NAIST-IS-DT9661006

博士論文

構造化文書データベースの
モデルと問合せに関する研究

加藤 弘之

1999年2月8日

奈良先端科学技術大学院大学
情報科学研究科 情報システム学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
博士(工学) 授与の要件として提出した博士論文である。

加藤 弘之

審査委員：　植村 俊亮 教授
伊藤 実 教授
松本 裕治 教授
吉川 正俊 助教授

構造化文書データベースの モデルと問合せに関する研究*

加藤 弘之

内容梗概

近年，データベース管理システムはその管理対象を拡大しており，構造化文書もその中に含まれつつある．構造化文書をデータベースで管理することにより，文書の作成，編集，検索といった文書処理のさまざまな側面に，同時実行制御，アクセス制御，障害回復といったデータベース機能を適用することができるようになる．

本研究では，構造化文書の論理構造に基づく統一的な操作を可能とするために，抽象データ型アプローチによりデータベースで構造化文書を管理する．本論文では，この管理手法のもとで次の二つの研究成果を報告する．一つは，構造化文書をデータベースビューとしてみなすための定義手法とこのようなビューに対する問合せ手法を提案する．もう一つは，構造化文書をデータベースで管理するための新しい概念モデルを提案する．これらの研究成果によって，データベースに格納されている構造化文書から，希望する内容と論理構造を有する構造化文書を検索することが可能になった．

伝統的なデータベース同様，論理的データ独立性維持のため応用に対して，構造化文書をデータベースビューとして定義できることが望ましい．よって，この構造化文書ビューの定義手法を提案する．また，ビューに対する問合せ処理について，伝統的な最適化手法が構造化文書固有の選択条件に対してどのように適用されるかを論じる．

*奈良先端科学技術大学院大学 情報科学研究科 情報システム学専攻 博士論文, NAIST-IS-DT9661006, 1999年2月8日.

更に，文書中の文字列とその文字列が表すデータベース中の意味データ間に構築されるオブジェクトレベルのリンク機構を有する構造化文書データベースモデルと，そのようなデータベースに対する問合せについて提案する．このリンク機構を有する文書の概念モデルは，表記文字列が保持される表記層と，表記層における部分文字列が表す意味データが保持される参照層の二つの層から構成されている．このリンク機構により，本研究で提案する概念モデルを管理する構造化文書データベースに対して，文書の文字列のみならずその意味に基づく問合せが可能となった．

キーワード

構造化文書，データベース，問合せ，抽象データ型

Studies on Model and Queries for Structured Document Databases*

Hiroyuki Kato

Abstract

Recently, Database Management Systems(DBMSs) have extended types of values, including structured documents, under their responsibility. Managing structured documents in DBMSs yields that database functionality, such as concurrency control, access control and recovery, can be applied to such document processing as authoring, editing and retrieving structured documents.

In this research, structured documents are managed in databases with the Abstract Data Types(ADTs) approarch to handle them uniformly based on their logical structure. Under this method of management for structured documents, in this dissertation, we report following two main contribution. We propose, first, methods how to define structured documents as database views and how to query such views. Secondly, we propose a novel model for structured documents in databases. By contribution of this research, it becomes possible to retrive structured documents with desired contents and logical structure.

As with traditional databases, it is desirable that structured documents can be defined as database views to keep logical data independence. We will develop how to define such structured document views. Also, we propose a method that how to apply traditional optimization techniques to selection condition in paticular case of retrieving structured documents.

*Doctor's Thesis, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DT9661006, February 8, 1999.

Moreover, we propose a novel conceptual model of structured documents with links between character strings in documents and corresponding database values. We also propose query operators on the conceptual model in databases. This model is logically viewed as consisting of two parallel layers; on the "appearance" layer, ordinary text (i.e. a linear sequence of character strings) is placed, while the "reference" layer holds an array of OIDs and literals. Each occurrence of OIDs or literals on the reference layer is associated with a contiguous substring of the appearance layer text, and represents the semantics of the associated substring. We have introduced this conceptual model as a new ADT, called *paratexts* which has a simple data structure and several expressive functions. By using these functionality, one can express rich queries, based on not only matching of character strings, but also semantics of them, on paratexts in databases.

Keywords:

structured documents, databases, query, abstract data types

発表リスト

学術論文

1 加藤弘之, 吉川正俊: “オブジェクトリンクを有する構造化文書を管理するデータベースのモデルと問合せ”, 電子情報通信学会論文誌 (D-I), Vol. J82-D-I, No. 1, pp. 150-164, 1999 年 1 月.

国際会議

2 Hiroyuki Kato and Masatoshi Yoshikawa: “Constructing Structured Document Views” Proc. of International Workshop of New Database Technologies for Collaborative Work Support and Spatio-Temporal Data Management(NewDB'98), pp. 440-447, November 19-20, 1998. (to appear In Lecture Notes in Computer Science(LNCS), Springer-Verlag.)

国内会議 (査読付)

3 加藤 弘之, 吉川 正俊, 植村 俊亮: “構造化文書とデータベース間の汎用リンク機構の実現法”, 情報処理学会アドバンスト・データベース・シンポジウム '95, pp. 9-18, 1995 年 12 月.

4 加藤 弘之, 吉川 正俊: “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”;, 情報処理学会アドバンスト・データベース・シンポジウム '97, pp. 79-86, 1997 年 12 月

研究会, 全国大会など

5 加藤 弘之, 吉川 正俊, 植村 俊亮: “Querying Structured Documents with Object Links” , 夏のデータベースワークショップ'97 北海道, 情報処理学会研究報告 97-DBS-113, 札幌, 1997 年 7 月, pp.293 - 298.

6 加藤 弘之, 吉川 正俊, 絹谷 弘子: “A Database System Integrating Structured Documents and Objects”, 第 55 回情報処理学会全国大会, 福岡, 1997 年 9 月, pp.3-7 - 3-8

7 加藤 弘之, 吉川 正俊: “オブジェクトリンクを有する構造化文書を管理するデータベースシステム ParaDocs - モデルと問合せ -”;, 電子情報通信学会技術研究報告, DE98-4 (Vol. 98, No. 42), pp. 23-30, 1998 年 5 月.

共同研究

8 Masatoshi Yoshikawa, Hiroyuki Kato and Shunsuke Uemura: “A Uniform Mechanism for Incorporating Database Objects into SGML Documents”, Proc. of International Symposium on Digital Libraries 1995 (ISDL'95), pp. 289-290, August 1995.

9 絹谷 弘子, 吉川 正俊, 加藤 弘之: “統合型文書データベースにおけるオブジェクトと文書内容の相互参照機構の構築”, 第 55 回情報処理学会全国大会, 福岡, 1997 年 9 月, pp.3-5 - 3-6

10 金本 博隆, 加藤 弘之, 絹谷 弘子, 吉川 正俊: “効率的な更新が可能な構造化文書の索引”, 情報処理学会研究報告, 98-DBS-114, 1998 年 1 月.

11 金本 博隆, 加藤 弘之, 絹谷 弘子, 吉川 正俊: “効率的な更新が可能な構造化文書索引手法”, 第 56 回情報処理学会全国大会, 1Y-01, 1998 年 3 月.

12 絹谷 弘子, 加藤 弘之, 吉川 正俊: “オブジェクトリンクを有する構造化文書を管理するデータベースシステム Paradocs - アーキテクチャ -”, 電子情報通信学会技術研究報告, DE98-5 (Vol. 98, No. 42), pp. 31-38, 1998 年 5 月 13 日.

13 Masatoshi Yoshikawa, Hiroyuki Kato, Hiroko Kinutani, Masahiro Watanabe: “The ParaDocs Document Database System and Visual User Interface for Information Retrieval”, In Advanced Database Systems for Integration of Media and User Environments '98(Advanced Database Research and Development Series Vol.9), pp. 81-86, World Scientific, 1998.

14 Hirotaka Kanemoto, Hiroyuki Kato, Hiroko Kinutani, Masatoshi Yoshikawa: "An Efficiently Updatable Index Scheme for Structured Documents", Workshop Proceedings of the 9th International Conference and Workshop on Database and Expert Systems Applications(DEXA'98), pp. 991-996, 1998.

15 Masatoshi Yoshikawa, Hiroyuki Kato, Hiroko Kinutani : "Design Framework of a Database for Structured Document with Object Links", IEICE Transaction on Informatin and Systems, Vol. E82-D, No. 1, pp. 147-155, January, 1999.

目 次

図目次

第 1 章

序論

1.1 研究の背景と動機

データベース管理システムは，形式が固定されておりあらかじめ決められた型に従ったデータから，柔軟であらかじめ型が決められていないデータへとその管理対象の範囲を広げつつあり，近年，構造化文書もその管理対象となっている [CACS94, BCD+95, BCD+98, YA94, VAB96]．構造化文書をデータベースで管理することの利点は，文書の作成，編集，更新など文書処理の様々な側面において，同時実行制御，アクセス制御，障害回復などのデータベース機能を適用することが可能となる点にある．更に，文書の内容のみならず論理構造も併せてデータベースで管理することで，これらデータベース機能を論理構造単位に適用することができるようになるだけでなく，構造化文書を管理するデータベースに対して文書の内容と構造の両方に基づく問合せが可能となる．このような理由により，構造化文書データベースに対する研究は重要である [SDAMZ94, BYN96, SDDTZ97]．

しかしながら，これまでの構造化文書データベースに関する研究は，構造化文書のデータベースによる管理手法に関する研究に偏っており，データベースで管理されている構造化文書の利用のための枠組に関する研究はほとんど存在しない．そこで本研究では，データベースで管理されている構造化文書を高度利用する際の枠組として，特に次の二つの点に着目する．

(1) データベースビューとしての構造化文書の構築.

(2) データベースで管理されている他の型の値との統合利用.

まず上記 (1) に関する研究の動機について論じる．構造化文書をデータベースビューとして構築することによる利点は，通常のデータベースビュー同様，論理的データ独立性にある．Boehm らは応用からのアクセスに依存した構造化文書の部分分解管理によって，部分文書の挿入や文書全体の再構築など構造化文書固有の処理が効率的に実行できることを示した [BAK95]．しかしながら，この分割構造をそのまま構造化文書モデルとすることは，文書の論理構造のうちあるものはデータベースのスキーマ構造に対応しているが，あるものは対応しておらず整然としていない．すなわち問合せの際，ある論理構造については経路式を用いて指定できるが，ある論理構造については経路式ではなく論理構造を操作する関数を用いて指定するということが生じる．更に，一般に，応用からのアクセス単位は時間と共に変化するので，文書の部分分解単位もこれに併せて変化させると，応用に対するデータ独立性が維持できなくなる．そこで，この論理的データ独立性維持のためには，構造化文書をデータベースビューとして構築することが必要となる．また，構造化文書ビューの構築により，部分文書の再利用の促進や，同じ文書情報源から，異なる内容や構造を有する文書を定義することができる．例えば，利用者の興味に基づく新聞記事の構築や，学習者の背景知識に基づくテキストブックの自動生成など，仮想文書 (virtual document) と呼ばれる文書の重要性が指摘されている [SM96]．以上の理由により，データベースビューとしての構造化文書に関する研究が必要である．

次に上記 (2) に関する研究の動機について論じる．文書をデータベースで管理することによる他の利点として，データベース中の文書データと他のデータの統合利用が挙げられる．すなわち，文書以外のデータ間の計算に基づく文書データの検索や，文書処理に基づく文書以外のデータの検索が可能となる．例として新聞記事と企業データに関する以下のような問合せが考えられる．

- 日本の企業名が本文に出現する記事の見出しの検索
- 1997 年 8 月 9 日の記事の本文に出現する企業の国籍の検索

しかしながらこれまで提案されたシステムでは，文書データと文書以外のデータの統合は文字列パターンマッチに基づいており，限界がある．すなわち，i) 同じ文字列が異なる対象を表す場合や，ii) 同じ対象を異なる文字列で表す場合などの状況に対処することができない．i) の例として，文字列 “president” は大統領という意味で用いられる場合と社長という意味で用いられる場合とがある．また，ii) の例として，新聞記事の見出しには略語や愛称などが用いられる場合がある．以上の理由により，データベースに格納されている文書データと他のデータ型の値の間の統合に関する研究が必要となる．

尚，本研究で扱う構造化文書は，今後幅広く用いられる可能性が大きく，事実上の国際標準でもある XML[WWWC98] で記述された文書とする．

上記 (1),(2) の研究を行なうためには，構造化文書を扱うデータベース問合せ言語を導入する必要がある．一般にデータベースに対する問合せ言語には，検索条件の指定と結果の構造の構築の二つの機能が必要とされる．XML 文書の操作という文脈において，検索条件の指定とは文書の論理構造に基づく所望の部分文書の特定であるのに対して，結果の構造の構築とは所望の論理構造を有する XML 文書の構築である．前者は検索対象に対する操作であり，後者は検索結果に対する操作である．また，内部に構造を有するデータを扱うデータベース問合せ言語は，検索対象に対する構造に基づく検索条件の指定と検索結果の構造の定義を，統一的な文法で記述するか非統一的な文法で記述するかによって，大きく二つに分類される．例えば，OQL[Cat97] や STRUQL[FFK+98] は，検索対象に対する構造に基づく選択条件の指定と検索結果の構造の定義を別々の文法を用いて記述している．これに対して，例えば，内部に複雑な構造を有するデータに対して *data forest* というラベル付き順序木の森を用いたデータモデルと，Datalog に似た問合せの記法と意味の導入により，検索対象と検索結果の操作を統一的に記述する研究も存在する [ACM97, CDSS98]．実用的な観点から，本研究では既に幅広く用いられており事実上の国際標準の一つとなっている OQL[Cat97] の構文を採用することにした．ISO による国際標準である SQL ではなく OQL の構文を採用した主な理由は，OQL の集合に対する操作の簡易性が挙げられる．これにより，検索対象に対する構造に基づく選択条件の指定と，検索結果の構造の定義には別々の文法を導入する．実際，

PREDATOR[SLR97] などの型拡張可能データベースシステムでは，新たな定義域を抽象データ型としてプラグインすることでSQLからその抽象データ型に定義されている関数や述語を用いることができ，この新たな定義域の値の操作が可能となる．したがって，本研究ではXML文書を管理するための抽象データ型の導入により，OQLスタイルの問合せにおいて，XML文書のための抽象データ型に定義されている関数や述語，演算子を記述でき，データベースに格納されているXML文書を操作できるようにした．

現在，ISOでは構造化文書を取り扱うためのデータベース言語SQL/MM[ISO97]を策定中である．しかしながら，この言語は非常に複雑なものとなりつつあり，利用者にとって必ずしも記述しやすい言語となっていない．また，最近，W3CでもXML文書のための問合せ言語についての活動を始めた[Wor98]．よって，いずれ，XML文書を扱うデータベース言語は標準化されると予想される．本研究で提案する構造化文書ビューと文書データと他の型のデータの統合利用は，本質的に，XML文書を扱うデータベース言語の構文とは直交するものである．

1.2 研究の概要

本研究では，先に述べた構造化文書データベースの研究における二つの着眼点に関して，それぞれ次のような主な成果を得た．

(1) データベースビューとしての構造化文書の構築

i) 構造化文書ビューの定義からビューの論理構造を導出するアルゴリズムの開発．

ii) 構造化文書ビューに対する問合せの書き換え規則の開発．

(2) データベースで管理されている他の型の値との統合利用

iii) 構造化文書をデータベースで管理する際の新しいデータモデルの開発．

iv) このデータモデルによって構造化文書が管理されているデータベースに対する問合せ言語と問合せ処理手法の開発.

以下，それぞれの研究成果に関してその概要と利点について述べる.

1.2.1 データベースビューとしての構造化文書の構築に関する研究成果の概要

構造化文書をデータベースビューとして構築する場合，通常のデータベースビューと同様に，構造化文書データベースに対する問合せ言語を用いて定義できる．構造化文書を扱う問合せ言語には，既に述べたように，OQL の構文を採用する．同じ構造化文書源から異なる構造化文書ビューを定義することで，利用者独自の文書を構築することが可能となる．例えば，利用者の興味に基づく新聞の構築が可能となる．この新聞は，購読者の興味の度合が大きい記事と興味の度合の小さい記事とで，記事の詳細度が異なる新聞のことである．実際，このような新聞の配信が始まっている [SM96]．図 1.1は，同じ新聞記事から異なる詳細度を有する文書ビューの例を示している．この種の文書は仮想文書 (virtual documents) と呼ばれている [SM96]．別の例としては，学習者の背景知識に基づく教科書や利用者の環境に基づくソフトウェアマニュアルなどが挙げられる．本研究では，これらの仮想文書の構築にはデータベース技術が貢献できることを示す.

このような構造化文書ビューに関して，ビュー定義からビューの論理構造をあらかじめ知っておくことの利点は二つある．一つは，利用者の問合せの記述支援であり，もう一つはビューに対する問合せ処理の最適化に利用できる．構造化文書ビューの論理構造をあらかじめ知っておくことで，ビューに対する問合せのうちビュー上に存在しない論理構造に関する問合せは，実データを用いず，問合せの構文情報を用いて処理することができる．これに対して文書ビューの論理構造があらかじめわからないと，この種の問合せは実データを参照することで処理される．したがって，構造化文書ビューの論理構造をあらかじめ知っておくことは，問合せ処理の最適化につながる．本研究では，この利点を享受するため，構造化文書ビューの定義からビューの論理構造を導出する

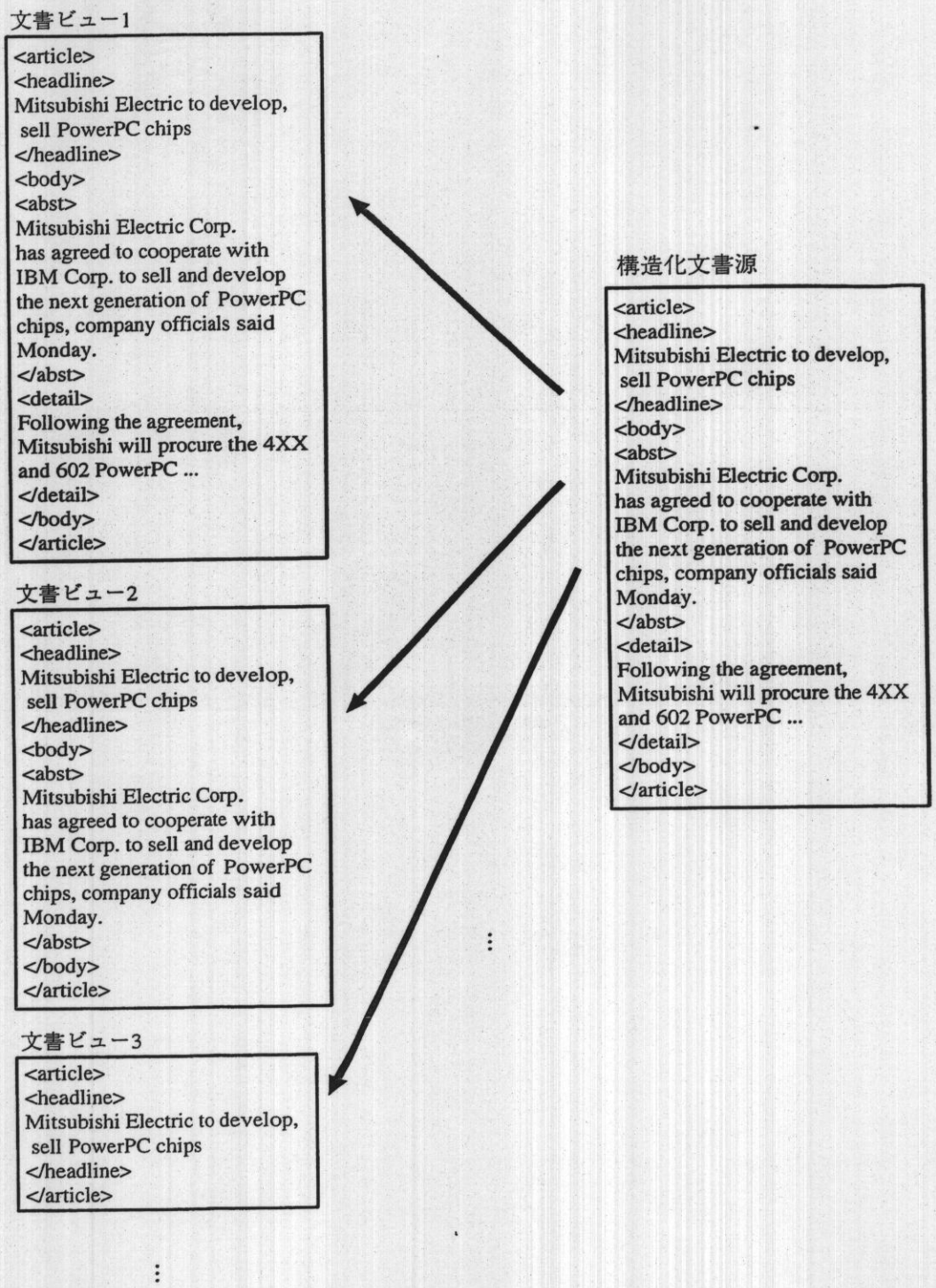

図 1.1 同じ新聞記事からの詳細度の異なる文書ビューの例

アルゴリズムを提案する．

また，構造化文書ビューに対する問合せの処理において，文書検索固有の条件式の評価手法を提案する．ビューに対する問合せは，その問合せ中のビューの出現をビュー定義に置換することで単純に処理することができる．しかしながら，一般にビューに対する問合せの条件式を，ビュー定義の条件式に移すことにより，最適化することが可能である．この問合せ最適化は，問合せ処理手順を木表現した際に，選択条件を先に評価するため木の葉へ移動することから，“selection push-down”[AHV95] と呼ばれる伝統的な最適化手法である．選択条件を先に評価することで，問合せ処理における中間結果の数を減らすことができる．これにより処理対象数が減少し，ほとんどの場合，処理全体の効率の向上につながる．本研究では特に，XML 文書固有の選択条件に着目し，この選択条件の push-down する際の選択条件の書き換え規則を提案する．

1.2.2 データベースで管理されている他の型の値との統合利用に関する研究成果

データベース中の文書データと文書以外のデータを統合利用するための，構造化文書の新しい概念モデルを提案する．構造化文書中には出現する文字列には，その意味がデータベースで格納されているものや，あるいはデータベースに格納されている値そのものを表す場合がある．構造化文書中の文字列とデータベース中のその文字列の意味データ間に双方向リンク機構を構築することで，意味データの計算に基づく文書検索や文書処理に基づく意味データの検索といった，データベース中のさまざまな型のデータの意味的統合利用が可能となる．

本研究ではこのようなリンク付きの文書を統一的に扱う概念モデルである *paratext* モデルを提案する．この paratext モデルにおいて，文書はその表記文字列が保持されている表記層と，表記層における部分文字列が表す意味が保持されている参照層の二つの層から構成されているものとみなす．この paratext データをデータベースで管理するための枠組と paratext データを管理しているデータベースに対する問合せ言語と問合せ処理を提案する．

1.3 論文構成

本論文は以下のような構成である．第2章では本研究で扱う構造化文書記述言語であるXMLについてその概要を述べる．第3章では，構造化文書をデータベースで管理する際の手法について既存の研究を分類した上で，本研究におけるアプローチを明確にする．第4章では第3章で記述した本研究で採用した型拡張可能データベースシステムに対して，構造化文書を管理するための抽象データ型を導入する．第5章では，データベースビューとしての構造化文書の構築と，構造化文書ビューに対する問合せの最適化について論じる．第6章では構造化文書をデータベースで管理する際の新しい概念モデルとその問合せを提案する．第7章は結論と今後の課題について論じる．

第 2 章

XML の概要

本研究では，構造化文書の対象として **XML(eXtensible Markup Language)** [WWWC98, JIS98b] で記述された文書を採用する．この章では，XML で規定されているもののうち，本論文で議論を展開する上で必要な部分の概要を記述する．

XML は，W3C(World Wide Web Consortium) のもとで開発された構造化データを記述するための言語である．XML は **SGML**[ISO86, JIS92, JIS98a] のきわめて簡単な方言である．SGML が文書を対象として開発された経緯により，XML で記述されたオブジェクトは，XML 文書と呼ばれるが，文書だけではなく，内部に論理構造を有するデータ全てが XML の記述対象である．XML の目標は，現在の HTML と同様に，汎用的な SGML が Web 上で配布，受信，処理できるようにすることである [JIS98b]．XML 文書はエレメント[1]と呼ばれるデータの論理構造を，開始タグと終了タグを明示的に文書中に埋め込むことで表現したデータオブジェクトである．

文書型定義 (Document Type Definition, DTD) は，文書中のエレメントの出現の順序を規定したものであり，本質的な役割は文書の論理構造に関する文法とみなすことができる．エレメントは DTD 中の**エレメント宣言**によって定義される．XML 文書中のあるエレメント A がエレメント B に含まれており，かつ B に含まれる他のエレメントには含まれていないとき，A を B の子

[1] JIS 規格 [JIS98b] では,「要素」と訳されているが，集合の要素などと区別するために，本論文では「エレメント」と記述することにする．

エレメントと呼ぶ．一つのエレメント宣言は，エレメント名とその内容モデル (content model) から構成されている．内容モデルは，子エレメント名と 5 種類の演算子（“,”, “*”, “+”, “|”, “?”）を用いた正規表現であり，各演算子の意味は以下の通りである．但し，r，r_1，r_2はエレメント名を表し，ϵは空列を表すものとする．

- “r_1,r_2” は，r_1と r_2の列，すなわち結合を表す．
- “r_1 | r_2” は，r_1または r_2を表す．
- “r*” は，rの 0 回以上の繰り返しを表す．
- “r+” は，“r,r*’ を表す．
- “r?” は，“$r|\epsilon$” を表す．

DTD において EMPTY キーワードを用いて宣言されたエレメントは，そのエレメントが文書中に現れたとき空でなければならない．空エレメントは開始タグがそのエレメントの全体を構成する．以下，データベースの世界における空値と区別するため，空エレメントの内容を EMPTY と記述する．

内容モデルには，**要素内容 (element content)** と**混在内容 (mixed content)** の 2 種類がある．要素内容は，文字データは含まず，子エレメントだけを必ず含み，子エレメント間は空白だけしか現れないような内容モデルである．これに対して，混在内容は，子エレメントに混在して文字データが含まれる可能性のある内容モデルである．

与えられた DTD に対して，その DTD に従っている XML 文書を**妥当な (valid)** XML 文書と呼ぶ．また，XML では DTD を持たない，もしくは DTD に従わない XML 文書の存在を許している．しかし，この XML 文書のタグの出現が XML の文法（例えば開始タグと終了タグの対が必ず存在するなど）に従っている必要がある．このような XML 文書を**整形式の (well-formed)** XML 文書と呼ぶ．各エレメントには，属性を定義することができる．これは DTD 中の属性宣言で定義する．

あるエレメントの XML 属性をそのエレメントの子エレメントとみなし，混在内容に関しては疑似エレメント [DD94] を導入することで，全ての XML 文書

```
<!ELEMENT article (year,title,author+,sec+)>
<!ELEMENT sec     (title,body)>
<!ELEMENT title   (#PCDATA)>
<!ELEMENT author  (#PCDATA)>
<!ELEMENT body    (#PCDATA)>
```

図 2.1 DTD の例

を要素内容モデルから構成されているものとすることができる．そこで，本研究では要素内容モデルから構成されている XML 文書を研究の対象とし，XML 属性及び混在内容については取り扱わない．

図 2.1にエレメント宣言だけから構成されている DTD の例を示す．この DTD において，エレメント article 中には，year エレメントの後に title エレメントが続き，その後に author エレメントが 1 回以上出現し，その後に sec エレメントが 1 回以上出現することを規定している．

第 3 章

構造化文書のデータベースによる管理手法の分類と本研究におけるアプローチ

構造化文書をデータベースで管理する際，文書の内容のみならずその論理構造も併せて管理することで，構造化文書が格納されているデータベースに対して文書の内容と構造の両者に関わる問合せをすることができる [BYN96]．この章では，構造化文書のデータベースによる管理手法を，文書の内容の管理，文書の論理構造の管理，そして物理的格納手法の三つの観点からそれぞれ，**3.1**，**3.2**，**3.3**で分類する．その後，それぞれの観点に対応する本研究のアプローチについて記述する．

3.1 文書内容の管理

文書データの検索は，これまで主に情報検索の分野で研究されてきており，その成果として，転置ファイルやパトリシア木などの全文索引に基づくデータ構造を用いることで洗練された様々な文字列パターンマッチ処理 (近接問合せ，接頭語問合せなど) を効率的に実行することができる [FBY92]．データベースで文書を管理する際に，このような技術を利用した既存の情報検索システム

(文書検索システム) をデータベースシステムに統合する様々な研究が存在する [YA94, BCD+95, ElM96, VAB96]. これらの研究の特徴は, SQL などのデータベースインタフェースを通して情報検索システムで格納されている文書データの操作を可能とするような枠組を提供していることである. そのためにデータベースと情報検索システムを緩やかに統合している.

3.2 論理構造の管理

構造化文書をデータベースで管理する際の, 文書の論理構造の管理方法は次の二つに分類される.

(1) データベースのスキーマ記述能力で管理する.
任意の文書の論理構造に対応できるデータベーススキーマを構築する. つまり, 構造化文書は管理されているデータベースのデータモデルに即してアプリケーション側から見える. 例えば, Christophides ら [CACS94] や Volz ら [VAB96] はオブジェクト指向データモデルを, Sacks-Davis ら [SDKR+95] は入れ子関係データモデルを構造化文書のモデルとして採用している. したがって, 問合せにおいて文書の論理構造はデータベースのスキーマ構造で表現されている. この手法の利点として文書の論理構造単位での, 同時実行制御, アクセス制御, 障害回復などのデータベース機能の実現が容易である点が挙げられる. しかしながら, 文書の論理構造のわずかな変化がデータベースのスキーマに影響を与えてしまうという欠点がある. また, 次のような欠点がある. 文書の論理構造を適切に管理できるよう, データベースのスキーマ記述能力を拡張しているが, この拡張はデータベース管理システム全体の拡張である. したがって, この拡張は他のデータ型にも影響を与えるだけでなく, データベース管理システム全体の拡張なのでそのコストは一般に非常に高くなる.

(2) データベースにプラグインされた構造化文書のための抽象データ型で管理する.
データベース中のある属性に一つの構造化文書全体が管理されており, 文書の論理構造はデータベースモデルにおけるスキーマ構造ではない. 問

合せにおいて文書の論理構造を指定するには，文書が管理されている抽象データ型特有の関数を用いる．文書の論理構造をその始まり位置と終り位置の対 (以下，本論文ではリージョンと呼ぶ) で定義し，その集合に対する代数演算 [ST94, CM94, CM95, Bur92, CCB95a, CCB95b] を用いて文書の論理構造を操作するものとして [BCD+95] がある．また，リージョンをもとに文書の論理構造を木で表現し，*tree inclusion primitive* [KM93] を用いて文書の論理構造を操作するものがある [YA94]．また，情報検索技術を利用した文書データのための抽象データ型をデータベースにプラグインする研究も存在する [DM97]. これらの研究では，論理構造単位でのデータベース機能の適用について議論はなされていない．

3.3 文書の格納手法

文書を格納する際に分解するかどうかに関しては，以下の三つに分類することができる．

(1) その論理構造に従って分解し， 格納する．
論理構造単位でのアクセスや再利用に適している．しかしながら，完全に分解してしまうと，文書全体を組み立て直す処理などに負荷がかかる．

(2) 部分的に分解し，格納する．
文書はアプリケーションから頻繁にアクセスされる単位に基づいて分解され，格納される [BANY97].

(3) 分解せずにそのまま格納する．
文書を分解しないことで適用できる特別な索引である PAT[GBYS92] を構築することができる．例えば，[BCD+95] が挙げられる．

3.4 本研究におけるアプローチ

本研究で扱う構造化文書は既に述べたように，実践的な立場から XML 文書とする．XML 文書をデータベースで管理する際，抽象データ型アプローチを

採用する．すなわち，XML 文書の新たな抽象データ型，XML 型を型拡張可能データベースにプラグインする．したがって，XML 文書の論理構造と文書内容の双方はこの抽象データ型によって管理される．抽象データ型アプローチの利点は既に広く認識されている [Sto96, CD96]．このアプローチの最大の利点は，モジュール性と拡張可能性にある．すなわち，各抽象データ型の追加や削除を，データベースの他の部分に影響を与えることなく逐次的に達成できる．最近では，データベースに新たな定義域を抽象データ型としてプラグインする際，型固有の最適化規則や同じ型でも異なる実装を持つ問合せ処理の装着といった，最適化を意識した定義域の追加に関する研究がなされている [SLR97, Ses98b]．これに対して，他の管理手法，例えば [CACS94] のように文書の論理構造をデータベーススキーマ記述能力によって達成する場合，先に述べた欠点が存在する．

次に，物理的格納手法について述べる．基本的には，「構造化文書は応用からのアクセスパターンに応じて部分分解して格納することで，文書全体の検索や部分文書の挿入を含めた文書処理全体がリーゾナブルなコストで実現できる」という Boehm らの主張 [BAK95] を採り入れる．しかしながら，この分割構造をそのまま構造化文書モデルとすることは，文書の論理構造のうちあるものはデータベースのスキーマ構造に対応しているが，あるものは対応しておらず整然としていない．すなわち問合せの際，ある論理構造については経路式を用いて指定できるが，ある論理構造については経路式ではなく論理構造を操作する関数を用いて指定するということが生じる．

そこで，本研究では構造化文書のデータベース内部表現としては文書の再利用と応用からの要請という観点からも適切に分解してデータベースで管理する一方で，応用に対しては一つ抽象データ型の値として入力できるようにする．換言すれば，これは関係モデルにおける基底表とビューとの関係に相当する．

3.4.1 データベースモデル

本論文において，XML 文書を管理するデータベースモデルはオブジェクト関係データベースモデルとし，利用者定義の型を逐次追加することができる型拡張可能データベースシステムとする．我々が仮定する型拡張可能データベー

スにおいて管理されるデータ型は，原子型と複合型に大きく分類される．原子型は更にINTEGER, STRING, BOOLEANなどの**従来の原子型**(*traditional atomic type*)と，TEXT, IMAGE, AUDIOなどの**値指向の抽象データ型**(*value-ADT*)[Cor97a]の二つに分類される．SQL/92[DD93]で規定されているデータ型は我々が仮定するデータベースにあらかじめ備えられているものとする．既に確立された手法[AB95]を用いて，型と値は次のように定義される．

定義 3.4.1: 型と値は以下のように形式的に定義される．但し，定義域と定義域名は関連付けられているものとする．

- $\hat{D}$が定義域名であるとき，$\hat{D}$は原子型である．$\hat{D}$に関連付けられている定義域D中の各aに関して，aはこの型の値である．
- $t_1, ..., t_n$を型，$A_1, ..., A_n$を属性とすると$[A_1 : t_1, ..., A_n : t_n]$は組型である．$v_1, ..., v_n$をそれぞれ型$t_1, ..., t_n$の値とすると，$[A_1 : v_1, ..., A_n : v_n]$はこの型の値である．
- tを型とすると，$\{t\}$は集合型である．型tの値の有限集合は，型$\{T\}$の値である．

□

従来の原子型のデータはそのリテラルによって値が同定されるのに対して，それ以外の型である値指向の抽象データ型と複合型のデータはオブジェクト識別子(OID)によって同定できるものとする[1]．我々が仮定するデータベースのIS_A階層は型に関する下位型関連(*subtyping relationship*)$\sqsubseteq$によって構築される．特に，我々は原子型である値指向の抽象データ型(value-ADT)についても下位型関連を定義する．この場合，複合型に対する下位型関連と異なり，型の意味である定義域を用い，いわゆる*domain-inclusion semantics*[AHV95]によって下位型関連を定義する．ここである原子型tに関して，$dom(t)$はtの定義域を表すものとする．tがINTEGERやSTRINGの場合のように，$dom(t)$の定義域は可算無限集合となる場合がある．

[1]実際は，値指向の抽象データ型の値はファイル名などハンドルを用いることが一般的であるが，このハンドルをオブジェクト識別子とみなすことができる．

定義 3.4.2: ある原子型の集合 U_a に関する下位型関連は U_a 上に定義される以下の条件を満たす最小半順序 (smallest partial order) $\sqsubseteq_a$ である.

- $t, t' \in U_a$ に関して, もし $dom(t) \subseteq dom(t')$ であるならば, $t \sqsubseteq_a t'$.

□

定義 3.4.3: 原子型と複合型を含むある型の集合 U に関する下位型関連は U 上に定義される以下の条件を満たす最小半順序 $\sqsubseteq$ である. 但し, $t, t', t_i, t_i' \in U$ とする.

- もし $t \sqsubseteq_a t'$ であるならば, $t \sqsubseteq t'$;
- もし 各 $i \in [1, n]$ に関して, $t_i \sqsubseteq t_i'$ であり, かつ $n \leq m$ であるならば
  $[A_1 : t_1, \cdots, A_n : t_n, \cdots, A_m : t_m]$
  $\sqsubseteq [A_1 : t_1', \cdots, A_n : t_n']$;
- もし $t \sqsubseteq t'$ であるならば, $\{t\} \sqsubseteq \{t'\}$;
- 任意の $t \in U$ に関して, $t \sqsubseteq$ ANY.

□

ある型 t に関して, $t+$ は t の全ての下位型 (定義より t を含む) の集合を表すものとする. あるデータベースインスタンス D 中の値の型に関して, t_a を従来の原子型, t_n をそれ以外の型, すなわち値指向の抽象データ型である原子型または複合データ型とするとき, $I_D(t_a)$ は t_a の定義域を表すのに対して, $I_D(t_n)$ は t_n のインスタンスの OID の集合を表すものとする. したがって,

$$I_D(t+) = \cup\{I_D(t') \mid 't \sqsubseteq t\}$$

となる.

第 4 章

構造化文書を管理する抽象データ型の設計

本研究では，既に述べたように，XML 文書を **3.4.1**で説明した型拡張可能データベースで管理する．このとき，XML 文書を管理する型として抽象データ型アプローチを採用する．抽象データ型アプローチの特徴はモジュール性と拡張可能性にある [Sto96, CD96]．すなわち，他の型に影響を及ぼさずに，データベースに対して型の追加や削除ができるという利点がある．

この章では，前述したデータベースで管理されている XML 文書を管理する抽象データ型 XML 型固有の関数や述語を導入する．これにより，このデータベースに対する問合せ[1]中にこれらの関数や述語を記述することで，データベース中の XML 文書を操作することが可能となる．

一般にデータベースに対する問合せには，検索条件の指定と結果の構造の構築の二つの機能が要求される．XML 文書の操作という文脈において，検索条件の指定とは論理構造に基づく所望の部分文書の特定であるのに対して，結果の構造の構築とは所望の論理構造を有する XML 文書の構築である．前者は検索対象に対する操作であり，後者は検索結果に対する操作である．

以下，この章は次のように構成されている．まず準備として通常の文書のための抽象データ型について **4.1**で記述する．次に，**4.2**では，XML 文書を管

[1]本研究では，既に述べたように OQL[Cat97] を採用する．

理する抽象データ型であるXML型を定義することで，XML文書を操作するデータベース問合せ言語を提案する．また，4.3で提案する言語の記述能力を非形式的に述べる．

4.1 準備

4.1.1 リージョン

文書中の文字列の出現位置を管理するのに，該当する単語の始まり位置と終り位置 (すなわち，文書の先頭から何文字目という情報) の対を用いる．この対は通常リージョンと呼ばれる．文字列長がlであるテキストのリージョンは単に整数値の対(a,b)で表現される．但し，$1 \leq a \leq b \leq l$である．

4.1.2 TEXT型

TEXT型は通常のプレインテキストを管理するための型であり，インスタンスは一次元の文字列である．文書の管理，とりわけ検索に関する研究はこれまで情報検索の分野でなされてきており，その成果として全文索引 (転置ファイル，パトリシア木，シグネチャファイルなど) に基づく高度な文字列パターンマッチ処理 (近接問合せ，接頭語問合せなど) を実現する枠組が提案されている．特に，単語など部分文字列の出現位置をリージョンで管理することで，文字列パターンマッチ処理は単なる整数の対に関する計算によって実現され，効率的に処理することが可能となる．このようなリージョンに基づくテキストモデルとして，[CCB95a, CCB95b, CM94] などがある．このようにこれまでの情報検索の分野での研究成果を適用することで，TEXT型を実現することができる．実際，既存のテキスト検索システムをデータベースに統合したシステムが提案されている [BYN96]．TEXT型には，全文索引に基づく文字列パターンマッチ処理でIRS(情報検索システム) が提供するような機能を持つ一連の述語を備えているものとする．例えば，式

TEXT *arg1* CONTAIN STRING *arg2*

はテキスト *arg1* 中に文字列 *arg2* が出現する時かつその時に限り真を返す．

STRING 型との違いは，TEXT 型の属性には全文索引が構築できることにある．

4.2 XML 型

XML 型は TEXT 型の下位型であり，XML 文書を管理するための型である．エレメント名を管理するための型として，STRING 型の下位型でエレメント名を値とする ELEMENT 型を導入する．ELEMENT 型の値のリテラルはエレメント名を '<' と '>' で囲むことで表現される．[CACS94] 同様，ELEMENT 型の変数を SQL で用いる場合，ELE_をつけることでその値域を制限する．

2で述べたように，文書型定義 (DTD) はエレメントの出現順序を規定しているものであり，その中心的な役割は文書の論理構造についての文法を定義することである．換言すると，ある DTD はそれにしたがって記述された XML 文書実現値の制約とみなすことができる．したがって，XML 型のデータベース属性のうち妥当な XML 文書を管理する属性は制約として DTD を保持するものとする．

以下，XML 型固有の関数や述語，換言すれば XML 型の値に関する操作体系について，**4.2.1**では論理構造に基づく検索対象からの部分文書の特定に関する操作体系について，**4.2.2**では検索結果の論理構造の定義に関する操作体系について記述する．

4.2.1 論理構造に基づく選択条件の指定に関する操作体系

検索対象に対する XML 文書の操作には論理構造に基づく部分文書の特定と，論理構造自身の特定の二種類ある．前者として **ST^e-式**[2]を，後者として二つの述語 `include` と `precede` を導入する．

Gonnet らは構造化文書をある文脈自由文法の導出木とみなした [GT87]．同様に XML において，DTD は文脈自由文法であり文書実現値はその文脈自由

[2]ST は Structured Text の略であり，^{e}は extension(外延) の略である．これは，検索対象の構造化文書の部分を特定する機能を表す式であることを意味している．

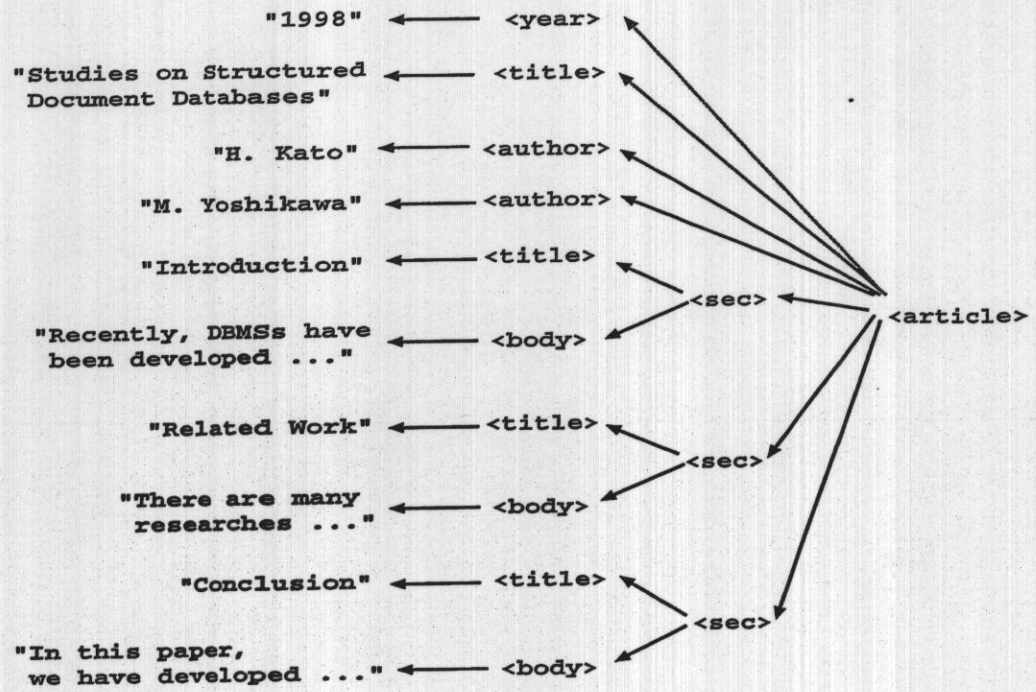

図 4.1 XML 文書の順序木表現の例

文法における導出木（順序木[3]）として表現することができる．本研究において，この順序木は非終端ノードには文書の論理構造であるエレメントが対応しているのに対して，終端ノードは文書の内容である文字列が保持されているものとする．以下にこの順序木の形式的な定義を与える．

定義 4.2.1: ある DTD に関して妥当な，XML 文書 Dは次のような順序木 Tとして表現できる．

- Tの根は Dの根エレメント名に対応する．
- T中のある非終端ノード iに関して，iの子ノードは iに対応するエレメントの子エレメント名に対応しており，子ノードの順序は D中に出現する子エレメントの出現順序に対応する．

[3]混在内容はこのような順序木で表現することはできないが，疑似エレメント [DD94] を導入することで基本的に同様の順序木として扱うことができる．しかしながら，詳細については本研究の範囲外とする．

- T中の終端ノードtには兄弟は存在せず，tの親ノードpはD中の子エレメントを持たないエレメントに対応している．tにはpに対応するエレメントの文書内容が対応している．

□

一般に，順序木上を深さ優先探索する際の各ノードの探索順は全順序となる．この順序木のノード間の順序はこの全順序に従うものとする．例として，図2.1に示したDTDに関して妥当なXML文書の順序木表現を図4.1に示す．上記の定義に従えば一つのXML文書に対応する順序木は一つだけである．以下，XML文書をこのような順序木としてモデル化し，この順序木上の操作体系について考える．

順序木のノード間の関連には親子関係と兄弟関係があるので，論理構造の操作体系のための基本関係は論理構造間の包含関係と順序関係である．これら二つの基本関係を用いることで、論理構造に基づく任意の部分文書の特定と、任意の論理構造間の関係を記述することができる。

まず，論理構造に基づく部分文書を特定する**ST^e-式**は次のように定義される．ST^e-式を評価した値の型はXML型を要素とするリストとなる．

定義 4.2.2: AをXML型の値または変数，EをELEMENT型の値または変数とすると，ST^e-式は次のように再帰的に定義される．

- ST^e-式 := A
- ST^e-式 := ST^e-式/E
- ST^e-式 := ST^e-式< E
- ST^e-式 := ST^e-式//E
- ST^e-式 := ST^e-式<< E

□

ST^e-式は必ずXML型の値または変数から始まることに注意されたい．先に記述したようにXML型の値をエレメント名を中間ノードとする順序木として考

えると，上で定義したST^e-式の意味はそれぞれ次の通りである．但し，ST^e-式の値を評価した結果のリストの順序は，前述した，順序木上の各ノード間に定義されている全順序に従うものとする．

- AはA自身を表す．
- ST^e-式/Eは，ST^e-式で特定される中間ノード（根も含む）の子ノードのうちエレメント名がEである部分文書のリストを表す．
- ST^e-式//Eは，ST^e-式で特定される中間ノード（根も含む）の子孫ノードのうちエレメント名がEである部分文書のリストを表す．
- ST^e-式$< E$は，ST^e-式で特定される中間ノードの兄ノードのうち文書要素名がEである部分文書のリストを表す．
- ST^e-式$<< E$は，ST^e-式で特定される中間ノードに先行するノードのうちエレメント名がEである部分文書のリストを表す．

尚，問合せの記述を簡単にするために，ST^e-式において，あるデータベース属性に保持されているXML文書で最も外側にタグ付けされているエレメント名は省略できるものとする．

既に述べたようにXML文書を，論理構造を中間ノードとする順序木とみなし，その順序木の部分木のリストを特定する式としてST^e-式を導入した．したがって，文書の論理構造に関する正確な知識を有しているならば，ST^e-式の二つの構文ST^e-式/EとST^e-式$< E$を組み合わせることで，論理構造に基づく任意の部分文書を特定することが可能である．しかしながら，文書検索においては，文書の論理構造に関する正確な知識なしに問合せを記述できることが望ましい．したがって，本研究では二つの構文ST^e-式//EとST^e-式$<< E$を導入することにした．

次に与えられた二つの部分文書間の論理構造の関係を特定する二つの述語includeとprecedeを導入する．A_1, A_2を，XML型の値または変数，Eをエレメント型の値または変数とし，かつ既に述べたようにエレメント名を中間ノードとする順序木として考えると，

- A_1 include A_2 は，A_1がA_2の先祖を頂点とする部分木であるとき，またそのときに限り真を返す．

- A_1 precede A_2は，A_1の出現がA_2の出現に先行するとき，またそのときにに限り真を返す．

4.2.2 論理構造の構築に関する操作体系

この節では，検索結果の XML 文書の論理構造をどのように定義するかについて述べる．すなわち，検索対象におけるの XML 型の値を部品として，これらの部品を組み合わせて新たな論理構造を有する XML 文書をどのように定義するかについて記述する．この XML 型の値を部品とした新たな論理構造を定義することは，換言すれば検索結果の XML 型の値の内部構造を構築するコンストラクタを定義することである．既に述べたように，XML 文書の論理構造はは DTD 中のエレメント宣言によって定義されている．また，エレメント宣言はエレメント名とその内容モデルから構成されている．したがって，これらコンストラクタは二つの種類に分割できる．一つは，内容モデルを記述する際に用いられる正規表現の演算子に一対一に対応する関数や演算子 (+,cat*(),cat(),?,|) であり，もう一つは，内容モデルに対してエレメント名を与える関数 (tagged()) である．これらの関数や演算子を用いることで任意の DTD に従う文書を構築することができる．

まず最初に，説明を明確にするために ST_TEXT 型を導入する．ST_TEXT 型は，TEXT 型の下位型でありかつ XML 型の上位型である．XML 型の値は根エレメントを必ず有しているのに対して，ST_TEXT 型の値は根エレメントを有している必要がないものとする．定義域包含に基づく下位型関連が定義されているので，dom(XML) $\subseteq$ dom(ST_TEXT) が成り立つ．したがって，dom(ST_TEXT)−dom(XML) は，根エレメントを有していない文書集合である．ST_TEXT 型の値に根エレメントを付加したものは必ず XML 型の値である．

ここで，nil と EMPTY の違いについて記述する．nil は空値を表すのに対して，EMPTY は ST_TEXT 型の値である．既に述べたように XML では内容

がEMPTYであるエレメントを許している．したがって，EMPTYに根エレメントを付加した値はXML型の値である．

以下，内容モデル構築のために導入される関数や演算子について記述する．

- 与えられたST_TEXT型の二つの文書を記述された順番に連結して一つのST_TEXT型の文書にする演算子“+”（内容モデルの定義に用いられる列演算子“,”に相当）
  この演算子は結合的でありかつ非可換である．結合法則が成り立つので，多項演算子(polyadic operator)とみなすことが可能である．例えば，$S_1 + \cdots + S_n$と記述することができる．この演算子は被演算子にnilを含む時，結果がnilとなる．すなわち，ST_TEXT型の値$S_1, \cdots, S_n$のうち，ある$i(1 \leq i \leq n)$に対してS_iがnilであるとき，$S_1 + \cdots + S_n = nil$である．

- ST_TEXTの値の順序付き集合値を入力とし，一つのST_TEXT型の値を返す集約関数を導入する．この集約関数には，`cat*()`と`cat()`がある．前者は入力がnilのときEMPTYを出力し，後者は入力がnilのときnilを出力する．

  - `ST_TEXT cat*(set of ST_TEXT` *arg1* `)`(内容モデルの定義に用いられる繰り返し演算子“*”に相当)
    入力されたST_TEXT型の順序付き集合に対して，その集合の要素全てが指定された順序に応じて連結された一つの文書を出力する．但し，この関数は入力がnilの場合，結果はEMPTYとする．

  - `ST_TEXT cat(set of ST_TEXT` *arg1* `)`(内容モデルの定義に用いられる繰り返し演算子“+に相当)
    入力されたST_TEXT型の順序付き集合に対して，その集合の要素全てが指定された順序に応じて連結された一つのST_TEXT型の値を出力する．但し，この関数は入力がnilの場合，結果もnilとする．

- 単項演算子“?”(内容モデルの定義に用いられる選択演算子“?”に相当)
  この演算子は入力がnilのときEMPTYを返す．入力がnil以外のとき入力された値を出力する．

- 与えられたST_TEXT型の複数の値のうちnilでない値を返す演算子“|”
  (内容モデルの定義に用いられる選択演算子“|”に相当)
  この演算子は,“+”同様,結合的でありかつ非可換である.結合法則が成り立つので,多項演算子とみなせる.例えば,$S_1 \mid \cdots \mid S_n$と記述することができる.この演算子は被演算子にnil値以外の値が複数該当する場合,未定義である.

尚,ST^i-式の値を評価した型は,ST^e-式とは異なり,ST_TEXT型であり,XML型の値ではない点に注意されたい.なせならば,結果の値に根エレメントが存在しない場合があるからである.したがって,上記の演算子の適用結果に対して,根エレメントを付加する関数taggedを導入する.

- **XML tagged(ELEMENT** *arg*1, **ST_TEXT** *arg*2)
  ELEMENT型の*arg*1をST_TEXT型の*arg*2の根エレメントに割り当てる.*arg*2がEMPTYの場合,EMPTYを要素とするエレメント*arg*1を根とするST_TEXT型の値が返る.*arg*2がnilの場合,この関数はnilを返す.

これらの関数や演算子を適用した式を**ST^i-式**[4]として以下のように定義する.

定義 4.2.3: EをELEMENT型の値または変数とすると,ST^i-式は次のように再帰的に定義される.

- ST^i-式 := ST^e-式
- ST^i-式 := ST^i-式 + ST^i-式
- ST^i-式 := cat*(ST^i-式)
- ST^i-式 := cat(ST^i-式)
- ST^i-式 := ST^i-式 ?
- ST^i-式 := ST^i-式 | ST^i-式

[4]STはStructured Textの略であり,^{i}はintension(内包)の略である.これは,検索結果の構造化文書の論理構造を定義する機能を表す式であることを意味している.

- ST^i-式 := tagged(E, ST^i-式)

□

ST^i-式を評価した値の型はST_TEXT型となる．

一般に，SQLにおけるGROUP BY句を伴った問合せに関して，SELECT句の選択リスト (select-list) はカラム名のリストとデータベース属性に対する集約関数の適用結果リストから構成されている．この選択リストに出現するカラム名は必ずGROUP BY句に指定されていなければならない．また，GROUP BY句に指定されていないデータベース属性がSELECT句に出現する場合，必ず集約関数が適用されていなければならない [Ram97]．ST_TEXT型固有の関数である，cat*() とcat() は集約関数であるので，GROUP BY句とこれら二つの関数の対応づけは既存のSQLにおけるGROUP BYと集約関数の対応づけに帰着できる．すなわち，我々の問合せにおいてあるGROUP BY句に着目した時，このGROUP BY句に指定されているデータベース属性に対応するエレメントと，指定されていないデータベース属性に対応するエレメント間には1対多関連が成り立つ．したがって，対応するSELECT句にこれら二つのエレメントが出現する場合，GROUP BY句に指定されていないエレメントにはこれら二つの集約関数のうちいずれかが必ず適用されなければならないものとする．

この章で定義したXML型固有の関数や述語を用いた問合せ例を図4.2に示す．この問合せは，図2.1に示すDTDを制約として持つXML型の属性aを有する表aに対する問合せである．この問合せで属性aに保持されている論文から，図4.3に示すDTDに従うXML文書を結果として出力する．この検索結果のXML文書は発表年 (<year>) ごとの著者 (<author>) ごとの論文のタイトル (<title>) から構成されている．この問合せの入力XML文書の順序木表現の例は図4.1に示したとおりであり，出力XML文書の順序木表現の例は図4.4に示す．

ここで，図4.2に示す問合せにおいて，既に述べた次の二つの点に注意されたい．一つは，経路式t.a/<author>に関して，属性aに保持されているXML文書で最も外側にタグ付けされているエレメント名<article>は省略することができる点である．もう一つは，GROUP BY句に指定されているye，auが

```
SELECT tagged(<pub-by-year>, ye+cat*(p))
FROM
   ( SELECT tagged(<pub-by-author>, au+cat*(ti)) AS p, ye
     FROM   t, t.a/<author> AS au, t.a/<title> AS ti,
            t.a/<year> AS ye
     GROUP BY ye, au
     ORDER BY ti )
GROUP BY ye
ORDER BY ye, au
```

図 4.2 問合せ例

```
<!ELEMENT pub-by-year (year,pub-by-author*)>
<!ELEMENT pub-by-author (author,title*)>
<!ELEMENT year     (#PCDATA)>
<!ELEMENT title    (#PCDATA)>
<!ELEMENT author   (#PCDATA)>
```

図 4.3 問合せ結果の DTD

SELECT 句に出現する,

```
SELECT tagged(<pub-by-author>, au+cat*(ti)) AS p, ye
```

において, GROUP BY に指定されていない ti には集約関数cat*() が適用されているという点である.

4.3 提案する言語の記述能力

この節では, 4.2で導入した関数や述語を用いた OQL 表記の問合せ言語の記述能力について記述する. 構造化文書に対する問合せに関して, 形式的な記述能力の基準はまだ存在しない [NBY97]. Sacks-Davis らは構造化文書に対す

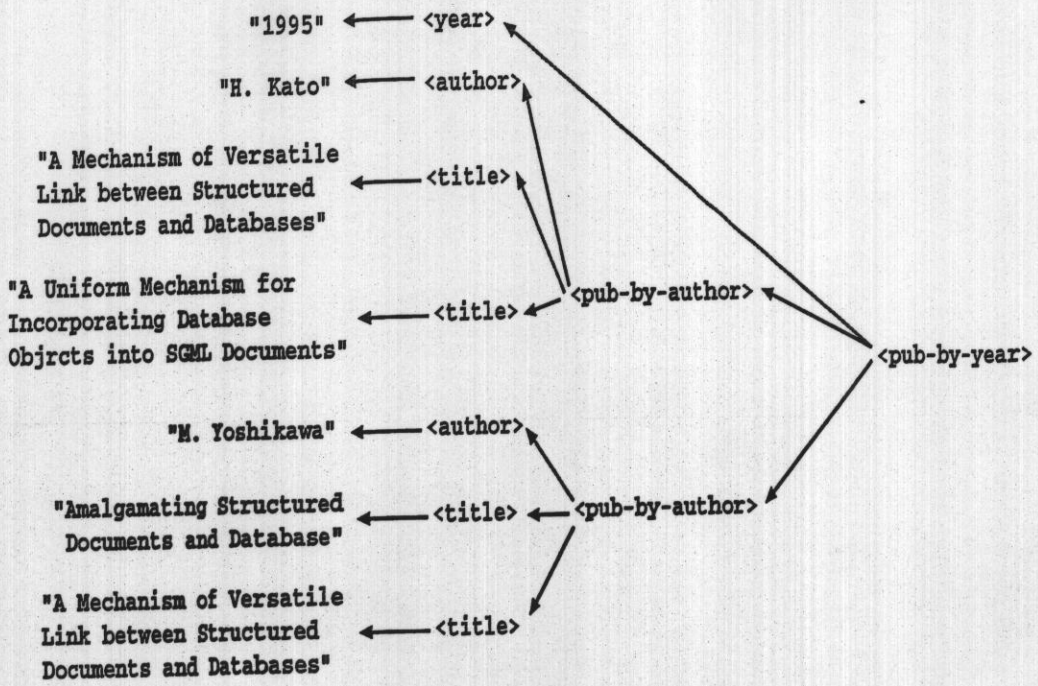

図 4.4 出力 XML 文書の順序木表現の例

る問合せを分類することで，構造化文書を操作する問合せ言語に必要とされる記述能力の非形式的な基準を示した [SDAMZ94]．また，この基準は構造化文書の問合せ言語に関する他の研究でも用いられている [NBY97]．したがって，Navarro ら [NBY97] 同様，Sacks-Davis らが示した非形式的な基準 [SDAMZ94] を用いて，本研究で提案する問合せ言語の記述能力を示す．

Sacks-Davis らは構造化文書を操作する問合せに必要とされる機能を次のように分類した．

(1) 単語によるアクセス (Word-by-word access)
文書中のに出現する単語の指定による問合せのことであり，例えば次のような問合せが考えられる．
「'parallel' を含みかつ，'computing' または 'processing' を含む文書 (<doc>) を検索せよ」

(2) ランク付け問合せ (Ranked queries)
問合せ条件に対して関連度の高い順に結果を返す問合せのことであり，例えば次のような問合せが考えられる．
「並列処理に関する文書を検索せよ」

(3) 部分文書に制限された問合せ範囲 (Query scope restricted to sub-documents)
任意の部分文書に関する問合せ条件を伴った問合せのことであり，例えば次のような問合せが考えられる．
「タイトル (<title>) に 'parallel' と 'processing' を含む文書 (<doc>) を検索せよ.」
「最初の段落 (<para>) に 'parallel' と 'processing' を含む文書 (<doc>) を検索せよ.」

(4) 部分文書の検索 (Retrieval of sub-documents)
文書全体ではなく，部分文書を結果として返す問合せのことであり，例えば次のような問合せが考えられる．
「並列処理に関する節 (<section>) を検索せよ.」
「並列処理に関する関連度が 0.2 以上の段落 (<para>) を含む節 (<section>) を検索せよ.」
「マシンアーキテクチャに関する関連度が 0.5 以上の段落 (<para>) を含む，並列処理に関する節 (<section>) を検索せよ.」
「段落 (<para>) に 'parallel' と 'processing' を含む節 (<section>) を検索せよ.」

(5) 文書の構造によるアクセス (Access by structure of documents)
文書の論理構造に関する問合せ条件を伴う問合せのことであり，例えば次のような問合せが考えられる．
「<article>を親として持つエレメント (部分文書) を検索せよ.」
「子を持つエレメント (部分文書) を検索せよ.」
「第一の子が<title>であるエレメント (部分文書) を検索せよ.」
「<section>内のエレメント (部分文書) を検索せよ.」
「<corres>を含む<doc>を検索せよ.」

「<section>を含む<section>を検索せよ.」

(6) DTD の異なる文書へのアクセス (Access to different types of document)
一つの問合せで DTD の異なる文書にアクセスすることであり，例えば次のような問合せが考えられる.
「タイトルに 'parallel' と 'processing' を含む記事，論文，本を検索せよ.」

(7) XML 属性によるアクセス (Access by XML attribute)
XML 属性に関する問合せ条件を伴う問合せのことであり，例えば次のような問合せが考えられる.
「XML 属性が CONFIDENTIAL=YES である<corres>を検索せよ. 」

(8) XML データと非 XML データを組み合わせたアクセス (Combined access to XML and non-XML data)
文書データと他のデータを統合的に扱う問合せのことであり，例えば次のような問合せが考えられる.
「この記事の全てのタイトルを検索せよ.」
「1994 年 3 月 30 日に挿入された文書を検索せよ.」

以下，[SDAMZ94] で示されている上記の問合せ例が，本研究で提案する問合せ言語でどのように記述されるかを示す．但し，表 t の属性 a に XML 文書が格納されているものとする．なお，次の四つの項目に関しては省略する．検索結果のランキングと XML 属性は本研究の対象外であるので，(2) ランク付け問合せと (7) XML 属性によるアクセスは省略する．また本研究において，(6) DTD の異なる文書へのアクセスは，異なるデータベース属性へのアクセスと等価であり自明なことであるので，省略する．更に，本研究ではデータベース問合せ言語である OQL を採用しているので，データベース中の XML 型以外のデータへのアクセスが可能なのは自明である．したがって，(8) XML データと非 XML データを組み合わせたアクセスも省略する.

(1) 単語によるアクセス

「'parallel' を含みかつ, 'computing' または 'processing' を含む文書 (<doc>) を検索せよ」

```
SELECT   doc
FROM     t.a//<doc> AS doc
WHERE    doc CONTAIN 'parallel'
AND      ( doc CONTAIN 'computing'
OR       doc CONTAIN 'processing' )
```

(3) 部分文書に制限された問合せ範囲

「タイトル (<title>) に 'parallel' と 'processing' を含む文書 (<doc>) を検索せよ.」

```
SELECT   doc
FROM     t.a//<doc> AS doc, doc//<title> AS ti
WHERE    ti CONTAIN 'parallel'
AND      ti CONTAIN 'processing'
```

「最初の段落 (<para>) に 'parallel' と 'processing' を含む文書 (<doc>) を検索せよ.」

```
SELECT   doc
FROM     t.a//<doc> AS doc
WHERE    doc//<para>[1] CONTAIN 'parallel'
AND      doc//<para>[1] CONTAIN 'processing'
```

(4) 部分文書の検索

「段落 (<para>) に 'parallel' と 'processing' を含む節 (<section>) を検索せよ.」

```
SELECT   sec
FROM     t.a//<section> AS sec, sec//<para> AS para
WHERE    para CONTAIN 'parallel'
AND      para CONTAIN 'processing'
```

(5) 文書の構造によるアクセス

「<article>を親として持つエレメント (部分文書) を検索せよ.」

```
SELECT   ele
FROM     t.a//ELE_e AS ele
WHERE    ele include ele/<article>
```

「子を持つエレメント (部分文書) を検索せよ.」

```
SELECT   ele1
FROM     t.a//ELE_e1 AS ele1, t.a//ELE_e2 AS ele2
WHERE    ele1 include ele2
```

「第一の子が<title>であるエレメント (部分文書) を検索せよ.」

```
SELECT   ele
FROM     t.a//ELE_e1 AS ele
WHERE    not exists ELE_e2 : ele/ELE_e2 precede ele/<title>
```

「<section>内のエレメント (部分文書) を検索せよ.」

```
SELECT   a/<section>/ELE_e
FROM     t
```

「<corres>を含む<doc>を検索せよ.」
SELECT doc
FROM *t.a*//<doc> AS doc
WHERE doc include doc//<corres>

「<section>を含む<section>を検索せよ.」
SELECT sec
FROM *t.a*//<section> AS sec
WHERE sec include sec//<section>

4.3.1 文書の構造によるアクセスの拡張

文書の構造に関してSacks-Davisらが示している問合せよりもより複雑な条件によるアクセスを考えることができる．例えば，論理構造の包含関係は経路式で表すことができるが，経路を返す問合せや，正規表現を用いた経路の特定などが考えられる[CCD+98]．これらの問合せを記述するためには，データモデルを拡張する必要がある．最も単純な拡張は，経路式を値とする型を導入することである[CACS94]．しかしながら，この拡張は既に研究されていることでありかつ，この拡張は本研究の主な成果である構造化文書ビューと構造化文書データベースの統合利用に対して，直接影響を及ぼすものではない．したがって，本研究では前述したデータモデルを用いて研究を進める．

第 5 章

構造化文書ビュー

構造化文書をデータベースビューとして構築する場合，通常のデータベースビューと同様に，構造化文書データベースに対する問合せ言語を用いて定義できる．構造化文書を扱う問合せ言語には，既に述べたように，OQL の構文を採用する．同じ構造化文書源から異なる構造化文書ビューが定義できると，利用者独自の文書の構築が可能となる．例えば，利用者の興味に基づく新聞の構築が可能となる．この新聞は，購読者の興味の度合が大きい記事と興味の度合の小さい記事とで，記事の詳細度が異なる新聞のことである．実際，このような新聞の配信が始まっている [SM96].

この章では，構造化文書ビューの定義から文書ビューの DTD を導出する手法と，文書ビューに対する問合せの最適化手法を提案する．

第 4章では，入力 XML 文書から与えられた DTD に従う XML 文書を出力するための関数，演算子，述語を導入した．第 2章で述べたように，XML では DTD の無い文書も許している．DTD の無い XML 文書ビューが定義されている場合でも，あらかじめビューの DTD を知ることで，利用者のビューに対する問合せ記述の支援ばかりでなく，問合せの最適化にも利用できる．

また，ビューに対する問合せは，その問合せ中のビューの出現をビュー定義に置換することで単純に処理することができる．しかしながら，一般にビューに対する問合せの条件式を，ビュー定義の条件式に移すことにより，最適化することが可能である．この問合せ最適化は，問合せ処理手順を木表現した際に，選択条件を先に評価するため木の葉へ移動することから，“selection push-

```
CREATE TABLE articles(
  issue_date DATE,
  reporter   REPORTER,
  headline   XML,
  body       XML,
  contents   VECTOR);
```

```
CREATE TABLE subscribers(
  usr        PERSON,
  interests VECTOR,
  ...        ...);
```

図 5.1 新聞記事と利用者に関するデータベーススキーマ

down"[AHV95] と呼ばれる伝統的な最適化手法である．この章では特に，XML 文書固有の選択条件に着目し，この選択条件を push-down する際の選択条件の書き換え規則を提案する．

データベースビューには，ビューに対する問合せを処理する時に同時にビュー定義を評価する仮想ビュー (virtual views) と，あらかじめビューを作成し問合せ処理はこの作成されたビュー上で行なう具現ビュー (materialized views) がある．本研究では仮想ビューとしての構造化文書ビューに対する問合せ処理に焦点を当てる．ネットワークを介した文書ビューの配信を考慮に入れると，仮想文書は具現ビューとして構築されるものである．しかしながら，利用者は配信された文書を全て保存しないかもしれない．このような状況においては，仮想ビューとしての構造化文書ビューに対する問合せ処理が必要となる．また，ビューの更新に関しては扱わない．このような制限のもとでもこの章の成果は十分有益なものである．

以下，まず準備として，第 4章で導入した XML 型固有の関数や述語を用いた OQL[Cat97] によって，仮想文書としての構造化文書ビューを定義できることを **5.1**で述べる．次に，ビュー定義からの DTD の導出手法について **5.2**で述べる．更に，ビューに対する問合せの最適化手法について **5.3**で述べる．

5.1 構造化文書ビューの構築

構造化文書ビューを構築するには，所望の文書構造と文書内容の双方を特定する必要がある．ここで文書の論理構造は，第 4章で導入した関数，演算子，

```
<!ELEMENT headline (#PCDATA)>
```

図 5.2 表 article の属性 headline の DTD

```
<!ELEMENT body       (abst,detail)>
<!ELEMENT abst       (paragraph+)>
<!ELEMENT detail     (paragraph+)>
<!ELEMENT paragragh (#PCDATA)>
```

図 5.3 表 article の属性 body の DTD

述語を用いると，所望の論理構造を特定できる．次に，文書内容に関しては，既に述べたように文字列の出現の仕方（接頭語検索，近接検索など）による特定方法と，ある（部分）文書全体を何らかの方法で定量化しその値を用いた特定方法の二種類がある．

文字列の出現の仕方による文書内容の特定に関しては，4.1.2節で述べたように全文索引を用いると効率的に実現できる．以下,（部分）文書全体の定量化による文書内容の特定について記述する.（部分）文書全体の定量化の手法の例として,（部分）文書全体を多次元ベクトルで表現する方法がある．これにより,（部分）文書同士の類似度計算は，ベクトル間の類似度計算に帰着できる．この（部分）文書全体の定量化は前処理として行なわれ，問合せ実行時には多次元ベクトル計算を実行する．問合せ時の処理効率を向上させるために，高次元データのための索引を利用する．

なお.（部分）文書全体の定量化の具体的な手法，例えばベクトルモデルを用いるならベクトルの作成方法については議論しない．また，この定量化された値を用いた処理を効率的に実現するための索引手法についても，本論文の範囲外である．すなわち，定量化の手法とは独立の議論を展開する．但し，定量化は前処理で行なわれるものとし，問合せ実行時には定量化された値を用いるものとする．

例として，新聞記事データベースから利用者の興味に基づく新聞の構築について考える．この新聞は，利用者の興味と記事の内容間の関連の度合に依存して詳細度の異なる記事から構成されるものとする．記事の詳細度は次に挙げる

3種類とする.

(1) 見出し(<headline>)と本文(<body>)から構成されており，本文に関しては概要(<abst>)と詳細(<detail>)から構成されている記事(<article>)，すなわち元の記事.

(2) 見出し(<headline>)と本文(<body>)から構成されており，本文に関しては概要(<abst>)だけから構成されている記事.

(3) 見出し(<headline>)だけから構成されている記事.

図5.1に，新聞記事と利用者の情報に関するデータベーススキーマを示す．この表articleのXML型の属性headline及びbodyの制約DTDをそれぞれ図5.2，図5.3に示す．表articlesの属性contentsには記事全体（見出しと本文）の内容を表すベクトル値が保持されているものとする．また，表subscribersの属性interestsには利用者の興味を表すベクトル値が保持されているものとする．関数sim()は，この二つのベクトル値の類似度によって，利用者の興味と記事の内容間の関連の度合を計算するものとする．この関数sim()の結果の値が，閾値*threshold*1以上の場合，閾値*threshold*2以上で閾値*threshold*1以下の場合，閾値*threshold*3以上で閾値*threshold*2以下の場合，に応じて記事はそれぞれ上記(1)，(2)，(3)の詳細度で出力されるものとする．このような，利用者の興味に基づく新聞を定義するOQL表現のビュー定義を図5.4に示す．このOQL表現は，発行日(issue_date)ごとの，ある利用者(someone)の興味に基づく新聞(news)を保持するビューyour_newspaperを定義している．このビューyour_newspaperの属性newsに保持されるXML文書の例を図5.5に示す.

```
CREATE VIEW your_newspaper(issue_date,news) AS
SELECT d, tagged(<your_news>, cat*(a))
FROM
 SELECT article.issue_date AS d,
        tagged(<article>, article.headline+article.body) AS a
 FROM   articles, subscribers
 WHERE  subscribers.usr=someone
   AND  sim(subscribers.interests=articles.contents)
                                          >= threshold1
 UNION
 SELECT article.issue_date AS d,
        tagged(<article>, article.headline
                  +tagged(<body>, article.body/<abst>)) AS a
 FROM   articles, subscribers
 WHERE  subscribers.usr=someone
   AND  sim(subscribers.interests=articles.contents)
                                          < threshold1
   AND  sim(subscribers.interests=articles.contents)
                                          >= threshold2
 UNION
 SELECT article.issue_date AS d,
        tagged(<article>, article.headline) AS a
 FROM   articles, subscribers
 WHERE  subscribers.usr=someone
   AND  sim(subscribers.interests=articles.contents)
                                          < threshold2
   AND  sim(subscribers.interests=articles.contents)
                                          >= threshold3
GROUP BY d
```

図 5.4 利用者の興味に基づく新聞ビューの定義例 1

```
<your_news>
<article>
<headline>
Mitsubishi Electric to develop, sell PowerPC chips
</headline>
<body>
<abst>
Mitsubishi Electric Corp. has agreed to cooperate with IBM Corp.
to sell and develop the next generation of PowerPC chips,
company officials said Monday.
</abst>
<detail>
Following the agreement, Mitsubishi will procure
the 4XX and 602 PowerPC...
...
</detail>
</body>
</article>
...
<article>
<headline>...</hadline>
</article>
...
<article>
<headline>...</headline><body><abst>...</abst><body>
</article>
...
</your_news>
```

図 5.5 構築されるビューの属性 a に保持される XML 文書例

入力 ビュー定義の OQL 表現の問合せ InOQL, 問合せの外延データベースにおいて，XML 型の属性に制約として保持されている DTD の集合 $\mathcal{D}$

出力 ビューの DTD viewDTD(エレメント宣言の並び)

手順

step 1: ビュー定義から UNION や INTERSECT を考慮に入れ，Select-From 式の集合に変換する．

step 2: step 1 の結果である Select-From 式の集合の各要素 sql_i に関して，付録 A の図 7.1 に示す手続き SQL2EDExpList(sql_i, Ans) を実行し，その結果である第 2 引数の $Ans.EDlist$ に保持されるエレメント宣言式の並びをマージして $EementDeclareList$ とする．

step 3: $EelementDeclareList$ 中の重複を取り除く．

step 4: $EelementDeclareList$ 中の各要素について，ele を宣言するエレメント名とし $cont$ をその内容モデルとする，エレメント宣言を viewDTD に書き込む．

step 5: $\mathcal{D}$ 中の該当する (部分)DTD を viewDTD に書き込む．

step 6: 異なる内容モデルを用いて，同じエレメント定義をしているエレメント宣言は，各内容モデルを “|” で結ぶことで，一つのエレメント宣言にする．

図 5.6 ビュー定義文から DTD の導出アルゴリズム **deriveDTD**

5.2 DTD の導出

XML では DTD の無い文書も許されている．この節では事前に DTD を定めずに XML 文書を定義しているビュー定義からの DTD の導出について記述

する．XML 文書ビューの DTD を導出することにより，ビューの論理構造が明確になり利用者のビューに対する問合せ記述の支援が可能となるばかりでなく，ビューに対する問合せの最適化にも役立つ．但し，ここでいう DTD の導出とはエレメント宣言の並びの導出のことである．既に述べたように，本研究では XML 属性については扱わない．

なお，提案するアルゴリズムを用いて導出される DTD には，非決定的な内容モデルが含まれる．非決定的な内容モデルとは，正規表現を有限オートマトンに変換する McNaughton-山田アルゴリズム ([ASU86] の 3.9 章のアルゴリズム 3.5) のような標準的なアルゴリズムを用いたとき，構文木のある終端ノードに対応する FOLLOW 集合の複数の要素が同じエレメントである場合をいう．しかしながら，SGML との互換性を維持しない XML においては，非決定的な内容モデルは許されいる．実際，現在最も普及している XML 解析器である “XML for Java”[IBM98] は，非決定的な内容モデルが含まれる DTD を扱っている．

5.2.1 DTD 導出の意義

構造化文書ビューの DTD をあらかじめ知ることがビューに対する問合せ処理の最適化にどのように役に立つかを説明する．一般にデータベースに対する問合せは次のような段階を経て実行される [Gra93].

(a) 構文解析 (Parsing)
利用者によって入力または，応用によって生成された，SQL のような高水準の言語で記述された問合せは，構文解析され内部形式へと変換される．

(b) 問合せの妥当性の検査 (Query Validation)
(内部形式の) 問合せがデータベース中に存在するオブジェクトを参照していることを，メタデータを用いて検証する．

(c) ビュー名の出現の解消 (View Resolution)
もし，(仮想) ビューに対する問合せである場合，ビューによって提供されているマクロや一貫性制約を問合せに統合する．最も伝統的な手法の

一つは，問合せ中のビュー名の出現をビュー定義に置換することによる問合せの展開 (query expand) である．

(d) 最適化 (Optimization)
(展開された) 問合せは最適化機構により，データベースに格納されたオブジェクトに対する直接的な一連の操作として最適な実行計画に変換される．

(e) 実行計画の変換 (Plan Compilation)
実行計画はデータベース問合せ実行エンジンによって実行可能なコードに変換される．

(f) 実行 (Execution)
データベース問合せ実行エンジンによって問合せが実行される．

問合せ処理システムは，構造化文書ビューのDTDをあらかじめ知っておくことで，ビューに対する問合せのうちビュー上に存在しない論理構造に対する問合せは，上記の (b) の段階でデータベース問合せ処理機構によって処理することができる．これに対して文書ビューのDTDがわからないと，この種の問合せは上記の (c) の段階もしくは上記の (f) の段階で処理される．

例えば，図1.1中の文書ビュー3のような構造だけからなる文書集合に対する以下のような問合せについて考える．

> <article>に含まれる<body>中に文字列'computer' が含まれる<article>を検索せよ．

この問合せは，問合せ処理システムが図1.1中の文書ビュー3のDTDをあらかじめ知っている場合，上記の問合せ処理段階 (b) で問合せ処理が完了する．すなわち，問合せの妥当性の検査において，この文書のDTDを参照することで，<article>に含まれる<body>は存在しないことがわかる．したがって，この問合せは妥当ではないので，上記の問合せ処理段階 (b) で問合せを処理することができる．これに対して，問合せ処理システムがあらかじめDTDを知らないとき，問合せに該当するデータがないにもかかわらず最悪の場合，上記の問合せ処理段階 (f) で実データに対する検索を実行してしまう．

したがって，DTD を持たない構造化文書ビューであっても，あらかじめ DTD を知っておくことによって，問合せ処理の最適化につながる．

5.2.2 関連研究との比較

Papakonstantinou ら [PV99] は，XML 固有の問合せ言語 XML-QL[DFF+98] によって記述されたビュー定義からビューの DTD を導出するアルゴリズムを提案している．彼らのアルゴリズムと我々のものを比較すると，アルゴリズムの入力と出力にそれぞれ一長一短がある．アルゴリズムの入力対象である問合せに関しては，彼らのアルゴリズムが対象とする問合せは非常に限定されたものであるのに対して，我々のアルゴリズムが対象とする問合せは *select-from-where* 形式の OQL[1] で記述可能な任意の問合せである．一方，アルゴリズムが出力する DTD の正確さに関しては，彼らのアルゴリズムは最もきつい[2]DTD を導出するのに対して，我々のアルゴリズムは常にきつい DTD を導出するとは限らない．以下，Papakonstantinou らのアルゴリズムとの比較を，対象とする言語，対象とする問合せの範囲，導出される DTD の正確さの三つの点についてそれぞれ記述する．

対象とする言語

Papakonstantinou らのアルゴリズムは対象とする言語は，XML 文書だけを操作対象とする XML-QL であるのに対して，この節で提案するアルゴリズムが対象とする言語は，XML 文書だけでなく，様々なデータ型の値を操作対象とするデータベース言語 OQL である．

データベースや人工知能の分野では，異種分散環境の統合に関する多数の研究が存在する [CGMH+94, PGGMU95]．Papakonstantinou らは異種分散環境における共通モデルとして XML 文書を採用し，仲介者 (mediator) は XML-QL を用いた問合せによって異種分散環境上のデータを統合することを提案している．このとき，XML-QL で記述された問合せから DTD をあらかじめ導出することによって，応用からの仲介者に対する問合せの処理を最適化することが

[1] OQL には，select-from-where 形式以外の問合せ形式もある．

[2] " きつさ" に関してはのちほど定義する．

可能となる．したがって，Papakonstantinou らが提案する DTD 導出のアルゴリズムが対象とする問合せ言語は，XML 文書だけを操作する言語である．

これに対して，既に述べたように本研究ではデータベース中に格納されている XML 文書を用いた，データベースビューとしての構造化文書ビューの構築について考えている．データベースビューとしての構造化文書ビューは，XML 文書だけを操作対象とする言語ではなく，データベース中に管理されている様々なデータ型の値を操作対象とするデータベース言語によって定義されるものである．したがって，この節で提案する DTD 導出のアルゴリズムが対象とする問合せ言語は，データベース言語の一つである OQL である．

対象とする問合せの範囲

Papakonstantinou らが提案するアルゴリズムが対象としている問合せは，一つの情報源（一つの DTD に対して妥当な文書集合）の射影（projection）としてのビュー定義に制限されている．これに対して，この節で提案するアルゴリズムが対象としている問合せは，任意の *select-from-where* 形式の OQL である．

Papakonstantinou らの提案するアルゴリズムが対象としているのは，彼らが「*pick-element* XML-QL 問合せ」と呼ぶ問合せである．この問合せは，SELECT 句にはエレメントに束縛 (bind) される変数が一つだけ出現し，WHERE 句には一つの情報源に対する一つの条件だけが出現するものである．これは，文書中の部分文書を抽出するだけの問合せに限定しており，ビュー定義というよりは射影（projection）とみなすことができる．よって，このアルゴリズムの特徴は問合せの WHERE 句の条件を導出する DTD に反映させるところにある．

Papakonstantinou らのアルゴリズムが対象とする問合せが部分文書の抽出という狭い意味でのビュー定義であるのに対して，この節で提案するアルゴリズムは任意の select-from-where 形式の OQL のを対象としている．したがって，部分文書の抽出のみならず結果の論理構造の再構成も含む，一般的なビュー定義を対象としている．換言すれば，我々のアルゴリズムが対象とする問合せは，Papakonstantinou らのアルゴリズムが対象とする問合せを包含しているといえる．後で述べるように，我々のアルゴリズムの特徴は，問合せの SELECT 句中に出現する新たなエレメントを定義するコンストラクタに着目して導出す

る DTD に反映させるところにある．

DTD の正確さ

Papakonstantinou ら [PV99] は，与えられた XML 文書集合に対する DTD の正確さ (precision) に関して，“tightness” という尺度を提案している．彼らは，[PV99] において,「ある DTD Dに対して妥当な任意の XML 文書が，ある DTD D'に対しても妥当であるとき，Dは D'より “きつい (tighter)” という．」と定義している．Papakonstantinou らのアルゴリズムは，与えられたビュー定義と情報源の DTD を用いて，“最もきつい (*tightest*)”DTD を導出するのに対して，我々のアルゴリズムは常に “最もきつい (*tightest*)”DTD を導出するとは限らない．

5.2.3 提案するアルゴリズムの概要

提案するアルゴリズムの要点を以下に示す．XML 文書ビューの論理構造は，ビュー定義文中に出現する tagged() 関数によって構築されている．一つの tagged() 関数はその定義より，DTD における一つのエレメント宣言に対応するものである．したがって，ビュー定義文中に現れる tagged() 関数を対応するエレメント宣言に変換することで，DTD を導出することができる．このビュー定義から DTD を導出するアルゴリズム **deriveDTD** を図 5.6に示す．但し，一つのエレメント宣言をエレメント宣言式として以下のように定義する．

エレメント宣言式 := エレメント名 内容モデルの正規表現

このアルゴリズム **deriveDTD** の概要は，次の通りである．ビュー定義中の tagged() 関数の第 1 引数を宣言するエレメント名とし，第 2 引数を子エレメント名から構成される内容モデルの正規表現に変換することで，着目している tagged() をエレメント宣言に変換することができる．ここで，**定義 4.2.3** より，tagged() 関数の第 2 引数には ST^i-式が指定されているので，ST^i-式を内容モデルの正規表現に変換する必要がある．基本的には，この ST^i-式中のデータベース属性の出現を該当するエレメント名（から構成される正規表現）に変換することで，内容モデルの正規表現を求めることができる．既に述べたよう

に，tagged() 関数は新たなエレメントを構築するコンストラクタであるので，OQL 中では必ず SELECT 句に出現する．したがって，tagged() 関数で用いられているデータベース属性は，必ずその SELECT 句に対応する FROM 句に存在している．このことを考慮に入れ，tagged() 関数の第 2 引数に出現するデータベース属性に該当するエレメント名（から構成される正規表現）を特定するには，以下のような手順となる．

(1) 着目しているデータベース属性が対応する FROM に必ず出現しているが，ビュー定義が入れ子 OQL で記述されている場合，FROM 句には SELECT 句とそれに対応する FROM 句が出現していることになる．この場合，着目しているデータベース属性は，この入れ子の SELECT 句において定義されている属性かもしれない．そこで，着目しているデータベース属性に関して，対応しているデータベース属性を入れ子の最も内側までたどると，外延データベースの属性にたどりつく．

(2) データベース属性が外延データベースの属性であれば，その属性に制約として保持されている DTD 中の該当するエレメント名に変換する．

このように，基本的にビュー定義における SELECT 句と FROM 句だけに着目すれば良い．したがって，ビュー定義を SELECT 句と FROM 句だけから構成される *Select-From* 式に変換する．Select-From 式は次のような形式である．但し，“[” と “]” で囲まれた部分は省略可能である．この Select-From 式は OQL の文法の範囲を越えていない．

```
Select-From式 := SQL
SQL := SELECT S1 [AS A1], S2 [AS A2], ..., Sm [AS Am]
       FROM   F1, F2, ..., Fn
Fj  := SQL
```

また，UNION 演算子や INTERSECT 演算子によって結びつけられている SELECT 句は，別の Select-From 式とする．したがって，一般に，select-from-where 形式の OQL で記述された任意のビュー定義は Select-From 式の集合に変換することができる．

Select-From式は入れ子構造をしており，提案するアルゴリズムは最も外側のSELECT句の各選択リスト項目 (select list item) に関して，入れ子の内側のSELECT句の対応する選択リストを最も内側のFROM句までたどることで，DTDを導出する．したがって，入力Select-From式のSELECT句に出現する選択リスト項目の数の最大数を m，入れ子の深さ n とした場合，このアルゴリズムの計算量は $O(mn)$ である．

このアルゴリズムの実現には，そのビュー定義中に出現するデータベース属性が，外延データベースにおけるXML型の属性なのか，ビュー定義文中で定義されているデータベース属性なのかを識別する必要がある．そのためには，OQL解析器とXML解析器，tagged() 関数の解析器を組み合わせる必要がある．

このアルゴリズム **deriveDTD** を用いて，図 5.4に示したビュー定義からDTDを導出する例を以下に示す．

step 1: 入力であるビュー定義はUNION演算子で結合されているので，次のような三つのSelect-From式から成る集合に変換される．

```
SELECT d, tagged(<your_news>, cat*(a))
FROM
 SELECT article.issue_date AS d,
        tagged(<article>,
                 article.headline+article.body) AS a
 FROM   articles, subscribers
---
SELECT d, tagged(<your_news>, cat*(a))
FROM
 SELECT article.issue_date AS d,
        tagged(<article>, article.headline
          +tagged(<body>, article.body/<abst>)) AS a
 FROM   articles, subscribers
---
SELECT d, tagged(<your_news>, cat*(a))
FROM
 SELECT article.issue_date AS d,
        tagged(<article>, article.headline) AS a
 FROM   articles, subscribers
```

step 2: **step 1** の結果である三つのSelect-From式それぞれを第1引数として，図7.1に示す手続きSQL2EDExListを実行した結果の *ElementDeclareList* は以下のようになる．

```
<your_news> <article>*
<article> <headline>,<body>
<your_news> <article>*
<article> <headline>,<body>
<body> <abst>
<your_news> <article>*
<article> <headline>
```

step 3: 重複を取り除くと次のようになる.

```
<your_news> <article>*
<article> <headline>,<body>
<body> <abst>
<article> <headline>
```

step 4: viewDTD は以下のようになる.

```
<!ELEMENT your_news (article*)>
<!ELEMENT article   (headline,body)>
<!ELEMENT body      (abst)>
<!ELEMENT article   (headline)>
```

step 5: 図 5.3と図 5.2に示す DTD のうち，該当する (部分)DTD をviewDTD に書き込む.

```
<!ELEMENT your_news (article*)>
<!ELEMENT article   (headline,body)>
<!ELEMENT headline  (#PCDATA)>
<!ELEMENT body      (abst,detail)>
<!ELEMENT abst      (paragraph+)>
<!ELEMENT detail    (paragraph+)>
<!ELEMENT paragragh (#PCDATA)>
<!ELEMENT body      (abst)>
<!ELEMENT article   (headline)>
```

step 6: 異なる内容モデルを用いて，同じエレメント定義をしているエレメント宣言は，各内容モデルを“|”で結ぶことで，一つのエレメント宣言にする．

```
<!ELEMENT your_news (article*)>
<!ELEMENT article   (headline,body) | (headline)>
<!ELEMENT headline  (#PCDATA)>
<!ELEMENT body      (abst) | (abst,detail)>
<!ELEMENT abst      (paragraph+)>
<!ELEMENT detail    (paragraph+)>
<!ELEMENT paragragh (#PCDATA)>
```

5.3 XML 文書ビューに対する問合せの最適化

この節では，XML 文書ビューに対する問合せ処理に対して，データベースの問合せ処理における伝統的な最適化の手法の一つを適用する．ビューに対する問合せは，その問合せ中のビュー名の出現をビュー定義に置換することで，単純に処理することができる．しかしながら，一般にビューに対する問合せの条件式を，ビュー定義の条件式に移すことにより，最適化することが可能であ

る．この問合せ最適化は，問合せ処理手順を木表現した際に，選択条件を先に評価するため木の葉へ移動することから，“selection push-down”[AHV95] と呼ばれる伝統的な最適化手法である．

この節では特に XML 文書固有の選択条件である，あるエレメント内に出現する文字列の特定に関する選択条件に着目し，この選択条件を push-down する際の書き換え規則を提案する．“selection push-down” に関する，既存の関係質問の書き換えについては省略する．

ビューに対する問合せにおいて，あるエレメント Eが指定されている時，この Eがビュー定義によって定義されたエレメントか，外延データベースにおいて，既に定義されているエレメントかによって，この選択条件を push-down する際の書き換え規則が異なってくる．この書き換え規則について以下の **5.3.1** で提案する．

また，ビューに対する問合せが，外延データベース中のエレメントの抽出である場合に，ビューに対する問合せ条件によっては，ビュー定義文中の処理を全く実行する必要がない問合せに変換することが可能となることがある．この問合せ変換規則については，**5.3.2**で提案する．

5.3.1 構造化文書固有の選択条件の評価に関する最適化

この節では，構造化文書固有の選択条件の評価に関する最適化に着目する．簡単のために，検索結果は定義されているビューの XML 型の属性値全体に固定する．すなわち，$viewname$ をビュー名，a をビュー $viewname$ の XML 型の属性，$conds$ を選択条件とすると，

$$\pi_{(a)}\sigma_{(conds)}(viewname)$$

形式の問合せに限定する．この $conds$ 中の条件式で，XML 文書固有の選択条件である，属性 a のエレメント E内の文字列の出現に関する条件 $cond_{str}$ について考える．すなわち，$cond_{str}$ の形式が

$$E \text{ CONTAIN } w^3$$

[3]述語 CONTAIN は第 4.1.2節で導入した．

に着目する．

伝統的な最適化の手法の一つに，選択条件を先に評価する “selection push-down”[AHV95] と呼ばれる手法がある．選択条件を先に評価することで，問合せ処理における中間結果の数を減らすことができる．これにより処理対象数が減少し，ほとんどの場合，処理全体の効率の向上につながる．

同様に，ビューに対する問合せに出現する条件式 $cond_{str}$ をビュー定義文の条件式に push-down することを考える．但し，伝統的な selection push-down がヒューリスティックな手法であるのと同様に，この選択条件の push-down により，必ずしも最適な問合せに変換されるとは限らない．

この節で提案する書き換え規則における新しい側面は，集約を伴う問合せの書き換え規則にある．集約を伴う問合せの最適化に関する研究はこれまでにもいくつか存在する [Cha98]．構造化文書を対象とした集約とこれまで最適化問題で扱ってきた集約との大きな違いは，集約の要素の保存性にある．例えば，数値型の値の集合の平均を計算する集約関数 Avg() の結果には，元の要素は保存されていない．したがって，定義中に従来型の集約が施されているビューに対して，利用者は元の要素に関する問合せはできない．これに対して，**4.2.2** で導入した構造化文書の集合に対する集約 cat() または cat*() の結果には，元の要素が保存されている．したがって，定義中にこれらの集約が施されているビューに対して，利用者は元の要素に関する問合せを記述することができる．この種の集約は内部に構造を有する型固有の集約とみなすことができる．

以下に，選択条件 E `CONTAIN` w を push-down する際の書き換え規則を示す．

(1) E がビュー定義文において tagged() 関数によって定義されているエレメントである場合．

$cond_{str}$ は，E の内容モデルを構成している各エレメントに対する条件式に分配され，一つ内側の選択条件に追加することができる．これを繰り返して最も内側の (inner-most) 選択条件にまで push-down することができる．但し，条件式の分配は，着目している tagged() 関数の第二引数に出現する第 4.2.2節で導入した演算子や関数の種類によって，その分配のされかたが以下のように決定される．但し，a, b は ST^i-式とする．

- 選択演算子が適用されている場合

- $a|b$ の場合
  $cond_{str}$は，a CONTAIN w OR b CONTAIN w に変換される．(規則 1.1)
- $a?$の場合
  $cond_{str}$は，a CONTAIN wに変換される．(規則 1.2)

- 列演算子+が適用されている場合 $(a+b)$
  $cond_{str}$は，a CONTAIN w OR b CONTAIN w に変換される．(規則 1.3)
- 集約関数 cat() または cat*() が適用されている場合 (cat(a) または，cat*(a))
  $cond_{str}$が変換されるだけでなく，問合せ全体が以下のように変換される．(規則 1.4)
  tagを集約関数 cat(a) または，cat*(a) が第二引数に現れる tagged() 関数とし，ビュー定義を$\pi_{(tag\ \mathtt{AS}\ att)}\sigma_{(conds)}(edb)GROUP\ BY glist$ とすると，このビューに対する問合せ

$$\pi_{(att)}\sigma_{(att\ \mathtt{CONTAIN}\ w)}(\pi_{(tag\ \mathtt{AS}\ att)}\sigma_{(conds)}(edb))GROUP\ BY glist$$

  は，以下のように変換される．

$$\pi_{(tag\ \mathtt{AS}\ att)}\sigma_{(conds)}(edb \bowtie \pi_{(glist)}\sigma_{(a\mathtt{CONTAIN}\ w)})GROUP\ BY glist$$

(2) Eがビュー定義文において外延データベースの XML 型の属性に関する ST^e-式 st_{ex}に対応する場合．
Eを st_{ex}に変換した条件式 $cond'_{str}$を外延データベースに対する選択条件に push-down することができる．(規則 2)

ここで，上記の (1) の特定は，ビュー定義文に Eに対応するエレメント名を第一引数とする tagged() 関数が出現し，かつ第二引数に第 4.2.2節で導入した演算子や関数が適用されているかどうかを判定すれば良い．これに該当しない場合は (2) の規則を適用すれば良い．しかしながら，この判定は OQL 解析器と XML 解析器，tagged() 関数の解析器を組合せないと実現することができな

い．これにより，関係 ADT と XML ADT の独立性を維持することは困難となる．

以下，この選択条件の書き換え例を示す．まず，図 5.7に示す定義によるビューに対する問合せとして，図 5.8に示す二つの問合せ **Q1**，**Q2**，について考える．このうち **Q1**をナイーブに実行する代数表現を図 5.9に示す．この代数表現に従って問合せを実行することは，“Clinton” を本文に含まない記事も最初に検索されてしまう．そこで，“Clinton” を本文に含む記事だけを最初に検索するように，**Q1**中の条件式

`article/ < body >  CONTAIN “Clinton”`

に着目する．OQL 解析器と tagged() 関数の解析器を組合せることで，この `article/<body>`が外延データベースの body 属性に関する ST-^{e}式に対応することがわかる．したがって，**Q1**は先ほどの条件式 $cond_{str}$の書き換え規則 (2) に従い，図 5.10に示した代数表現に変換される．

また，**Q2**に関しては条件式，

`article CONTAIN “Clinton”`

において，`article`はビュー定義文中で tagged() によって定義されており，この tagged() 関数の第二引数は列演算子 “+” によって構成されているので，$cond_{str}$の書き換え規則 (1.3) により図 5.11に示す問合せに変換される．

次に，集約関数が出現する場合の例，すなわち書き換え（規則 1.4）の適用例を示す．簡単のために，図 5.12に示すビュー view1 に対する問合せとして図 5.13に示す **Q3** について考える．この問合せ **Q3**をナイーブに実行すると，全ての記事の見出しを発行日 (issue_date) 毎に結合した結果に対して，“Clinton” が出現するかどうかの条件式の評価を行ない，検索結果を出力する．これに対して，上で述べた (規則 1.4) の適用によって変換された，図 5.14に示す問合せの実行手順は，まず “Clinton” が見出しに出現する記事の発行日を特定し，その日付けの見出しを集約関数によって結合することで，検索結果を出力する．

更に，構造化文書固有の問合せには，あるエレメント中に出現する文字列の順序関係や近接関係を特定する条件式がある [SDAMZ94]．このような条件式がビューに対する問合せで用いられた場合において，ビューにおける一つの

エレメントが外延データベースにおいては複数のXML型のデータベース属性にまたがる場合がある．このような場合でも，データベース属性間の順序がDTDで特定されている場合は，この順序関係と既存の全文索引手法とを組み合わせることで，効率的に問合せを処理することができる．説明のために，図5.7に示した定義で構築されるビューに対して，二つの文字列s1,s2がs1,s2の順番で同時に出現する<article>を特定する処理は，転置索引を用いることでs1,s2の出現位置を特定することができるので，以下の二つの処理に分解することができ，それぞれを実行することで効率的に実現できる．但し，単語の出現位置やエレメントの出現位置は，所属する表中のレコードIDとリージョンの対で管理されているものとする．

- 同一文書フラグメント内でs1,s2の順に出現しており，かつそのフラグメントが<headline>または<body>に相当していることを特定する．
- s1が<headline>に相当する文書フラグメントに，s2が<body>に相当する文書フラグメントにそれぞれ出現しており，かつそれぞれの文書フラグメントが同一の<article>を構成していることを特定する．

ただし，順序が特定されていない文書フラグメントの集合に対するこの種の問合せに関する最適化手法については，その問合せの意味的妥当性も含めて，今後の課題である．

5.3.2 外延データベース中のエレメントの抽出に関する最適化

この節では，ビューに対する問合せのうち，外延データベースで定義されているエレメントを抽出する問合せに関する最適化について記述する．すなわち，*viewname*をビュー名，*st*を外延データベースで定義されているエレメントを特定するST^e-式，そして*conds*を外延データベースで定義されているエレメントや属性に関する選択条件からのみ構成されている条件式とすると，

$$\pi_{(st)}\sigma_{(conds)}(viewname)$$

形式の問合せに限定する．この種の問合せにおいては，ビュー定義中に出現するtagged()関数は実行する必要がなく，ビューに対する問合せの選択条件

$cond$ を外延データベースに対する選択条件 $cond'$ に変換し，選択リストの要素 st は該当する外延データベースのデータベース属性に関する ST^e-式 st' に変換することで，検索結果を計算することができる．

例えば図 5.7で定義されているビュー対する問合せ例として，図 5.15に示す問合せ **Q4** について考える．この問合せをナイーブに実行する代数表現を図 5.16に示す．この代数表現において，選択リストの要素，article/<headline> は ST^e式であり，かつ外延データベース articles の headline 属性で定義されているエレメントである．また，選択条件式 issue_date=someday は外延データベース articles で定義されているデータベース属性 issue_date に関する条件式である．したがって，上で述べたように，選択条件式 issue_date=someday を外延データベースに対する選択条件式とし，選択リストの要素を該当する外延データベースのデータベース属性に関する ST^e-式である headline とすることで，検索結果を計算することができる．この変換された問合せを図 5.17に示す． したがって，図 5.7のビュー定義中の tagged() 関数を実行することなく，外延データベース上でビュー定義中の条件式とビューに対する問合せの条件式を評価することで，headline 属性を出力結果とすれば良い．

但し，この変換の実現にはビューの論理構造が外延データベースで定義されているのか，ビュー定義中で tagged() 関数によって定義されているのかを判断するために，OQL 解析器と XML 解析器，tagged() の解析器の三つを組合せる必要がある．

更に問合せ結果がビューのある論理構造の部分を用いて別の論理構造を構築する場合などは，より複雑な問合せ変換も考えられる．しかしながら，このような結果の論理構造の構築に着目した問合せの変換については，今後の課題とする．

```
CREATE VIEW articleview(issue_date, article) AS
SELECT article.issue_date,
       tagged(<article>, article.headline+article.body)
FROM   articles, subscribers
WHERE  subscribers.usr=someone
  AND  sim(subscribers.interests=articles.contents)
                                        >= threshold1
UNION
SELECT article.issue_date,
       tagged(<article>, article.headline
                        +tagged(<body>, article.body/<abst>))
FROM   articles, subscribers
WHERE  subscribers.usr=someone
  AND  sim(subscribers.interests=articles.contents)
                                        < threshold1
  AND  sim(subscribers.interests=articles.contents)
                                        >= threshold2
UNION
SELECT article.issue_date,
       tagged(<article>, article.headline)
FROM   articles, subscribers
WHERE  subscribers.usr=someone
  AND  sim(subscribers.interests=articles.contents)
                                        < threshold2
  AND  sim(subscribers.interests=articles.contents)
                                        >= threshold3
```

図 5.7 ビュー定義例

Q1:

SELECT article
FROM articleview
WHERE article/<body> CONTAIN "Clinton"

Q2:

SELECT article
FROM articleview
WHERE article CONTAIN "Clinton"

図 5.8 ビューに対する問合せ例

$\pi_{\text{(article)}}\sigma_{(article/\text{<body> CONTAIN "Clinton"})}$

$(\pi_{\text{(issue\_date,tagged(<article>, headline+body) AS article)}}$

$\sigma_{(relvalue>theshold1)}(tmp1)$

$\cup$

$\pi_{\text{(issue\_date,tagged(<article>, headline+tagged(<body>, body/<abst>)) AS article)}}$

$\sigma_{(relvalue\leq theshold1 \text{ AND } relvalue\geq theshold2)}(tmp1)$

$\cup$

$\pi_{\text{(article.issue\_date,tagged(<article>, articles.headline) AS article)}}$

$\sigma_{(relvalue\leq theshold2 \text{ AND } relvalue\geq theshold3)}(tmp1))$

但し,

$tmp1 = \sigma_{subscribers.usr=someone}(articles \times subscribers)$

$relvalue = sim(subscribers.interests, articles.contents)$

図 5.9 ナイーブな問合せ代数表現

$\pi_{\text{(article)}}$

$(\pi_{\text{(issue\_date,tagged(<article>, headline+body) AS article)}}$

$\sigma_{(relvalue>theshold1 \text{ AND body CONTAIN } \text{“Clinton”})}(tmp1)$

$\cup$

$\pi_{\text{(issue\_date,tagged(<article>, headline+tagged(<body>, body/<abst>)) AS article)}}$

$\sigma_{(relvalue \leq theshold1 \text{ AND } relvalue \geq theshold2 \text{ AND body/<abst> CONTAIN } \text{“Clinton”})}$

$(tmp1)$

$\cup$

$\pi_{\text{(issue\_date,tagged(<article>, headline) AS article)}}$

$\sigma_{(relvalue \leq theshold2 \text{ AND } relvalue \geq theshold3)}(tmp1))$

但し,

$tmp1 = \sigma_{subscribers.usr=someone}(articles \times subscribers)$

$relvalue = sim(subscribers.interests, articles.contents)$

図 5.10 変換された問合せ **Q1**

π(article)

(π(issue_date,tagged(<article>, headline+body) AS article)

σ(*cond*1 AND (headline CONTAIN "Clinton" OR body CONTAIN "Clinton"))($tmp1$)

∪

π(issue_date,tagged(<article>, headline+tagged(<body>, body/<abst>)) AS article)

σ(*cond*2 AND (healine CONTAIN "Clinton" OR body/<abst> CONTAIN "Clinton"))

($tmp1$)

∪

π(issue_date,tagged(<article>, headline) AS article)

σ(*cond*3 AND healine CONTAIN "Clinton")($tmp1$))

但し,

$cond1 = relvalue > theshold1$

$cond2 = relvalue \leq theshold1$ AND $relvalue \geq theshold2$

$cond3 = relvalue \leq theshold2$ AND $relvalue \geq theshold3$

$tmp1 = \sigma_{subscribers.usr=someone}(articles \times subscribers)$

$relvalue = sim(subscribers.interests, articles.contents)$

図 5.11 変換された問合せ **Q2**

```
CREATE   VIEW healineview(issue_date, headlines) AS
SELECT   issue_date,
         tagged(<head-by-date>, cat(headline)) AS headlines
FROM     articles
GROUP BY issue_date
```

図 5.12 ビュー定義例

Q3:

SELECT headlines
FROM headlineview
WHERE headlines CONTAIN "Clinton"

図 5.13 ビューに対する問合せ例 **Q3**

$\pi_{tagged(\text{<head-by-date>},\text{cat(headline)})}$
$(articles \bowtie \pi_{(issue\_date)}\sigma_{(\text{headline CONTAIN "Clinton"})}(articles))$
$GROUP\ BY$issue_date

図 5.14 変換された問合せ例 **Q3**

Q4:

SELECT article/<headline>
FROM articleview
WHERE issue_date = someday

図 5.15 ビューに対する問合せ例 **Q4**

$\pi_{(\text{article/<headline>})}\sigma_{(\text{issue\_date=someday})}(\text{articleview})$

図 5.16 問合せ例 **Q4** をナイーブに実行する代数表現

$\pi_{\text{(headline)}}\sigma_{(cond1 \text{ AND } \text{aricles.date=someday})}(tmp1)$

$\cup$

$\pi_{\text{(headline)}}\sigma_{(cond2 \text{ AND } \text{aricles.date=someday})}(tmp1)$

$\cup$

$\pi_{\text{(headline)}}\sigma_{(cond3 \text{ AND } \text{aricles.date=someday})}(tmp1)$

但し,

$cond1 = relvalue > theshold1$

$cond2 = relvalue \leq theshold1 \text{ AND } relvalue \geq theshold2$

$cond3 = relvalue \leq theshold2 \text{ AND } relvalue \geq theshold3$

$tmp1 = \sigma_{subscribers.usr=someone}(articles \times subscribers)$

$relvalue = sim(subscribers.interests, articles.contents)$

図 5.17 変換された問合せ **Q4**

第 6 章

オブジェクトリンクを有する構造化文書モデル

この章では，構造化文書をデータベースで管理する際の概念モデルを提案する．この概念モデルは，文書中の文字列とその文字列が表すデータベース中の意味データ（本研究では注釈データと呼ぶ）間に構築されるオブジェクトレベルのリンク機構を有する文書モデルである．この概念モデルを抽象データ型としてデータベースに導入することにより，文書の文字列のみならずその意味に基づく問合せが可能となる．この抽象データ型に対しては，単純でかつ既存の文書モデルに対して最小の拡張で済むように，単純な構造と実用的な記述能力を持つ操作体系を与えることにした．

この章の残りは，次のような構成である．**6.1**でこの章で導入する概念モデルの必要性についてその動機を述べる．**6.2**ではこの概念モデルの導入による利点を例を挙げて説明する．この概念モデルを抽象データ型としてデータベースで管理することについては**6.3**で論じる．**6.4**ではデータベース内部表現から概念モデルへのビュー作成も含めた問合せについて記述する．概念モデルである文書ビューに対する問合せがどのように内部表現に対する問合せとして処理されるかについては**6.5**で述べる．最後に，**6.6**は議論と今後の課題について述べる．

6.1 動機

構造化文書をデータベースで管理することによる利点の一つに，データベース中の文書データと他のデータとの統合利用が挙げられる．すなわち，文書以外のデータ間の計算に基づく文書データの検索や，文書処理に基づく文書以外のデータの検索が可能となる．例として新聞記事と企業データに関する以下のような問合せが考えられる．

- 日本の企業名が本文に出現する記事の見出しの検索
- 1997年8月9日の記事の本文に出現する企業の国籍の検索

この問合せはこれまで提案された多くの構造化文書データベースでも記述することが可能である．例えば，Yanらのシステム[YA94]では，オブジェクト指向データベースと文書検索システムの統合を達成しており，T/RDBMS[BCD+95]でも同様の問合せができることを示している．

しかしながらこれまで提案されたシステムでは，文書データと文書以外のデータの統合は文字列パターンマッチに基づいており，限界がある．すなわち，i) 同じ文字列が異なる対象を表す場合や，ii) 同じ対象を異なる文字列で表す場合などの状況に対処することができない．i) の例として，文字列 “president” は大統領という意味で用いられる場合と社長という意味で用いられる場合とがある．また，ii) の例として，新聞記事の見出しには略語や愛称などが用いられる場合がある．工業製品マニュアルや新聞記事などの重要な文書を扱う文書データベースの高度利用の際に，文書データと文書以外のデータの統合を文字列パターンマッチで行なうことは，極めて不十分であると我々は考える．このような文書には，データベース中に管理されているような対象を意味として表す文字列が多く出現する．製品マニュアル中の部品名はこれに該当する．また，新聞記事に出現する人名，会社名，地域名，日付などが表す意味的対象は通常データベースで管理されているものである．このような文書中の文字列が表す意味データを，本研究ではその文字列の注釈データ (annotation data) と呼ぶ．但し，文字列の注釈データとは次の二種類である．

- データベースに格納されているオブジェクトや値 (OID で特定)

- データベースで管理可能な型の値 (リテラルで特定)

この章では，文書中に出現する文字列と対応するその注釈データを関連付けることで，文書データと文書以外のデータ間のオブジェクトレベルでの統合を可能とするリンク機構を提案する．このリンク機構により上述の i)，ii) の場合における問題点を解決することができる．更に，後で示すように，注釈データに関する型固有の関数を用いることで意味的に高度な問合せを記述することが可能であり，データベースに定義されている索引を利用することで問合せを効率的に処理することが可能となる．重要な文書を管理する際，文書の設計段階からオブジェクトレベルの統合を考慮すべきであるというのが我々の主張である．オブジェクトレベルでの文字列とその注釈データの統合は，完全に自動化することは難しく一部は手動で行なう必要があるので，比較的高いコストを要する．しかしながら，特に重要な文書においては，そのコストに見合う十分な利点があると考える．

6.1.1 概念モデルと実現手法の分離

著者の所属する研究室では，これまでこのような文書データと文書以外のデータ間のオブジェクトレベルでの統合を可能とするリンク機構を実現する手法について研究を進めてきた．その成果は次の通りである．

- SGML 文書に対して特別なエレメントを導入し，そのエレメントの属性にデータベース問合せ言語を指定することで，エレメントの内容とデータベースオブジェクト間にリンク機構を確立する研究 [YIU96].
- [YIU96] におけるデータベース問合せを評価するタイミングを考慮に入れた研究 [YKU95].
- HyTime[NKN91, ISO92, JIS94] の独立リンクを用いた文字列と注釈データ間の双方向リンク機構の確立に関する研究 [加藤 95].

これらは，実現手法に基づく研究であり，他にもいくつかの実現手法が考えられる [絹谷 98]．本研究では，この研究課題を抽象化し実現手法とは独立の概

念モデルとして議論を展開する．実現手法に関しては，応用からの要請により上記の手法を含めた様々な手法が考えられる．

6.2 概念モデル paratext

この節では，オブジェクトレベルでのリンク付き文書モデルである，paratextモデルの必要性について，具体的な例を挙げて記述する．まず，データベース中の文書以外のデータを用いた構造化文書ビューについて記述する．例えば，図5.1に示した新聞記事に関するデータベーススキーマarticleから，図6.1に示す論理構造を有する構造化文書ビューの構築について考える．このビュー定義を図6.2に示す．このビュー定義で用いられているdate2text及びreporter_nameはそれぞれDATE型の値をTEXT型の値に，REPORTER型の値をTEXT型の値に変換する利用者定義関数である．また，4.2.2で導入した関数tagged()の第2引数の型をTEXT型に変更し，以下のように拡張する．

- XML tagged(ELEMENT *arg*1, TEXT *arg*2)
  ELEMENT型の*arg*1をTEXT型の*arg*2の根エレメントに割り当てる．*arg*2がEMPTYの場合，EMPTYを要素とするエレメント*arg*1を根とするST_TEXT型の値が返る．*arg*2がnilの場合，この関数はnilを返す．

このビュー定義によって，news_articlesというXML型の属性を一つ持つビューnewsが構築される．

この構造化文書ビューに対する次のような問合せQ5について考える．

Q5: 1996年9月26日から二週間以内の記事で，本文(<body>)に文字列'WVE'を含む記事(<article>)の日付(<date>)と見出し(<headline>)を検索せよ．

この問合せを単なるXML文書として処理する場合，次のような問題が生じる．

- <date>に関する選択条件を評価するために，14個の文字列，例えば，“September 26, 1996”, “September 27, 1996”, ... , “October 9, 1996”とのパターンマッチ処理を実行する必要がある．

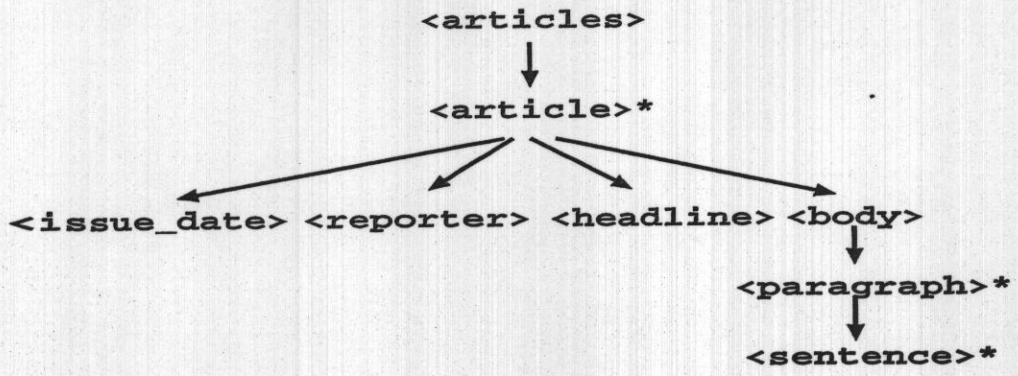

図 6.1 新聞記事の論理構造

```
CREATE VIEW news (news_articles) AS
SELECT tagged(<articles>, cat*(a))
FROM
        SELECT tagged(<article>,
                  tagged(<date>,date2text(issue_date))
                  +tagged(<reporter>,reporter_name(reporter))
                  +headline+body)  AS a
        FROM   articles
```

図 6.2 新聞記事のビュー定義

- <date>に関する部分データ (すなわち，年，月，日) を認識することができない．したがって，日付の部分を組み合わせた問合せの処理は非効率である．

これは，データベース内部表現ではDATE 型で管理されていた日付データをビューを作成する際，文字列データに変換してしまったために上記のような問題点が生じている．データベース内部表現におけるDATE 型の値とビューで作成した XML 文書の<date>間にリンク機構を確立することにより；<date>に関する選択条件を文字列処理としてではなくDATE 型の値に関する選択条件と

することで，DATE型固有の関数や索引が利用でき，Q5は効率的に処理することが可能となる．これにより，例えばより複雑な次のような問合せも効率的に実行することができる．

Q6: 文字列'WVE'と'FastCircuit'が両方本文(<body>)に現れる記事(<article>)から2週間以内の記事のうち，文字列'WVE'を本文に含むものの日付(<date>)と見出し(<headline>)を検索せよ．

同様のことが新聞記事の記者(<reporter>)についても適用できる．

上記のリンク機構を一般化する．一般に文書中には，従来のデータベースで管理されてきた文字列以外の値を意味データとして表す文字列が存在する．この文字列に対して，その文字列が表す意味データを注釈データと呼ぶ．このような文字列とその注釈データを関連付けることで，そのデータベースに対して表現能力の高い問合せが可能となる．例えば，新聞記事をあるデータベースで管理する際，記事中に出現する企業を表す文字列と，この文字列の注釈データであるそのデータベース中で管理されている企業データとの間にオブジェクトレベルでの関連付けを施すことで，以下のような問合せを効率的に処理することが可能となる．

Q7: 1996年9月26日の記事で見出し(<headline>)に従業員1000人以上の企業が出現する記事(<article>)を検索せよ．

もし，この関連付けがないと，該当する企業名による文字列パターンマッチによってこの問合せを処理しなければならない．しかしながら，新聞記事の見出しには略語や愛称などが用いられる場合があり，これらを考慮した複数の文字列によるパターンマッチが必要となり，この種の処理は非効率である．更に，この関連付けは実際にデータベース格納されているデータだけでなく，データベースで管理可能なデータもその対象とすることができる．例えば，利用者が韓国企業の1998年以降の動向に興味を持っている場合，新聞記事に対する次のような問合せが考えられる．

Q8: 同じ段落(<paragraph>)に韓国企業と1998年以降の日付の両方が同時に出現する記事の見出し(<headline>)を検索せよ．

この問合せに関しては，データベースに格納されている企業データと該当する文字列間の関連付けだけでなく，データベースに格納されてはいないが管理可能な日付データと日付を表す文字列間の関連付けを施すことで，効率的に処理することができる．

このような関連づけを持つ構造化文書を統一的に扱う概念モデルである *paratext* モデルを提案する．この paratext モデルにおいて，文書はその表記文字列が保持されている表記層と，表記層における部分文字列が表す意味が保持されている参照層の二つの層から構成されているので，表記文字列だけでなくその文字列の意味に基づく文書操作が可能となる．次の節では，この *paratext* モデルをデータベースで管理する手法について記述する．

6.3 抽象データ型アプローチによる paratext の実現

前述した問合せを実現するために，“ある部分文字列がその文字列の注釈データと対応づけられている構造化文書” という概念を表す抽象データ型，*PARAXML* をデータベースに導入する．この抽象データ型 PARAXML は単純な構造を持ち，実用的な記述能力を持つこの型固有の関数や述語をその操作体系として備えている．まず，PARAXML 型の上位型である PARATEXT 型を導入してから，PARAXML 型を導入する．これらの型とこれまで導入した文書に関する型との IS_A 関係を図 6.3に示す．

本論文で導入される抽象データ型は基本的にデータベースモデルとは直交するものではあるが，**3.4.1**で述べたデータベースモデルを仮定する．また，テキストデータと注釈データは同一のデータベースに格納されるものとする．

6.3.1 PARATEXT 型

PARATEXT 型は「ある部分文字列にその文字列の注釈データがリンク付けされた文書」を管理するための型である．PARATEXT 型の値は概念的に二つの層，表記層と参照層から構成されているものとする．表記層には TEXT 型

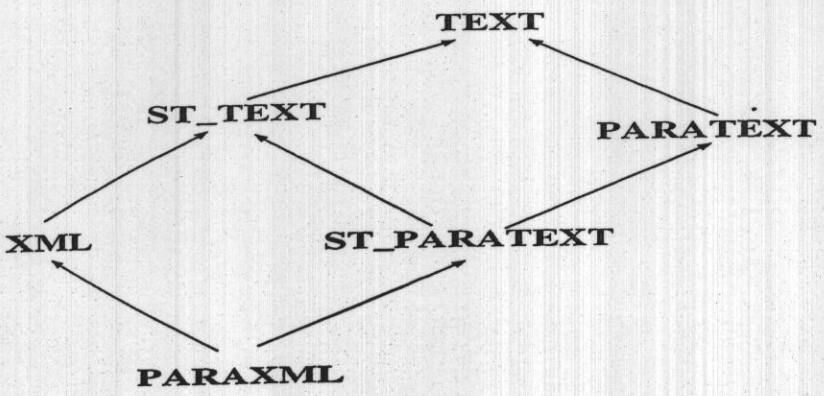

図 6.3 テキスト型に関する IS_A 階層.

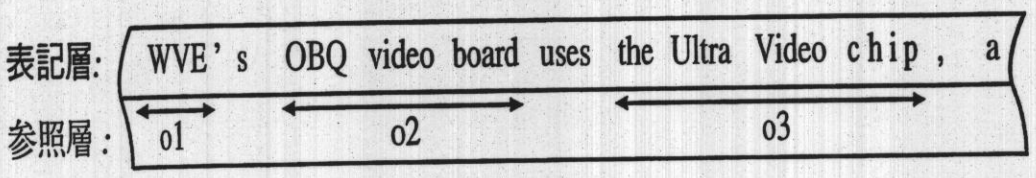

図 6.4 PARATEXT 型の値の例

の値である一次元文字列が保持されており，参照層には，原子型の値のリテラルもしくはそれ以外の型の OID が保持されており表記層の部分文字列とリージョン値により対応付けられている．図 6.4に PARATEXT 型の値の例を示す．

PARATEXT 型の値は以下のように形式的に定義される．

定義 6.3.1: 与えられたデータベースインスタンス Dに対して，D上の PARATEXT 型の値 p は三つ組 (t, R, τ) で表される．但し，

- $t \in I_D(TEXT+)$
- R は t に対して定義される全てのリージョンの集合である．
- τ は全域写像 (total mapping) である．この写像の定義域は Rであり，値域は $I_D(\mathrm{ANY}+) - I_D(TEXT+)$ の有限部分集合の集合である．

□

ここで，次の三つのことについて注意されたい．一つめは，リージョン集合 R に対して重なりに関する制限を課してないと言う点である．つまり，たとえば “the capital of Japan” という文字列とその部分文字列 “Japan” の双方に対して関連する注釈データをリンク付けすることが可能である．二つめは，参照層に注釈データが保持されていない部分文字列に関しては，該当するリージョンのτによる写像は空集合となることである．三つめは，PARATEXT 型の値を単純化するために，TEXT 型のインスタンスの OID が参照層には出現しないように制限している点である．

定義 6.3.2: あるデータベースインスタンス Dが自分自身で閉じている (self-contained) ということは，D中の全ての PARATEXT 型の値 p に対して，p が D上で定義されている時であり，かつその時に限る．□

本論文では，以下自分自身で閉じているデータベースインスタンスについてのみ考える．

与えられた PARATEXT 型のある値 $p \equiv (t, R, \tau)$ に対して，p の表記層に保持されている値や参照層に保持されている値を取り出す，次のような式が定義される．

- $\hat{}(p) \equiv t$
- $\_(p) \equiv \cup\{\tau(r) | r \in R\}$

6.3.2 PARAXML 型

PARAXML 型は PARATEXT 型の下位型であり，表記層に XML 型のインスタンスである XML 文書実現値が保持されている PARATEXT 型の値を管理するための型である．したがって，PARATEXT 型の値同様，PARAXML 型の値は三つ組 (s, R, τ) で定義される．但し，s は XML 型のインスタンスである．

PARAXML 型の操作体系は，ST-式や XML 型固有の関数や述語，さらに ^() や_() など上位型で定義されたものを再定義したものから成る．したがっ

```
CREATE TABLE articles(
  issue_date DATE,
  reporter   REPORTER,
  headline   PARAXML,
  body       PARAXML,
  contents   VECTOR);
```

図 6.5 新聞記事のデータベーススキーマ例

て，4.2.2同様，これらの関数を再定義する際便宜上定義される型として，表記層が ST_TEXT 型のインスタンスが保持されている PARATEXT 型の値の型である ST_PARATEXT 型がある．

表記層に保持されている構造化文書と参照層に出現する値との関連は，PARAXML 型の属性の制約とみなすことができる．この制約を我々は *document and data type definition*(略して D^2TD) と呼び，四つ組 $(d, E, edt, coverage)$ で表わす．但し，

- d は DTD.
- E は d で定義されている文書要素の部分集合.
- edt は定義域を E，値域を $(ANY+) - (TEXT+)$ とする全域関数.
- $coverage$ は定義域を E，値域を {full,partial} とする全域関数.

この D^2TD において全域関数 edt は，表記層に保持されている構造化文書の文書要素に関して，その文書要素に該当するリージョンの参照層に出現することができる値のデータ型を規定している．また，全域関数 $coverage$ は，参照層の値のリージョンが文書要素のリージョンと一致している (total) か，参照層の値のリージョンが文書要素のリージョンの部分でも良い (partial) かを示すものである．この D^2TD を参照することで，利用者は各文書要素の参照層にどの型の値が注釈データとして保持されているかを知ることができる．

ここで，6.2で例として用いた新聞記事のデータベース内部表現である表 article において，属性headline やbody は PARAXML 型として図 6.5のよ

うに再定義される.

6.4 問合せ言語

この節では，PARAXML 型の属性を持つ構造化文書ビューの定義と，作成されたビューに対する問合せについて記述する.

6.4.1 ビュー定義

データベース内部表現から PARAXML 型の属性を持つビューをどのように定義するかについて記述する．既に述べたようにデータベース内部表現において，構造化文書は部分分解されている．そのうちある文書断片は PARAXML 型として管理されているが，他の断片は PARAXML 型以外の型として管理されている．アプリケーションから統一的に構造化文書を扱うためには，次の二つが要求される.

i) データベース内部表現において部分分解されている文書断片を一つの型 (PARAXML 型) の値に統合する.

ii) 内部表現において PARAXML 型以外の型として管理されている値を PARAXML 型の値に変換する.

上記，i) に関しては第4章で導入した関数を用いることで実現できる．ii) に関しては，表記層に保持される値と参照層に保持される値から，PARATEXT 型の値を構築するコンストラクタ para() を導入する.

- PARATEXT para(TEXT *arg*1, ANY *arg*2)
  TEXT 型の *arg*1 を表記層の値とし，ANY 型[1]の *arg*2 を文字列 *arg*1 全体に対応する参照層に保持される値とする PARATEXT 型の値を返す.

PARAXML 型の属性を有するビュー定義の例として，(ビューの章のデータベーススキーマ) に示したデータベース内部表現から同じ日付の記事はまとめ

[1]但し，先に述べたように簡単のため，TEXT 型は適用されないものとする

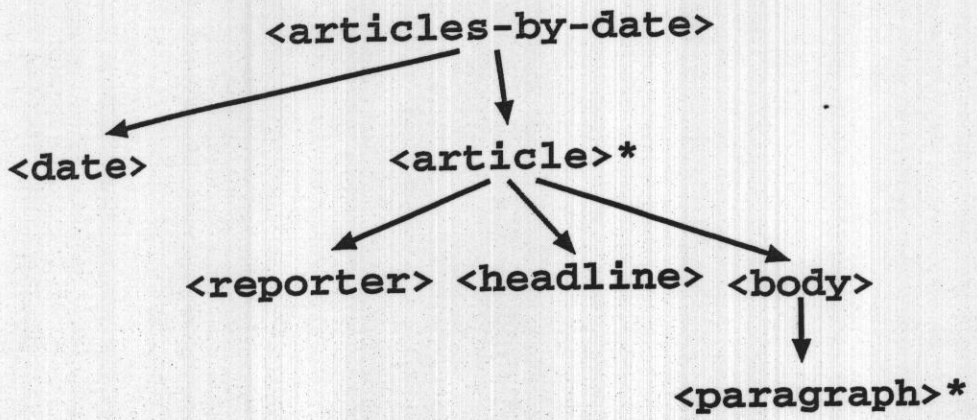

図 6.6 ビューの論理構造例

```
CREATE VIEW newspaper(news) AS
SELECT tagged(<articles-by-date>,
        tagged(<date>,para(date2text(issue_date),issue_date))
            +cat*(a))
FROM
 SELECT tagged(<article>,
         tagged(<reporter>,
                para(reporter_mame(reporter), reporter))
         +headline+body)
 FROM articles
GROUP BY issue_date
```

図 6.7 ビュー定義例

て一つの文書とする，図 6.6に示す論理構造を持つ文書ビューを構築するためのビュー定義文を図 6.7に示す．この文によってnews という PARAXML 型の属性を一つ持つビューnewspaper が構築される．

6.4.2 ビューに対する問合せ

これまで導入したPARAXML型の操作体系を用いることで，文書中の文字列の表す意味に基づく問合せをOQL[Cat97]に我々の拡張を施した表記で記述することができる．拡張された点は，OQLのSELECT-FROM-WHERE形式の問合せにおいて，FROM句に変数の型宣言が記述できる点にある．但し，問合せの安全性の観点から，変数は必ず値域制限(range restriction)されていなければならない．このチェックは文法的にチェック可能である[AHV95]．例えば，**Q8**において変数dは，FROM句においてDATE型であることが宣言されており，かつFROM句で値域制限されている変数pを伴う式d IN _(p)により値域制限される．したがって，問合せ**Q8**は安全な問合せと判断できる．

6.2で述べた自然言語による問合せ例を上で定義したビューに対する問合せとしてこの拡張されたOQLでどのように表現されるかを図6.8と図6.9に示す．但し，仮定するデータベースには以下に示す二つの関係が格納されているものとする．

```
newspaper(news)
company(name, country, emps)
```

このビューのPARAXML型の属性news の制約であるD^2TDは四つ組

$$(d, E, edt, coverage)$$

で表される．但し，dは属性news の表記層に保持されているXML文書のDTDであり，図6.6に示されている文書の論理構造を規定している．また，文書要素集合Eおよび全域関数edtと$coverage$の値域をまとめて表すと以下のような表になる．

E	edt	$coverage$
<date>	DATE	full
<reporter>	REPORTER	full
<headline>	ANY	partial
<body>	ANY	partial

ここで，問合せの記述を簡単にするために次の省略を許すものとする．ある文書要素全体に注釈データが対応付けられている場合で，_() が一つだけの要素からなる集合 (singleton) を返すときは，その要素を返すものとする．例えば，**Q6**において，WHERE 句中の_(news/<date>) は，DATE 型の値を返すものとする．

6.5 問合せ処理

ビューに対する問合せはデータベース内部表現に対する問合せに変換される．この変換は，既存の OQL 解析器と XML 解析器の組み合わせとわずかな拡張で実現できる．この節では，変換された問合せをどのように処理するかについて記述する．ビューに対する問合せに関して，その問い合わせ条件などで参照する文書の論理構造の単位が内部表現上の単位と適合している場合その問合せ処理は単純であるのに対して，この単位が内部表現上の単位をまたがる場合などは，ナイーブな処理を行なうとそのコストが非常に高価になってしまう．そこで，このような場合の最適化手法についていくつか提案する．以下，単純な問合せ処理について **6.5.1**で記述し，その後で，いくつかの最適化手法について **6.5.2**で記述する．

6.5.1 単純な問合せ処理

この節では，ビューに対する問合せで，その問い合わせ条件などで参照する文書の論理構造の単位が内部表現上の単位と適合している場合の問合せ処理について記述する．例えば，図 6.8及び図 6.9に示した概念モデルに対する問合せ例は，図 6.7に示したビュー定義を介して，それぞれデータベース内部表現に対する問合せとして図 6.10及び図 6.11に示す問合せにそれぞれ変換される．

次に，変換されたデータベース内部表現に対する問合せがどのように処理されるかについて記述する．すなわち，PARAXML 型固有の関数や述語の処理について記述する．PARAXML 型以外の型の値に関しては通常のデータベースにおける問合せ処理と同様である．既に述べたように我々は文書内容の更新は仮定していない．また，文書の論理構造が変化しない単位で分割したの

Q5:1996 年 9 月 26 日から二週間以内の記事で，本文 (<body>) に文字列'WVE' を含む記事 (<article>) の日付 (<date>) と見出し (<headline>) を検索せよ．

```
SELECT   r.news/<date>,a/<headline>
FROM     newspaper AS r, r.news/<article> AS a
WHERE    _(r.news/<date>) >= DATE '1996-09-26'
AND      _(r.news/<date>)<= DATE '1996-09-26'
                                  + INTERVAL '14' DAY
AND      a/<body> CONTAIN 'WVE'
```

Q6: 文字列 'WVE' と 'FastCircuit' が両方本文 (<body>) に現れる記事 (<article>) から 2 週間以内の記事のうち，文字列 'WVE' を本文に含むものの日付 (<date>) と見出し (<headline>) を検索せよ．

```
SELECT   r2.news/<date>, a2/<headline>
FROM     newspaper AS r1, newspaper AS r2,
         r1.news/<article> AS a1,
         r2.news/<article> AS a2
WHERE    ^(a1/<body>) CONTAIN 'WVE'
AND      ^(a1/<body>) CONTAIN 'FastCircuit'
AND      _(r2.news/<date>) >= _(r1.news/<date>)
AND      _(r2.news/<date>) <= _(r1.news/<date>)
                                  + INTERVAL '14' DAY
AND      ^(a2/<body>) CONTAIN 'WVE'
```

図 6.8 概念モデルへの問合せ例 1

がデータベース内部表現である．したがって，データベース内部表現における PARAXML 型の値に対して位置に基づく転置ファイル形式の索引を構築することができる．通常の転置ファイルとの違いは，参照層に保持されている値に対してもその出現位置を管理する点である．したがって，データベース内部

Q7:1996年9月26日の記事で見出し(<headline>)に従業員1000人以上の企業が出現する記事(<article>)を検索せよ.

```
SELECT  a
FROM    newspaper AS r, company AS c, r.news/<article> AS a
WHERE   c IN _(n.articles/<article>/<headline>)
AND     c.emps >= 1000
AND     _(n.articles/<article>/<date>) = DATE '1996-09-26'
```

Q8:同じ段落(<paragraph>)に韓国企業と1998年以降の日付の両方が同時に出現する記事の見出し(<headline>)を検索せよ.

```
SELECT  a/<headline>
FROM    newspaper AS r, company AS c, r.news/<article> AS a,
        a/<body>/<paragraph> AS p
WHERE   c IN _(p)
AND     d IN _(p)
AND     c.country = 'Korea'
AND     d >= DATE '1998-01-01'
```

図 6.9 概念モデルへの問合せ例 2

表現においてPARAXML型のある属性に保持されている全ての値に対して，単語，文書要素名，参照層に出現する値の三種類の転置ファイルを作成する．これらの転置ファイルにおいて，出現位置は，所属する表中のレコードIDとリージョンの対で管理される．これらの索引を用いることで，データベース内部表現におけるPARAXML型の値の操作のうち，以下の形式の式は効率良く評価することができる．

- $\hat{}(s)$ CONTAIN w
- v IN $\_(s)$

例えば，図6.9中の**Q6**の条件式$\hat{}(a2.body)$ CONTAIN $'WVE'$は，属性bodyに作成されている単語に関する転置ファイル形式の索引で，文字列'WVE'の

```
Q5:
 SELECT   tagged(<date>,
                    para(date2text(issue_date), issue_date)),
          headline
 FROM     article
 WHERE    issue_date >= DATE '1996-09-26'
 AND      issue_date <= DATE '1996-09-26' + INTERVAL '14' DAY
 AND      ^(body) CONTAIN 'WVE'
Q6:
 SELECT   tagged(<date>,para(date2text(a2.issue_date), a2.issue_date)),
          a2.headline
 FROM     article AS a1, article AS a2
 WHERE    a1.body CONTAIN 'WVE'
 AND      a1.body CONTAIN 'FastCircuit'
 AND      a2.issue_date >= a1.issue_date
 AND      a2.issue_date <= a1.issue_date + INTERVAL '14' DAY
 AND      ^(a2.body) CONTAIN 'WVE'
```

図 6.10 データベース内部表現への問合せ例 1

出現を確認することで効率的に評価される．また，図 6.9中の Q8の条件式 c IN _($a.body$. < paragraph >) AND $c.country$ =' $Korea$'は，属性 body に作成されている参照層の値に関する転置ファイル形式の索引と論理構造に関する転置ファイル形式の索引を用いることで，company.country='Korea' である company の OID の出現位置と，文書要素名が<paragraph>である論理構造の出現位置に関して，同じレコード ID 同士のリージョン間の包含関係を調べることで，効率的に評価される．

```
Q7:
 SELECT  tagged(<article>,
             tagged(<reporter>,para(reporter_mame(reporter),reporter))
                 +headline+body)
 FROM    article AS a, company AS c
 WHERE   c IN _(a.headline)
 AND     c.emps >= 1000
 AND     a.issue_date = DATE '1996-09-26'
Q8:
 SELECT  a.headline
 FROM    article AS a, company AS c,
         a.body/<paragraph> AS p, DATE AS d
 WHERE   c IN _(p)
 AND     d IN _(p)
 AND     c.country = 'Korea'
 AND     d >= DATE '1998-01-01'
```

図 6.11 データベース内部表現への問合せ例 2

6.5.2 問合せ最適化

この節では，ナイーブに問合せを処理するとそのコストが高価な場合の最適化手法について考察する．典型的には，ビューの表記層の文字列が内部表現では文字列以外の値として管理されている場合における文字列検索が挙げられる．また，ビューに対する問合せで文字列の順序関係や近接関係が内部表現において複数の文書フラグメントにまたがる場合の最適化については **5.3.1** で記述した手法が適用できるので，この節では特に記述しない．

以下，構造化文書ビューの表記層における文字列検索の最適化について記述する．表記層の文字列 s がデータベース内部表現では文字列以外の値 v として管理されている場合は，問合せ処理の際にそのような v をすべて s に変換するナイーブな処理方法は効率が悪い．そこで，表記層における s(の部分文字列)

の検索を効率的に処理するためには以下に示す二つの方法がある．

(a) ビュー作成時に vに適用される，文字列を返す関数 $f()$ が固定されているならば，あらかじめ $f(v)$ の結果の文字列を全文索引に登録しておくことで，効率的に処理することが可能となる．

(b) 文字列 s を値 v(の一部) に変換して，参照層に対する値 v(の一部) の出現の特定に関する問合せに変換することで，効率的に処理することが可能となる．

例えば，図 6.7に示した定義で構築されるビューの表記層に対して，文字列 “1998 年” の出現の特定に関する問合せは以下に示す二つの方法のどちらかを採用することで，効率的に処理される．

(a) ビュー作成時にデータベース内部表現におけるデータベース属性 `issue_date` に適用される文字列を返す関数が *date2text*() に固定されているならば，あらかじめ表`article` の各組に関して *date2text*(`issu_date`) の結果の文字列を全文索引に登録しておけば，この全文索引を用いることで，文字列 “1998 年” の出現位置を効率的に特定することができる．

(b) YEAR field が 1998 である DATE 型の値の参照層における出現位置を特定することで，文字列 “1998 年” の出現位置を効率的に特定することができる．

6.6 議論と今後の課題

この章で提案した文書中の文字列と注釈データ間のリンクの構築は，文書を新たに作成する時に著者によってなされるものと仮定している．このリンク構築を自動化することは困難ではあるが，既存の技術を用いることで著者のリンク構築をサポートするエディタを開発することは可能である．文書中の単語の型は，Glean project[CS97] のために開発された技術を適用することで導出することが可能である [LSCS97]．例として単語 “appointed” に着目する．この Glean 技術は supertagging 技術 [JS94] を用いて，パターン

- Name/Person appinted Name/Person Position
- Name/Person was appointed to Position

を抽出することができる．したがって，著者の “appointed” という単語入力に対してその直前の単語の型が Person であることを導出し，データベース中の PERSON 型の値をその単語の注釈データの候補として利用者に提示するエディタを開発することが可能である．更に，これまで構築されたリンクの辞書を用いることでその候補を絞り込むことも可能である．このリンク構築の半自動化に関する研究は今後の課題である．

また，この章で提案した文書を管理するための新しいデータ型である PARATEXT 型と PARAXML 型は複合型としても実現可能であるが，我々は抽象データ型として実現することにした．抽象データ型としてデータベースに導入することで，型のインタフェースと実装を分割することができる．これらの型の実現方法はその参照層のデータ構造により複数の実装が考えられる [絹谷 98]．したがって，アプリケーションに応じて，これらの型固有の操作関数を効率的に処理する実装法を選択することができる．これに対して，複合型として実現した場合，利用者はこれらの型の値を操作する際，構成子を入れ子にした複雑な式を記述しなければならい．そこで，本研究では利用者が簡単に値を操作できるようなインタフェースを備えた抽象データ型によるアプローチを採用した．

構造化文書とデータベースオブジェクト間の対応づけに関する研究として，[ACM93, ACM95, BRS96] がある．これらの研究は，既存の文書に対して文書の論理構造とデータベーススキーマの対応づけを，*structuring schema*[ACM93, ACM95] や *likeness mechanism*[BRS96] によって実現している，これに対して，我々の文書の概念モデルである paratext は，新規文書作成時に文書中の任意の文字列とデータベースオブジェクト間のインスタンスレベルでの対応づけを行なうものである．

第 7 章

結論

7.1 まとめ

本研究では，データベースビューとしての構造化文書の定義とビューに対する問合せに関して論じた．このような構造化文書ビューに対する問合せ記述の支援や問合せ最適化のために，DTD を持たない構造化文書ビュー定義から DTD を導出する手法を提案した．また，データベース問合せの伝統的な最適化手法である “selection push-down” について，文書固有の選択条件がビューで定義される論理構造の種類に応じてどのように “push-down” されるかを整理した．これにより，文書固有の条件を伴う構造化文書文書ビューに対する問合せの処理を最適化することが可能となった．

更に，データベース中の文書データと他のデータの統合利用のため，新たな構造化文書モデルとして抽象データ型 *PARAXML* 型を設計した．これにより，文書中の表記文字列による文字列パターンマッチだけでなく，表記文字列の表す意味に基づく問合せが可能になることを示した．また，そのような問合せの処理について最適化手法も提案した．

7.2 議論と今後の課題

本研究ではデータベースでXML文書を管理する際，抽象データ型アプローチを採用した．抽象データ型をデータベースシステムに逐次的に導入することで，データベースの管理対象を広げることに関する研究成果は既に存在する[Sto96]．このアプローチは抽象データ型アプローチ(ADTアプローチ)と呼ばれ，その最大の利点は，モジュール性と拡張可能性にある[Sto96, CD96]．すなわち，各抽象データ型の追加や削除を，データベースの他の部分に影響を与えることなく逐次的に達成できる．これに対してSeshadriらは，データベースに新たな定義域を抽象データ型としてプラグインする際，型固有の最適化規則や同じ型でも異なる実装を持つ問合せ処理の装着といった，最適化を意識した定義域の逐次追加として，拡張抽象データ型アプローチ（E-ADTアプローチ）を提案している[SLR96, SLR97, Ses98b, Ses98a]．このE-ADTアプローチの特徴は，関係(relation)を他のADTと同様に関係ADTとみなし，問合せの最適化に関して各抽象データ型において独立性が維持される点を強調している．しかしながら，E-ADTアプローチにおいてはデータベースビューを取り扱っていない．本研究を通じて，E-ADTアプローチの利点であるといわれる問合せ最適化に関する各抽象データ型間の独立性が，内部に構造を有するデータ型に関するデータベースビューに対しては成り立たないことが判明した．すなわち，**5.2**で述べたビューDTDの導出手法や，**5.3**で述べた文書ビューに対する問合せの最適化手法，**6.5**で述べたparatextビューの問合せ手法では，関係ADTとXML ADT間の相互作用が必要となる．この相互作用を分析し，内部に構造を有する型に対する一般的な最適化規則に関する研究は，今後の課題である．

本研究を通じて扱ったビューは，仮想ビューであった．しかしながら，ネットワークを介した文書ビューの配信を考慮に入れると，具体ビュー(materialized view)を対象としたに対する問合せ処理を考える必要があるが，それは今後の課題とする．しかしながら，ネットワークを介した文書ビューの配信の場合でも，利用者は配信された文書を保存しないかもしれない．このような状況においては，第5章で記述した仮想ビューとしての構造化文書ビューに対する問合せ処理の最適化手法が有効であるものと考える．また，ビュー更新に関して

は，本研究では扱わなかったし，本研究の想定している応用例では発生しないものであると考える．

本研究で扱ったXML文書は，要素内容モデルだけを研究の対象としたが，混在内容モデルに関しては疑似エレメントを導入することで，また，XML属性に関しては属性が定義されているエレメントの子エレメントとして考えることで，本研究で提案した手法は基本的に適用することができる．但し，DTDの導出やtagged()関数の定義などにおいて，疑似エレメントと通常のエレメント，XML属性に対応するエレメントの三つの取り扱いを区別する必要がある．この区別をどのようにするかについては今後の課題である．

謝辞

本研究を行うにあたり適切な御指導ならびに並々ならぬ御援助を賜わりました植村俊亮教授に心から深く感謝の意を表し，御礼を申し上げます．植村先生には，ご多忙中にも関わらず本研究の主指導教官になって頂き，本論文の草稿に対しまして，多数のコメントを頂きました．

ご多忙中にも関わらず本研究の副指導教官になって頂きました伊藤実教授に深く御礼申し上げます．伊藤先生には，本研究に対する貴重なコメント以外に，データベースの理論的側面に関して講義や輪講を通じて御教授頂きました．特に，[AHV95] の輪講を通じてデータベースの理論的基礎を御教授頂きました．ここに，心から感謝の意を表します．

ご多忙中にも関わらず本研究の副指導教官になって頂きました松本裕治教授に深く御礼申し上げます．松本先生には，本研究に対する貴重な御意見以外に，研究に対する取り組み方をお教え頂きました．ここに，衷心より感謝致します．

ご多忙中にも関わらず本研究の副指導教官になって頂きました吉川正俊助教授に深く御礼申し上げます．吉川先生には，研究に関する御助言，御提案，御援助を含む，終始丁寧な御指導を頂きました．特に，常日頃の御議論は本研究を進めるにあたり不可欠のものでありました．また，吉川先生の研究に取り組む姿勢から多くの貴重なことを学ぶことができました．ここに心より感謝の意を表します．

付録で述べたプロトタイプを共同開発した，マルチメディア統合システム講座の絹谷弘子氏，志村壮是氏及び山本陽平氏に感謝の意を表します．

本研究に対する有益なコメント並びに，[AHV95] の輪講を通じてデータベースの理論的基礎に関して御教授頂きました関教授に心より感謝の意を表します．

また，終始有益な御助言いただきました石川佳治助手に心からの感謝の意を表します．石川先生の研究に対する姿勢から多くの事を学びました．

計算機科学の理論的基礎に関して，示唆に富む御議論を拝した知識工学講座の石原靖哲助手に衷心より感謝の意を表します．

[AHV95] の輪講を通じてデータベースの理論的基礎に関する多くのことを学び，それが本研究を遂行する上で非常に役に立ちました．この輪講を通じてデータベースに関する基礎を御教授頂いた，中西隆一助手，高田喜朗助手をはじめとする知識工学講座と計算機言語学講座の皆様に感謝いたします．

本論文の草稿に対して，貴重なコメントを頂いた絹谷弘子氏に感謝いたします．絹谷氏には，本研究の遂行に当たり研究会などを通じて多くの議論を頂きました．ここに感謝の意を表します．

XML の仕様やツール群などを御教授頂いた朝倉浩氏，志村壮是氏，山本陽平氏に衷心より感謝の意を表します．

研究会などを通じて，本研究に対する有益な御助言および御協力頂いたマルチメディア統合システム講座の皆様に感謝いたします．

参考文献

[ABB95] Serge Abiteboul and Catriel Beeri. The power of languages for the manipulation of complex values. *The International Journal on Very Large Data Bases*, Vol. 4, No. 4, pp. 727–794, October 1995.

[ACM93] Serge Abiteboul, Sophie Cluet, and Tova Milo. Querying and updating the file. In *Proc. of the 19th International Conference on Very Large Data Bases (VLDB)*, pp. 73–84, August 1993.

[ACM95] Serge Abiteboul, Sophie Cluet, and Tova Milo. A database interface for files update. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 386–397, May 1995.

[ACM97] Serge Abiteboul, Sophie Cluet, and Tova Milo. Correspondence and translation for heterogeneous data. In *Proc. of 6th International Conference on Database Theory (ICDT '97), Lecture Notes in Computer Science, Vol. 1186*, January 1997.

[AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases.* Addison-Wesley, 1995.

[ASU86] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques and Tools.* Addison-Wesley, 1986.

[BAK95] Klemens Boehm, Karl Aberer, and Wolfgang Klas. Building a

cnfigurable database application for structured documents. *Multimedia - Tools and Applications*, 1995.

[BANY97] Klemens Boehm, Karl Aberer, Erich J. Neuhold, and Xiaoya Yang. Structured document storage and refined declarative and navigational access mechanisms in hyperstorm. *The International Journal on Very Large Data Bases*, Vol. 6, No. 4, pp. 296–311, November 1997.

[BCD+95] G. Elizabeth Blake, Mariano P. Consens, Ian J. Davis, Pekka Kilpeläinen, Eila Kuikka, Per-Å. Larson, Tim Snider, and Frank W. Tompa. Text / Relational Database Management Systems: Overview and Proposed SQL Extensions. Technical Report CS-95-25, UW Centre for the New OED and Text Research, Department of Computer Science, University of Waterloo, June 1995.

[BCD+98] Lauri J. Brown, Mariano P. Consens, Ian J. Davis, Chris R. Palmer, and Frank W. Tompa. A Structured Text ADT for Object-Relational Databases. *Theory and Practice of Object Systems*, Vol. 4, No. 4, pp. 227–244, 1998. Objects, Databases, and the WWW, a special issue.

[BRS96] Stephen Blott, Lukas Relly, and Hans-Jörg Schek. An open abstract-object storage system. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 330–340, June 1996.

[Bur92] Forbes J. Burkowski. An algebra for hierarchically organized text-dominated databases. *Information Processing & Management*, Vol. 28, No. 3, pp. 333–348, 1992.

[BYN96] Ricardo Baeza-Yates and Gonzalo Navarro. Integrating contents and structure in text retrieval. *ACM SIGMOD Record*, Vol. 25,

No. 1, pp. 67–79, March 1996.

[CACS94] Vassilis Christophides, Serge Abiteboul, Sophie Cluet, and Michel Scholl. From structured documents to novèl query facilities. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 313–324, May 1994.

[Cat97] Rick G. G. Cattell, editor. *The Object Database Standard: ODMG2.0.* Morgan Kaufmann, 1997.

[CCB95a] Charlie L. A. Clarke, Gordon V. Cormack, and Forbes J. Burkowski. An Algebra for Structured Text Search and A Framework for its Implementation. *The Computer Journal*, Vol. 38, No. 1, pp. 43–56, 1995.

[CCB95b] Charlie L. A. Clarke, Gordon V. Cormack, and Forbes J. Burkowski. Schema-independent retrieval from heterogeneous structured text. In *Proc. of the 4th Annual Symposium on Document Analysis and Information Retrieval*, pp. 279–290, Las Vegas, April 1995.

[CCD+98] Stefano Ceri, Sara Comai, Ernesto Damiani, Piero Fraternali, Stefano Paraboschi, and Letizia Tanca. XML-GL: A Graphical Language for Querying and Reshaping XML Documents. In *Position papers for W3C Query Language Workshop.* 1998.

[CD96] Michael J. Carey and David J. DeWitt. Of objects and databases: A decade of turmoil. In *Proc. of the 22nd International Conference on Very Large Data Bases (VLDB)*, pp. 3–14, Bombay, September 1996.

[CDSS98] Sophie Cluet, Claude Delobel, Jérôme Siméon, and Katarzyna Smaga. Your Mediators Need Data Conversion! In *Proc. ACM*

SIGMOD International Conference on Management of Data, 1998.

[CGMH+94] Sudarshan Chawathe, Hector Garcia-Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey Ullman, and Jennifer Widom. The tsimmis project: Integration of heterogeneous information sources. In *IPSJ DBS-ken 100-2*, pp. 7–18, October 1994.

[Cha98] Surajit Chaudhuri. An overview of query optimization in relational systems. In *Proc. ACM Symp. on Principles of Database Systems*, pp. 34–43, June 1998.

[CM94] Mariano P. Consens and Tova Milo. Optimizing queries on files. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 301–312, May 1994.

[CM95] Mariano P. Consens and Tova Milo. Algebras for querying text regions. In *Proc. ACM Symp. on Principles of Database Systems*, pp. 11–22, May 1995.

[Cor97a] Cornell University . PREDATOR: DESIGN AND IMPLEMENTATION. http://simon.cs.cornell.edu/Info/Projects/ PREDATOR/designdoc.html, Aug 1997.

[Cor97b] Cornell University . PREDATOR home page. http://www.cs.cornell.edu/Info/Projects/ PREDATOR/, Aug 1997.

[CS97] Raman Chandrasekar and B. Srinivas. Using syntactic information in document filtering: Acomparative study of part-of-speech tagging and supertagging. In *Proceedings of RIAO '97*, June 1997.

[DD93] C. J. Date and Hugh Darwen. *A Guide to The SQL Standard, 3rd ed.* Addison-Wesley, Reading, MA, 1993.

[DD94] Steven J. DeRose and David G. Durand. *MAKING HYPERMEDIA WORK -A User's Guide to HyTime-.* KLUWER ACADEMIC PUBLISHERS, 1994.

[DFF+98] Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu. XML-QL : A Query Language for XML, Aug 1998. http://www.w3.org/TR/NOTE-xml-ql/.

[DM97] Stefan Dessloch and Nelson Mattos. Integrating SQL databases with content-specific search engines. In *Proc. of the 24th International Conference on Very Large Data Bases (VLDB)*, pp. 528—537, Athens, August 1997.

[ElM96] Special issue on integrating text retrieval and databases. In Eliot Moss, editor, *Bulletin of the Technical Committee on Data Engineering*, Vol. 19, pp. 13–27. IEEE COMPUTER SOCIETY, March 1996.

[FBY92] William B. Frakes and Ricardo Baeza-Yates, editors. *Information Retrieval – Data Structures & Algorithms* –. Prentice-Hall, 1992.

[FFK+98] Mary Fernández, Daniela Florescu, Jaewoo Kang, Alon Levy, and Dan Suciu. Catching the Boat with Strudel: Experiences with a Web-Site Management System. In *Proc. ACM SIGMOD International Conference on Management of Data*, 1998.

[GBYS92] Gaston H. Gonnet, Ricardo A. Baeza-Yates, and Tim Snider. New indices for text: PAT trees and PAT arrays. In W. B. Frakes and R. Baeza-Yates, editors, *Information Retrieval – Data*

Structures & Algorithms –, chapter 5, pp. 66–82. Prentice-Hall, 1992.

[Gra93] Goetz Graefe. Query evaluation techniques for large databases. *ACM Computing Surveys*, Vol. 25, No. 2, pp. 73–170, June 1993.

[GT87] Gaston H. Gonnet and Frank Wm. Tompa. Mind your grammar: a new approach to modelling text. In *Proc. of the 13th International Conference on Very Large Data Bases (VLDB)*, pp. 339–346, September 1987.

[IBM98] IBM Corporation. XML for Java. http://www.alphaworks.ibm.com/, Feb 1998.

[ISO86] ISO 8879: 1986. *Information Processing – Text and Office System – Standard Generalized Markup Language (SGML)*, Oct. 15 1986.

[ISO92] ISO/IEC 10744: 1992. *Hypermedia/Time-based Structuring Language (HyTime)*, 1992.

[ISO97] ISO/IEC CD 13249-2:199x(E).*Information Technology – Database Languages – SQL Multimedia and Application Packages (SQL/MM)– Part2: Full-Text*, 1997.

[JIS92] JIS X 4151:1992 文書記述言語 SGML (Standard Generalized Markup Language), 日本規格協会, 1992.

[JIS94] JIS X 4155: ハイパメディア及び時間依存情報の構 造化言語 HyTime (Hypermedia/Time-based Structuring Language), 日本規格協会, 1994.

[JIS98a] JIS X 4151:1998 文書記述言語 SGML (Standard Generalized Markup Language)(追補 1), 日本規格協会, 1998.

[JIS98b] TR X 0008:1998 拡張可能なマーク付け言語 XML (eXtensible Markup Language), 日本規格協会, 1998.

[JS94] Aravind K. Joshi and B. Srinivas. Disambiguation of Super Parts of Speech (or Supertags): Almost Parsing. In *Proceedings of the International Conference on Computational Linguistics (COLING '94)*, August 1994.

[KM93] Pekka Kilpeläinen and Heikki Mannila. Retrieval from hierarchical texts by partial patterns. In *Proc. of ACM SIGIR'93*, pp. 214–222, 1993.

[LSCS97] Zoé Lacroix, Arnaud Sahuguet, Raman Chandrasekar, and B. Srinivas. A Novel Approach to Querying the Web: Integraging Retrieval and Browsing. In *Proceedings of ER97 Workshop on Conceptual Modeling for Multimedia Information Seeking,*, 1997.

[NBY97] Gonzalo Navarro and Ricardo Baeza-Yates. Proximal nodes: A model to query document databases by content and structure. *ACM Trans. on Information Systems*, Vol. 15, No. 4, pp. 400–435, October 1997.

[NKN91] Steven R. Newcomb, Neill A. Kipp, and Victoria T. Newcomb. The 'hytime' hypermedia/time-based document structuring language. *Communications of the ACM*, Vol. 34, No. 11, pp. 67–83, November 1991.

[PGGMU95] Yannis Papakonstantinou, Ashish Gupta, Hector Garcia-Molia, and Jeffrey Ullman. A query translation scheme for rapid implementation of wrappers. In *Proc. of the Fourth International Conference on Deductive and Object-Oriented Databases (DOOD'95), Lecture Notes in Computer Science*, pp. 161–186, Singapore, December 1995.

[PV99] Yannis Papakonstantinou and Pavel Velikhov. Enhancing semistructured data mediators with document type definitions.

In *Proc. of IEEE International Conference on Data Engineering*, March 1999. (*to appear*)

[Ram97] Raghu Ramakrishnan. *Database Management Systems.* McGraw-Hill, 1997.

[SDAMZ94] 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.

[SDDTZ97] Ron Sacks-Davis, Tuong Dao, James A. Thom, and Justin Zobel. Indexing documents for queries on structure, content and attributes. In *International Symposium on Digital Media Information Base (DMIB'97)*, Nov. 1997.

[SDKR+95] Ron Sacks-Davis, Alan Kent, Kotagiri Ramamohanarao, James Thom, and Justin Zobel. Atlas: A nested relational database system for text applications. *IEEE Trans. on Data and Knowledge Engineering*, Vol. 7, No. 3, pp. 454–470, June 1995.

[Ses98a] Praveen Seshadri. Enhanced abstract data types in object-relatinal databases. *The International Journal on Very Large Data Bases*, Vol. 7, pp. 130–140, 1998.

[Ses98b] Praveen Seshadri. PREDATOR:A Resource for Ddatabse Research. *ACM SIGMOD Record*, Vol. 27, No. 1, March 1998.

[SLR96] Praveen Seshadri, Miron Livny, and Raghu Ramakrishnan. E-ADTs: Turbo-Charging Complex Data. *IEEE Data Engineering Bulletin*, Vol. 19, No. 4, pp. 11–18, 1996.

[SLR97] Praveen Seshadri, Miron Livny, and Raghu Ramakrishnan. The case for enhanced abstract data types. In *Proc. of the 24th In-*

ternational Conference on Very Large Data Bases (VLDB), pp. 66—75, Athens, August 1997.

[SM96] Eric Skinner and John McFadden. Microdocument database architectures. *<TAG> The SGML Newsletter*, Vol. 9, No. 10, pp. 1–7, October 1996.

[ST94] Airi Salminen and Frank W. Tompa. Pat expressions: an algebra for text search. *Acta Linguistica Hungarica*, Vol. 41, No. 1-4 (1992-93), pp. 277–306., 1994.

[Sto96] Michael Stonebraker. *OBJECT-RELATIONAL DBMSs- THE NEXT GREAT WAVE.* Morgan Kaufmann, 1996.

[VAB96] Marc Volz, Karl Aberer, and Klemens Böhm. Applying a flexible oodbms-irs-coupling to structured document handling. In *Proc. of IEEE 12th International Conference on Data Engineering*, pp. 10–19, Feb.-Mar. 1996.

[Wor98] World Wide Web Consortium. QL'98 - The Query Languages Workshop. http://www.w3.org/TandS/QL/QL98/, December 1998.

[WWWC98] World Wide Web Consortium. Extensible Markup Language (XML) 1.0. http://www.w3.org/TR/1998/REC-xml-19980210, February 1998. W3C Recommendation 10-February-1998.

[YA94] Tak W. Yan and Jurgen Annevelink. Integrating a structured-text retrieval system with an object-oriented database system. In *Proceedings of the Twentieth International Conference on Very Large Databases*, pp. 740–749, Santiago, Chile, 1994. Industrial Case.

[YIU96] Masatoshi Yoshikawa, Osamu Ichikawa, and Shunsuke Uemura. Amalgamating sgml documents and databases. In *Proc. of the*

5th International Conference on Extending Database Technology (EDBT'96), Lecture Notes in Computer Science, No. 1057, Springer-Verlag, pp. 259–274, March 1996.

[YKU95] Masatoshi Yoshikawa, Hiroyuki Kato, and Shunsuke Uemura. A uniform mechanism for incorporating database objects into sgml documents. In *Proc. of the International Symposium on Digital Libraries*, pp. 289–290, August 1995.

[加藤 95] 加藤弘之, 吉川正俊, 植村俊亮. 構造化文書とデータベース間の汎用リンク機構の実現法. 情報処理学会アドバンスト・データベース・シンポジウム, pp. 9–18, 12 月 1995.

[絹谷 98] 絹谷弘子, 吉川正俊. オブジェクトと文書内文字列間の相互参照機構を有するデータベースシステムの ParaDocs のアーキテクチャ. 情報処理学会デジタル・ドキュメント・シンポジウム'98, January 1998.

付録

A　ビュー定義から DTD の導出アルゴリズム

この節では，第5章で述べた OQL 表現のビュー定義から，文書ビューの DTD を導出するアルゴリズムの詳細を記述する．

図 7.1に示した手続き SQL2EDExpList は，入力 Select-From 式 *sql* に対して，入れ子形式の入力のある SELECT 句に着目して，その SELECT 句中に出現するデータベース属性に関する内容モデルの正規表現を *Ex.AElist* に，組 (データベース属性名，内容モデルの正規表現) のリストとして出力する．また，その着目している SELECT 句に出現するエレメント名に関する内容モデルの正規表現を *Ex.EDlist* に，組 (エレメント名，内容モデルの正規表現) のリストとして出力する．手続き SQL2EDExpList はこの *Ex.AElist* と *Ex.EDlist* をそれぞれ *Ans.AElist* と *Ans.ESlist* に追加しながら，入れ子形式の入力 Select-From 式を最も内側までたぐっていく．但し，内容モデルの正規表現には，データベース属性名が出現する場合がある．よって，この手続きの最後で二つのリストを走査しながらデータベース属性名を対応するエレメント名に置換する．

図 7.2に示した手続き ElementDeclareExp は，前述した手続き SQL2EDExpList 中で用いられており，“ST^i-式 *st* [AS *a*]” 形式の一つの選択リスト項目を入力とし，*Ans.AElist* に組 (データベース属性名，内容モデルの正規表現) のリストを，*Ans.ESlist* に組 (エレメント名，内容モデルの正規表現) のリストをそれぞれ出力する．

図 7.3に示した手続き RegularExp は，手続き SQL2EDExpList 中で用いられており，入力 ST^i-式に対して，内容モデルの正規表現を出力する．

```
procedure SQL2EDExpList(sql, Ans) result Ex
    sql is a OQL query formd as:
      SQL := SELECT S1 [AS A1], S2 [AS A2], ..., Sm [AS Am]
             FROM   F1, F2, ..., Fn
      Fj  := SQL
    Ans, Ex are pair (EDlist, AElist)
      EDlist is a list of pair(ele, cont)
      AElist is a list of pair(att, cont)
      ele holds a element name
      att holds a database attribute name
      cont holds a regular expression of content model

 for each i from 1 to m    ElementDeclareExp(S_i [AS A_i],E)
  append E.EDlist into Ans.EDlist
  append E.AElist into Ans.AElist
  U ← database attributes a_k occurring in E.EDlist.cont
  for each a_k in U
   if a_k occur in the SELECT-clause of the FROM-clause F_j
    SQL2EDExpList(F_j, Ans)
   else
    for each ED_p in Ans.EDlist from p = 1 to length(Ans.EDlist)
     for each AE_q in Ans.AElist from q = 1 to length(Ans.AElist)
      if AE_q.att occur in ED_q.cont
       replace occurrence of the AE_q.att into AE_p.cont
 if database attribute a occur in Ans.EDlist.cont
  replace occurrence of a in Ans.EDlist.cont into root element name in
the DTD associated with a
 return ANS
```

図 7.1 アルゴリズム **deriveDTD** で用いられる手続き SQL2EDExList

```
procedure ElementDeclareExp(sli, Ans)
        sli is formed as "st [AS a]"
        st is a ST^i-expression
        Ans is a pair(EDlist, AElist)
        EDlist is a list of pair(ele, cont)
        AElist is a list of pair(att, cont)
        ele holds a element name
        att holds a database attribute name
        cont holds a regular expression of content model

 if st is formed by tagged(element, cnt)
  append (element, RegularExp(cnt)) into Ans.EDlist
  for each tagged function tagged in cnt
     ElementDeclareExp(tagged,Ans)
  if "AS a" exists in the first argument
   append (a, element) into Ans.AElist
 else
  if "AS a" exists in the first argument
   append (a, RegularExp(st) into Ans.AElist
  else
   append (nil, RegularExp(st) into Ans.AElist
```

図 7.2 手続き SQL2EDExpList で用いられる手続き ElementDeclareExp

```
procedure RegularExp(st) result regular expression of st
        st is a ST^i-expression

 if st is formed by cat(st')
     return RegularExp(st')+
 else if st is formed by cat*(st')
     return RegularExp(st')*
 else if st is formed by st'?
     return RegularExp(st')?
 else if st is formed by st'_1+...+st'_l
     return RegularExp(st'_1),...,RegularExp(st'_l)
 else if st is formed by st'_1|...|st'_l
     return RegularExp(st'_1)|...|RegularExp(st'_l)
 else if st is formed by tagged(element, st')
     return elemet
 else if st is formed by st'/element
     return element
 else if st is formed by st'//element
     return elements
 else if st is formed by st' < element
     return element
 else if st is formed by st' << element
     return element
 else return st
```

図 7.3 手続き ElementDeclareExp で用いられる手続き RegularExp

B　実装

第6章で述べたオブジェクトリンクを有する構造化文書モデルに関して，図6.3に示した型階層のうちPARATEXT型のプロトタイプを実装した．

B.1　概要

米国のコーネル大学でSeshadriらによって開発された，型拡張可能オブジェクト関係データベースシステム *PREDATOR*[Cor97b] のversion 0.1に対して，図 6.3に示した型階層のうちPARATEXT型を追加した．オブジェクト識別子を実現していないなど，PREDATORの機能の制限により，PARATEXT型のインタフェースは第6章で述べた仕様とは異なる．

B.2　PARATEXT型のインタフェース

PREDATORには，第6章で述べた仕様でPARATEXT型を実装するには障害となるいくつかの制限事項がある．この節では，まずその制限事項について述べ，その後でこの制限事項を考慮に入れて実装したPARATEXT型のインタフェースについて記述する．

第6章で述べた仕様でPARATEXT型を実装するには障害となる制限事項には，PREDATOR自身の制限事項とPREDATORのversion固有の制限事項とがある．まず，オブジェクト識別子を扱っていないというPREDATOR自身の制限事項がある．また，SQLにおいて演算子INが使用できないというversion 0.1固有の制限事項がある．これら二つの制限事項により，**6.3.1**で導入したPARATEXT型固有の関数のうち_() はそのままの仕様で実装できない．そこで，本実装ではPARATEXT型の値の参照層に関する述語 refincludeを定義する．但し，PREDATOR version 0.1では型固有のメソッドの返り値にBOOLEAN型が適用できないので，真偽値は整数値（0と1）で代用することにした．また，オブジェクト識別子の代わりに，表名と主キーの組を用いることにした．

PARATEXT型固有の述語と関数として，以下のような仕様の述語 refinclude

と関数 hilite を，PARATEXT 型のメソッドとして実装した．

- refinclude(*tablename*,*key*) result INT
  メッセージを受けたPARATEXT型のインスタンスの参照層に，表*tablename*中の主キーの値が*key*であるオブジェクトが保持されているなら1を，保持されていないなら0を返す．

- hilite(*tablename*,*key* − *list*) result PARATEXT
  メッセージを受けたPARATEXT型のインスタンスの参照層に保持されている，表*tablename*中の主キー値が*key* − *list*に含まれているオブジェクトに対応する表記層の部分文字列がハイライトされる．

例えば，次のような問合せを記述できる．但し，表 Article の属性 body には，新聞記事が PARATEXT 型の値として保持されているものとする．

表 Article の属性 body に保持されている PARATEXT 型の新聞記事の参照層に，表 Company 中の主キーが NEC または Fujitsu であるオブジェクトが保持されている記事を，それぞれのオブジェクトが保持されている参照層に対応する表記層の部分文字列をハイライトして出力せよ．

```
SELECT  body.hilite('Company','NEC,Fujitsu')
FROM    Article
WHERE   body.refinclude('Company','NEC') = 1
OR      body.refinclude('Company','Fujitsu') = 1
```

B.3 参照層に関する検索コマンド：*refgrep*

データベースに対するコマンドとして，UNIX におけるシステムコマンド grep と類似したコマンド *refgrep* を実装した．grep が通常の表記文字列に対する与えられたパターン文字列を有する文書検索であるのに対して，refgrep は参照層の値に対するパターンで文書検索を行なう．コマンド refgrep の仕様は以下の通りである．

refgrep *pattern target*:

但し，*pattern* は参照層に対するパターンであるので SQL の問合せであり，*target* はデータベース中の値であるので，“表名．属性名” を指定する．

以下にコマンド *refgrep* の使用例を示す．

表 Article の属性 body に保持されている PARATEXT 型の新聞記事に対して，従業員が 20000 人以上の企業が出現する記事を検索せよ．

refgrep “select id from Company where totalemp > 20000;” Article.body: