Annevelink, J., "Integrating a Structured-Text Retrieval System with an Object-Oriented Database System," Proc. of the 20th Intl. Conf. on VLDB, pp. 740-749, 1994.

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

[17] http://www.w3.org/XML/Activity

[18] http://www.chrystal.com/ (Astoria)

[19] http://www.inso.com/ (DynaBase)