(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2006-53724
(P2006-53724A)
(43)公開日　平成18年2月23日(2006.2.23)

(51)Int.Cl.		F I		テーマコード（参考）
G06F 17/30	(2006.01)	G06F 17/30	330B	5B075
G06F 12/00	(2006.01)	G06F 17/30	140	5B082
		G06F 12/00	513D	
		G06F 12/00	547Z	

審査請求 未請求　請求項の数 8　ＯＬ　（全 18 頁）

(21)出願番号　特願2004-234344 (P2004-234344)
(22)出願日　平成16年8月11日 (2004.8.11)

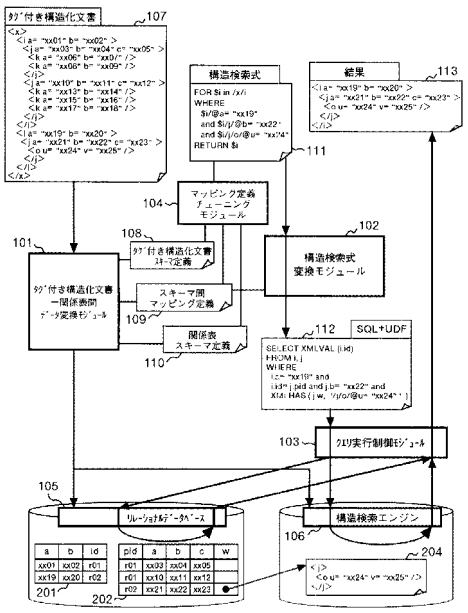
(71)出願人　000005108
株式会社日立製作所
東京都千代田区丸の内一丁目６番６号
(74)代理人　100068504
弁理士　小川　勝男
(74)代理人　100086656
弁理士　田中　恭助
(74)代理人　100094352
弁理士　佐々木　孝
(72)発明者　今木　常之
東京都国分寺市東恋ヶ窪一丁目２８０番地
株式会社日立製作所中央研究所内
(72)発明者　西澤　格
東京都国分寺市東恋ヶ窪一丁目２８０番地
株式会社日立製作所中央研究所内

最終頁に続く

(54)【発明の名称】ＸＭＬデータ管理方法

(57)【要約】
【課題】　ＸＭＬ文書－関係表間スキーママッピング定義の最適化および透過的な構造検索
【解決手段】　マッピング定義チューニングモジュール１０４は、構造検索式の発行履歴を参照して、発行頻度の高い検索処理の効率化を目的に、ＸＭＬ文書がリレーショナルデータベース１０５と構造検索エンジン１０６に適切に分解されて格納されるよう、スキーマ間マッピング定義１０９を変更する。構造検索式変換モジュール１０２は、スキーマ間マッピング定義１０９に基づいて構造検索式を変換する。クエリ実行制御モジュール１０３は、リレーショナルデータベース１０５と構造検索エンジン１０６それぞれにクエリを発行して、それぞれの結果から元の構造検索式に対する結果を再構成する。
【選択図】　図１

図 1



【特許請求の範囲】
【請求項１】
木構造型のタグ付き構造化文書を、関係データベース、および構造検索専用データベースを用いて管理する方法において、
構造化文書格納定義に従って、単一の構造化文書を、関係データベースに格納する第１の構造部分と、構造検索専用データベースに格納する第２の構造部分に分解し、
該第１の構造部分について、タグを取り除いたデータ自体を抽出して、該格納定義で対応付けられた関係表のカラムに格納し、
該第２の構造部分について、タグを含んだまま構造検索専用データベースに格納し、
元の構造化文書に対する構造検索式を、該格納定義に従って、該関係データベース用の第１の検索式と、該構造検索専用データベース用の第２の検索式に変換し、
該関係データベースに対して該第１の検索式を発行して、その結果を受け取り、該構造検索専用データベースに対して該第２の検索式を発行して、その結果を受け取り、
両結果から、該構造検索式に対する結果と等価な構造化文書を構築する手順を有することを特徴とするＸＭＬデータ管理方法。
【請求項２】
木構造型のタグ付き構造化文書を、関係データベース、および構造検索専用データベースを用いて管理する方法において、
構造化文書格納定義に従って、単一の構造化文書を、関係データベースに格納する第１の構造部分と、構造検索専用データベースに格納する第２の構造部分に分解し、
該第１の構造部分について、タグを取り除いたデータ自体を抽出して、該格納定義で対応付けられた関係表のカラムに格納し、
該第２の構造部分について、タグを含んだまま構造検索専用データベースに格納し、
元の構造化文書に対する構造検索式を、該格納定義に従って、該関係データベース用の第１の検索式と、該構造検索専用データベース用の第２の検索式に変換し、
該第２の検索式をそれと同等の構造検索処理を実行する該第１の検索式に埋め込み可能な、関係データベース検索式拡張関数で表現し、
該拡張関数を該第１の検索式に埋め込んだ拡張関数付き関係データベース用検索式を生成し、
該拡張関数付き関係データベース用検索式を、該関係データベースに対して発行し、該構造検索
式に対する結果と等価な構造化文書を取得することを特徴とするＸＭＬデータ管理方法。
【請求項３】
タグ付き構造化文書に対する構造検索式の発行履歴を記録し、
発行頻度の高い検索式の検索処理効率を指標として、
構造化文書格納定義、および、関係表のスキーマ定義を更新することを特徴とする請求項１に記載のＸＭＬデータ管理方法。
【請求項４】
タグ付き構造化文書の中で、該構造検索専用データベースに格納された部分構造化文書に対する同一の構造検索式の発行頻度が所定数を越える場合に、
該構造検索専用データベースに格納された該部分構造化文書を格納する関係表を新規に作成し、
該構造化文書格納定義に、該部分構造化文書と、該新規に作成した関係表との対応付けを追記し、
該構造検索専用データベースに格納された該部分構造化文書を、該新規に作成した関係表に格納し直すことを特徴とする請求項３に記載のＸＭＬデータ管理方法。
【請求項５】
タグ付き構造化文書の中で、あるタグで示される要素が該タグと同型の要素を子要素として持つ自己再帰的な構造を持ち、
かつ該タグのデータが関係データベースに格納されており、

かつ該タグに対する階層の深さを問わない同一の構造検索式が出現する場合に、
該自己再帰的なデータを、構造検索専用データベースに格納し直すことを特徴とする請求項３に記載のＸＭＬデータ管理方法。
【請求項６】
タグ付き構造化文書の中で、関係表データベースに格納されたデータに対する多段の階層を指定した同一の構造検索式が出現する場合に、
該データを構造検索専用データベースに格納し直すことを特徴とする請求項３に記載のＸＭＬデータ管理方法。
【請求項７】
タグ付き構造化文書の中で、関係表データベースに格納されたデータに対する多段の階層を指定した同一の構造検索式が出現する場合に、
該構造検索式に登場する全階層のデータを並べたカラムを持つ単一の関係表を新規に作成し、
該データを該新規に作成した関係表に格納し直すことを特徴とする請求項３に記載のＸＭＬデータ管理方法。
【請求項８】
タグ付き構造化文書の中で、関係表データベースに格納されたデータに対する多段の階層を指定した同一の構造検索式が複数出現する場合に、
該複数の構造検索式に登場する全階層のデータの和集合を並べたカラムを持つ単一の関係表を新規に作成し、
該データを、該新規に作成した関係表に格納し直すことを特徴とする請求項３に記載のＸＭＬデータ管理方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、データベース管理システムに関する。特にリレーショナルデータベース（あるいは関係データベース、以下、ＲＤＢという）を用いるＸＭＬ（ｅＸｔｅｎｓｉｂｌｅ Ｍａｒｋｕｐ Ｌａｎｇｕａｇｅ）文書の管理方法に係わり、特に一つのＸＭＬ文書をＸＭＬ構造検索エンジンとＲＤＢに分解して管理する方法に係わり、特に該文書に対する検索履歴に基づいて分解方法を適宜改善しつつ、ユーザに対してはこの分解方法について透過的な構造検索インタフェースを提供する方法に関する。
【背景技術】
【０００２】
現在、ＸＭＬ文書の管理に特化したネイティブＸＭＬデータベース（ＮＸＤＢ）と呼ばれる製品がいくつか存在する。しかし、ＮＸＤＢ は何れも発展途上であり、一般に大量データの管理や集計処理の目的には性能的に不十分であるため、基幹系業務などには適さないとされている。ＸＭＬの基幹業務応用は、ＸＢＲＬ（ｅＸｔｅｎｓｉｂｌｅ Ｂｕｓｉｎｅｓｓ Ｒｅｐｏｒｔｉｎｇ Ｌａｎｇｕａｇｅ）などのビジネス関連のＸＭＬ仕様の登場により今後の発展が期待されるため、大量のＸＭＬ文書を十分な性能で処理可能な技術が必要とされている。一方、主要ＲＤＢ製品においてもＸＭＬ文書管理機能が提供されている。ＲＤＢは長年にわたる改良により大量データの処理にも十分耐えうる性能を提供するため、ＸＭＬの基幹系業務応用にも適していると言える。
【０００３】
主要ＲＤＢ製品に関する代表的なＸＭＬ文書管理方法については、非特許文献１および非特許文献２に記載されている。
【０００４】
非特許文献１の方法は、管理対象であるＸＭＬ文書の文書スキーマと、ＲＤＢの関係表スキーマとの間の対応関係に従って、ＸＭＬ文書に含まれるタグ付けされたデータを構造分解して、複数の関係表に分けて値単位で格納する。このような格納方式を、以下、ＸＭＬ文書スキーマ－関係表スキーマ間のマッピング方式と呼ぶ。非特許文献１の方法は、Ｘ

ＭＬ文書スキーマの定義を元に、その定義に妥当であるＸＭＬ文書を格納するための関係表スキーマの定義、および、該ＸＭＬ文書スキーマと該関係表スキーマとの間の対応関係の定義（以下、スキーママッピング定義という）を、自動的に作成する。またＸＭＬの標準検索仕様ＸＰａｔｈ形式の構造検索式を、スキーママッピング定義に従って、関係表検索式（以下、ＳＱＬ式という）に自動変換する。

【０００５】

非特許文献２の方法も、基本的にスキーママッピング方式である。ただしスキーママッピング定義は、ＲＤＢに格納されたデータからＸＭＬ文書を構築する方向で、ユーザがマニュアルで定義する。またＸＭＬの標準検索仕様ＸＱｕｅｒｙ形式の構造検索式を、スキーママッピング定義に従って、ＳＱＬ式に自動変換する。

【０００６】

【非特許文献１】店ML Schemas in Oracle XML DB　R. Murthy, S, Banerjee; VLDB2003

【非特許文献２】轍uerying XML Views of Relational Data　J. Shanmugasundaram, et al., VLDB2001

【発明の開示】

【発明が解決しようとする課題】

【０００７】

一般に、ＲＤＢでのＸＭＬ文書管理は、ＸＭＬ文書に含まれるタグ付けされたデータを構造分解して、複数の関係表に分けて値単位で格納する方式に則っている。このようなＸＭＬ文書スキーマ－関係表スキーマ間のスキーママッピング方式には、ＸＭＬ文書を管理するうえで以下のような欠点が存在する：

（ａ）検索効率を考慮したマッピング定義

一般に、マッピング方法の違いによって検索性能は異なってくる。最適な検索性能を得るためには、マッピング定義のチューニングが必要であるが、ユーザにとってこの作業は大変な負担となる。

（ｂ）非定型データの管理

ＸＭＬでは、厳密なスキーマ定義に従わない非定型部分データを文書中に含むことが可能であり、これによるデータ表現の柔軟性がＸＭＬ利用拡大の大きな要因となっているが、ＲＤＢではこのようなデータをＬＯＢとよばれる一次元の文字列データとして管理することになるため、その部分に対して高度な検索をかけることができない。

（ｃ）複雑な構造を持つ文書の管理

ＸＭＬでは、木構造に基づいたデータモデルにより、複雑なデータ構造を表現することが可能である。一方、関係表は一次元の値の集まりを単位としてデータを管理するため、木構造のような複雑なデータは、複数の関係表間における外部参照関係によって表現しなくてはならない。しかし、ＸＭＬ文書スキーマの階層が深い場合は多数の関係表に分けて管理することになるため、検索効率および格納効率の点で望ましくない。このように、ＲＤＢとＸＭＬとのデータモデルの違いに基づく関係表での管理が非効率的なＸＭＬ文書が存在する。

（ｄ）検索指定方法

関係表にスキーママッピングしたＸＭＬ文書に対する検索は、そのマッピング定義に沿って定義される必要があるため、ユーザがマッピング定義を意識して関係表検索式（以下、ＳＱＬ式）を記述する必要がある。また、（ａ）の課題にあげたように検索効率性を考慮してマッピング定義を変更した場合は、ＳＱＬ式も記述し直す必要がある。一般に、ユーザにとっては、ＸＭＬ文書スキーマのみを意識して構造検索を指定できることが理想であり、ＸＭＬ文書の管理においては本来存在しないこれらの必要性は、ユーザにとって大変な負担となる。

【０００８】

上記のＸＭＬ文書スキーマ－関係表スキーマ間スキーママッピング方式における（ａ）～（ｄ）の欠点を克服するために、本発明ではそれぞれ以下の課題を解決することを目的とする。

【０００９】
第一に、スキーママッピング定義の自動チューニング機能を提供すること。
【００１０】
第二に、従来ＬＯＢで管理していたような非定型部分データに対しても構造検索機能を提供すること。
【００１１】
第三に、関係表での管理が非効率的なデータを切り分けて、効率的な手段で管理すること。
【００１２】
第四に、ＸＭＬ文書の関係表への格納方法に関して、透過なＸＭＬ文書の構造検索機能を提供すること。
【課題を解決するための手段】
【００１３】
まず、第二、第三の課題を解決するために、ＲＤＢの外部データベース、あるいはＲＤＢのプラグインとして存在するＸＭＬ構造検索エンジンと連携する。
【００１４】
従来、関係表のＬＯＢカラムに格納していた非定型部分データを構造検索エンジンに格納することによって、第二の課題を解決する。関係表での管理が非効率的なデータも、代わりに構造検索エンジンで管理することによって、第三の課題を解決する。
【００１５】
また、第一の課題を解決するために、クエリ発行履歴を参照し、頻出クエリの検索性能効率化を指標として適切なスキーママッピング定義を導出するマッピング定義チューニングモジュールを導入する。第三の課題解決における関係表での管理が不適切なＸＭＬ部分データの切り分けもこのモジュールで行う。
【００１６】
さらに、第四の課題を解決するために、ＸＭＬ文書に対する構造検索式を、スキーママッピング定義に基づいてＳＱＬ式に自動変換するクエリリライト機能を提供する。検索対象が構造検索エンジンで管理されている部分データにも及ぶ場合は、この検索エンジンへの検索式をＵＤＦ（Ｕｓｅｒ　Ｄｅｆｉｎｅ　Ｆｕｎｃｔｉｏｎ）として含むＳＱＬ式に変換する。このクエリリライトにより、ユーザは、第一～第三の課題解決における、ＸＭＬ文書の関係表および構造検索エンジンへの格納方法の違いに対して、透過的に構造検索指定が可能となる。
【発明の効果】
【００１７】
ＸＭＬ文書スキーマ－関係表スキーマ間のスキーママッピング方式において、
（１）クエリ発行履歴に基づいて、検索処理コストを削減するようにスキーママッピング定義を自動的に改善することが可能である。
（２）非定型の部分データを構造検索エンジンで管理することによって、該部分データに対する構造検索が可能である。
（３）関係表での管理が非効率的なデータを切り分けて構造検索エンジンで管理することによって、非効率的な検索処理を回避することが可能である。
（４）クエリリライト機能により、ＸＭＬ文書の関係表および構造検索エンジンへの格納方法に関し、透過的にＸＭＬ文書に対する構造検索を指定することが可能である。
【発明を実施するための最良の形態】
【００１８】
以下、本発明の実施の一形態を、図面を参照しながら説明する。なお簡単のために、本明細書中では以下に述べる発明の実施の形態を単に「本実施例」と呼ぶことにする。
【００１９】
図１を用いて、本実施例の概略構成について説明する。
【００２０】

本実施例のシステムは、以下に挙げる４つのモジュールを基本構成要素として成立している：

・　タグ付き構造化文書－関係表間データ変換モジュール１０１
・　構造検索式変換モジュール１０２
・　クエリ（問合せ）実行制御モジュール１０３
・　マッピング定義チューニングモジュール１０４

以下、それぞれのモジュールについて概説する。

【００２１】

タグ付き構造化文書－関係表間データ変換モジュール１０１は、タグ付き構造化文書（以下、ＸＭＬ文書という）１０７を構造分解してタグを取り除いたデータ本体を、リレーショナルデータベース（以下、ＲＤＢという）１０５の関係表のカラムに対応して属性値を格納する。ただし一部のＸＭＬ文書については、タグが付いた部分木単位でＸＭＬ文書専用の構造検索エンジン１０６に格納する。ＸＭＬ文書のうち、ＲＤＢ１０５に格納する部分、格納先の関係表カラム、および構造検索エンジン１０６にタグごと格納する部分の区別は、タグ付き構造化文書スキーマ定義（以下、ＸＭＬ文書構造定義という）１０８、スキーマ間マッピング定義１０９、および関係表スキーマ定義１１０に従って決定される。

【００２２】

ＸＭＬ文書構造定義１０８はＸＭＬ文書の構造定義を、関係表スキーマ定義１１０は関係表のスキーマ定義をそれぞれ表す。ＸＭＬ文書１０７は、ＸＭＬ文書構造定義１０８に対して妥当である必要があるし、ＲＤＢ１０５に格納されている関係表２０１、２０２は、関係表スキーマ定義１１０に従って構成されている。スキーマ間マッピング定義１０９は、ＸＭＬ文書のノード値（タグで修飾された要素値、あるいは属性値）とそれを格納する関係表のカラムの対応付けを定義する。

【００２３】

構造検索式変換モジュール１０２は、ＸＱｕｅｒｙ、ＸＰａｔｈなどのＸＭＬの標準検索仕様に従ってユーザが定義した構造検索式１１１を、ＲＤＢ１０５用の検索仕様であるＳＱＬ言語の検索式（以下、ＳＱＬ式という）１１２に変換するモジュールである。この変換は、スキーマ間マッピング定義１０９に従って行われる。検索範囲が構造検索エンジンに格納した部分ＸＭＬ文書にも及ぶ場合には、ＳＱＬ式１１２中に構造検索エンジン用の拡張関数（ＵＤＦ）を埋め込んだ式に変換する。

【００２４】

クエリ実行制御モジュール１０３は、ＵＤＦを含んだＳＱＬ式１１２を、ＳＱＬ部分とＵＤＦ部分に分離し、前者をＲＤＢ１０５に、後者を構造検索エンジン１０６に対して発行し、その結果を統合して、元の構造検索式１１１に対する結果１１３を構築するモジュールである。このモジュールは、ＲＤＢ１０５にプラグイン処理機構がある場合は、その機能上で自然に実現される（この場合については、図３を用いて後述する）。

【００２５】

マッピング定義チューニングモジュール１０４は、ユーザの構造検索式１１１の発行履歴を参照して、頻出する検索式の処理の効率化を指標として、ＸＭＬ文書構造定義１０８を参照しつつ、スキーマ間マッピング定義１０９、および関係表スキーマ定義１１０を適宜更新する。関係表スキーマ定義１１０の更新に伴う関係表の変更は、ＲＤＢ１０５の機能に任せる。

【００２６】

以下、図１に示すシステムを実現するためのハードウェア構成について説明する。本システムは、ハードウェア的にはＣＰＵ、メモリ、外部記憶装置、入力装置、表示装置などを備える１台又は複数台の計算機によって構成される。ＸＭＬ文書１０７、ＸＭＬ文書構造定義１０８、スキーマ間マッピング定義１０９および関係表スキーマ定義１１０は、ファイルとして記憶装置上に格納される。構造検索式１１１は、テキストエディタを介して入力装置から入力されるか、図示しないアプリケーションプログラムを介して生成され、

メモリに格納される。結果１１３は、メモリに格納され、表示装置やプリンタに出力されるか、さらに処理のためにアプリケーションに渡されるデータである。構造検索エンジン１０６は、記憶装置に格納されるＸＭＬ文書の木構造ファイルを有し、これらファイルを管理するためのデータベース・マネージメント・システムである。タグ付き構造化文書－関係表間データ変換モジュール１０１、構造検索式変換モジュール１０２、クエリ実行制御モジュール１０３およびマッピング定義チューニングモジュール１０４は、計算機のメモリに格納され、そのＣＰＵによって実行されるプログラムである。ＲＤＢ１０５は、記憶装置上に格納されるリレーショナルデータベースを有し、このデータベースを管理するためのデータベース・マネージメント・システムである。データ変換モジュール１０１、構造検索式変換モジュール１０２、クエリ実行制御モジュール１０３およびマッピング定義チューニングモジュール１０４の一部又は全部がＲＤＢ１０５に組み込まれて実装されてもよい。これらモジュール、ＲＤＢ１０５および構造検索エンジン１０６は、同一の計算機上で実行されてもよいし、その一部又は全部がネットワークを介して異なる計算機上で実行されてもよい。また本システムは、クライアント－サーバ型のシステムで実現されてもよい。

【００２７】

以上が、本実施例の概略である。以降、本実施例における、データ変換モジュール１０１の動作概要を図２で、構造検索式変換モジュール１０２の動作概要を図３で、クエリ実行制御モジュール１０３の動作概要を、実現方法のバリエーション別に図３、図４、図５を用いて説明する。

【００２８】

図２を用いて、データ変換モジュール１０１の動作について説明する。本説明では、ＸＭＬ文書構造定義１０８に対して妥当であるＸＭＬ文書１０７をＲＤＢ１０５に格納する場合を例にとる。ＸＭＬ文書構造定義１０８は、ルート要素ｘの下に複数のｉ要素が出現し、各ｉ要素は属性ａ，ｂを持ち、さらにその下には複数のｊ要素が出現し、各ｊ要素は属性ａ，ｂ，ｃを持ち、さらにその下には複数のｋ要素が出現し、各ｋ要素は属性ａ，ｂを持つことを表している。ｓｔｒは文字列を示す。×０…ｎは、対応する要素が０個からｎ個まで出現可能なことを示す。また、各ｊ要素の下には、ｋ要素以外にも任意の要素が登場し得ることを｛ＡＮＹ｝で示す。なお、図２のＸＭＬ文書構造定義１０８の記法は実施例を限定するものではなく、同様の意味を表現し得るＸＭＬ文書構造の定義仕様であれば、どのような記法でも適用可能である。例えば、ＸＭＬ文書の標準的な文書構造定義仕様であるＤＴＤ（Ｄｏｃｕｍｅｎｔ　Ｔｙｐｅ　Ｄｅｆｉｎｉｔｉｏｎ）では、上記と同様の文書構造定義を以下のように表現する：

```
＜!ＥＬＥＭＥＮＴ　ｘ　ｉ＊＞
＜!ＥＬＥＭＥＮＴ　ｉ　ｊ＊＞
＜!ＡＴＴＬＩＳＴ　ｉ
　　ａ　ＣＤＡＴＡ　＃ＲＥＱＵＩＲＥＤ
　　ｂ　ＣＤＡＴＡ　＃ＲＥＱＵＩＲＥＤ＞
＜!ＥＬＥＭＥＮＴ　ｊ　ＡＮＹ＞
＜!ＡＴＴＬＩＳＴ　ｊ
　　ａ　ＣＤＡＴＡ　＃ＲＥＱＵＩＲＥＤ
　　ｂ　ＣＤＡＴＡ　＃ＲＥＱＵＩＲＥＤ
　　ｃ　ＣＤＡＴＡ　＃ＲＥＱＵＩＲＥＤ＞
＜!ＥＬＥＭＥＮＴ　ｋ　ＥＭＰＴＹ＞
＜!ＡＴＴＬＩＳＴ　ｋ
　　ａ　ＣＤＡＴＡ　＃ＲＥＱＵＩＲＥＤ
　　ｂ　ＣＤＡＴＡ　＃ＲＥＱＵＩＲＥＤ＞
```

一方、ＸＭＬ文書１０７を格納する関係表２０１、２０２、および２０３のスキーマは、関係表スキーマ定義１１０で与えられる。この例では、関係表ｉがａ，ｂ，ｉｄの３つのカラムを、関係表ｊがｐｉｄ，ａ，ｂ，ｃ，ｗ，ｉｄの６つのカラムを、関係表ｋがｐ

ｉｄ，ａ，ｂ，ｉｄの４つのカラムを持つことをそれぞれ表現している。ここでｉｄとｐｉｄは親と子のつながりを示す識別子である。ｉｄは自身を親とする識別子、ｐｉｄは子に設けられる識別子であり、どの親に接続するかを示す識別子である。ｉｄとｐｉｄが同一である場合に親子関係の接続があることを示す。なお、図２の関係表スキーマ定義１１０の記法は実施例を限定するものではなく、同様の意味を表現し得る関係表スキーマの定義仕様であれば、どのような記法でも適用可能である。例えば、一般に関係表のスキーマはＳＱＬ式で表作成時に定義するため、そのＳＱＬ式を関係表スキーマ定義１１０として利用できる。

【００２９】

上記のＸＭＬ文書構造定義１０８および関係表スキーマ定義１１０に基づいて、両定義間の値の対応付けを定義するのが、スキーマ間マッピング定義１０９である。１行目の「／ｘ／ｉ／＠ａ⇔ｉ．ａ」は、ＸＭＬ文書１０７のｉ要素の属性ａの値を、関係表ｉ（２０１）のカラムａに格納することを表現している。２，４，５，６，８，９行目も同様である。３行目の「／ｘ／ｉ／ｊ／．．⇔ｊ．ｐｉｄ＝ｉ．ｉｄ」は、ｊ要素の親を関係表ｊ（２０２）のカラムｐｉｄで示し、関係表ｉ（２０１）のカラムｉｄを外部参照していることを表現している。７行目も同様に、関係表ｋ（２０３）と関係表ｊ（２０２）の間の外部参照を表現している。１０行目は、ＸＭＬ文書構造定義１０８において定義されていないｊ要素の部分内容を関係表ｊ（２０２）のカラムｗに格納することを示している。ただし実際にはその部分構造は、タグごと構造検索エンジン１０６に格納され、その格納イメージ２０４に対して構造検索エンジン１０６上で付されたファイルのＩＤ（この例ではｘｉ－ａ）のみが関係表ｊ（２０２）のカラムｗに格納される。

【００３０】

データ変換モジュール１０１は、まず関係表スキーマ定義１１０に基づきＲＤＢ１０５を介してその記憶領域内に関係表ｉ，ｊ，ｋの各領域を確保する。次にデータ変換モジュール１０１は、ＸＭＬ文書１０７からｉ，ｊ，ｋ又はｏ要素の１つを取り出し、ＸＭＬ文書構造定義１０８を参照して取り出した要素の形式がそのスキーマ定義に合致するか否かチェックする。定義に合致すれば、データ変換モジュール１０１は、スキーマ間マッピング定義１０９を参照して取り出した要素の各属性の属性値をＲＤＢの該当する関係表の１レコードとして格納し、そのレコードのｉｄとｐｉｄを設定する。ｉｄはその関係表のレコード位置に応じた識別子を生成して設定する。ｐｉｄにはメモリに保存された親要素のｉｄがあればそのｉｄを格納する。次にデータ変換モジュール１０１は、当該要素の要素名とそのｉｄをメモリに一時保存する。データ変換モジュール１０１は、ＸＭＬ文書１０７からｏ要素を取り出したとき、ＸＭＬ文書構造定義１０８を参照して取り出した要素がＡＮＹに相当することを認識し、関係表ｊの該当するレコードのカラムに構造検索エンジン１０６で指定されたファイルＩＤを設定し、取り出したｏ要素にｊのタグを付けた部分構造を構造検索エンジン１０６に送る。構造検索エンジン１０６は、受け取った部分構造をその記憶領域に格納イメージ２０４として格納する。データ変換モジュール１０１は、ＸＭＬ文書１０７のすべての要素を取り出し終わるまでＸＭＬ文書１０７から次の要素を取り出すステップに戻って上記処理を繰り返す。

【００３１】

図３を用いて、構造検索式変換モジュール１０２の動作について説明する。本説明では、図２の例で取り上げたＸＭＬ文書構造定義１０８、スキーマ間マッピング定義１０９、および関係表スキーマ定義１１０に従って、データ変換モジュール１０１によりＲＤＢ１０５の関係表２０１、２０２、２０３、および構造検索エンジン１０６に格納イメージ２０４として格納されたＸＭＬ文書１０７に対して、ＸＱｕｅｒｙ標準に従って記述された構造検索式１１１を処理する場合を例にとる。

【００３２】

構造検索式１１１は、ＦＯＲ句により変数＄ｉにｉ要素名を代入する。従って、ＷＨＥＲＥ句はｉ要素の属性ａが“ｘｘ１９”であり、そのｉ要素は、属性ｂが“ｘｘ２２”であるようなｊ要素を子要素として持ち、さらにそのｊ要素は、属性ｕが“ｘｘ２４”であ

るようなｏ要素を子要素として持つことを条件としている。さらに、上記の条件で抽出したｉ要素全体を結果として取得することを要求している。

【００３３】

構造検索式変換モジュール１０２は、スキーマ間マッピング定義１０９を参照して、上記の意味を持つ構造検索式１１１を構造指定ＵＤＦを含むＳＱＬ式１１２に変換する。マッピング定義１０９に従うと、上記の検索式の意味は、関係表ｉ（２０１）のカラムａ，関係表ｊ（２０２）のカラムｂ，および、関係表ｊ（２０２）のカラムｗに対して条件を指定していることと等価である。ｏ要素はｊ要素の子要素としては定義されていないため、マッピング定義１０９の１０行目を適用して、関係表ｊ（２０２）のカラムｗに対する条件となる。但し、このカラムは構造検索エンジン１０６に格納されたイメージ２０４への参照であるため、この条件は構造指定ＵＤＦで表現される。

【００３４】

以上から、構造検索式１１１を変換したクエリ１１２は、関係表ｉ（２０１）から、カラムａが“ｘｘ１９”であるようなレコードを抽出し、関係表ｊ（２０２）のレコードから、カラムｂが“ｘｘ２２”、かつ、カラムｗで参照される構造検索エンジン１０６の格納イメージ（２０４）が、属性ｕの値＝“ｘｘ２４”であるようなｏ要素を含んでいるようなレコードを抽出し、さらに両レコードの間に外部参照関係が成り立っていることを条件として指定するＳＱＬ式として生成される。関係表ｊ（２０２）のカラムｗに対する構造指定は、ＸＭＬＨＡＳというＵＤＦで表現される。これは、第一引数で指定したカラムの値が指し示す構造検索エンジン１０６上の格納イメージが、第二引数で指定したＸＰａｔｈ構造式にマッチする部分データを含むか否かを判定するブール関数である。

【００３５】

また構造検索式１１１は、上記の条件を満たすｉ要素全体を結果として取得することを要求しているため、ＳＱＬ式１１２のＳＥＬＥＣＴ句には、ＸＭＬＶＡＬというＸＭＬ文書構築ＵＤＦを指定する。これは、引数でＩＤ指定された要素について、全ての子孫要素をＲＤＢ１０５および構造検索エンジン１０６から抽出して、ＸＭＬ文書を再構築して返すスカラ関数である。

【００３６】

図３～図５を用いて、クエリ実行制御モジュール１０３の動作を説明する。

【００３７】

図３は、ＲＤＢ１０５がプラグイン処理機構３０１を持つ場合を示している。この場合、クエリ実行制御モジュール１０３が実現すべき機能はＲＤＢ１０５に組み込まれていることになる。ここでは、説明のためにこの機能を単独で実現するプログラムをクエリ実行制御モジュールと呼び、ＲＤＢ１０５自身と区別する。クエリ実行制御モジュール１０３は、ＵＤＦを含むＳＱＬ式１１２をネイティブなＳＱＬ部とＵＤＦ部に分離し、ＲＤＢ１０５がＳＱＬ部を処理し、プラグイン処理機構３０１がＵＤＦ部を処理する。構造指定ＵＤＦであるＸＭＬＨＡＳは、実際には構造検索エンジン１０６で処理されるため、ほとんどの構造検索エンジンが対応している構造検索仕様ＸＰａｔｈのクエリ３０２として、該エンジンに対して発行する。但し、ＸＭＬＨＡＳがＲＤＢ１０５に組込みのプラグインとして実現されている場合は、ＲＤＢ１０５上で直接この構造指定ＵＤＦを実行する。ＸＭＬ文書構築ＵＤＦであるＸＭＬＶＡＬも、プラグイン処理機構３０１上で実行する。

【００３８】

クエリ実行制御モジュール１０３は、クエリを実行する際に、先に構造検索エンジン１０６に対する条件でデータを絞るか、あるいは関係表に対する条件でデータを絞るか、クエリの処理効率を指標にして決定する。

【００３９】

ＳＱＬ式１１２を例にとると、前者の場合は、まずＸＭＬＨＡＳの条件判定に適合する構造検索エンジン１０６上の格納イメージ２０４を抽出し、そのＩＤ（この例ではｘｉ－ａ）と関係表ｊ（２０２）のカラムｗの値が一致することも条件に含めて、関係表２０１，２０２からデータを抽出することになる。

【００４０】
一方、後者の場合は、まず関係表ｉ（２０１）のカラムａと関係表ｊ（２０２）のカラムｂ、および関係表ｉ（２０１）のカラムｉｄと関係表ｊ（２０２）のカラムｐｉｄの間の外部参照関係を条件にデータを絞り、抽出した関係表ｊ（２０２）のレコードのカラムｗの値が指し示す、構造検索エンジン１０６上の格納イメージ２０４に対してＸＭＬＨＡＳによる条件判定を行う。
【００４１】
上記のようなクエリ実行手順の決定は、一般的なＲＤＢ１０５が備える実行計画決定処理により最適化される。従って、プラグイン処理機構３０１を備えるＲＤＢ１０５を利用する場合は、クエリ実行制御モジュール１０３を新たに設ける必要はない。
【００４２】
一方、ＲＤＢ１０５にプラグイン処理機構３０１が備わっておらず、クエリ実行制御モジュール１０３をＲＤＢ１０５の外部に設ける必要がある場合の動作概要について、図４、図５を用いて説明する。クエリ実行制御モジュール１０３をＲＤＢ１０５の外に新たに設ける場合、上記のようなクエリ実行手順も独自に決定する必要がある。
【００４３】
図４を用いて、先に構造検索エンジン１０６に対する条件でデータを絞る場合について説明する。クエリ実行制御モジュール１０３がＵＤＦを含むＳＱＬ式１１２を受けとると、該モジュール内のＵＤＦ分離処理４０１がネイティブなＳＱＬ式４０３とＵＤＦ部に分離する。次にクエリ実行制御モジュール１０３は、ＵＤＦ部を構造検索エンジン１０６に対するＸＰａｔｈ検索式３０２として発行し（丸付き数字１）、この式にマッチするデータを含む格納イメージ２０４のＩＤ（この例ではｘｉ－ａ）を獲得し、ＲＤＢ１０５に一時表ｘ（４０４）として格納する。次にクエリ実行制御モジュール１０３は、関係表ｊ（２０２）のカラムｗに格納されているＩＤが、この一時表ｘに含まれることも条件にして、ＳＱＬ式４０３により関係表２０１，２０２からデータを抽出する（丸付き数字２）。ただしＳＱＬ式４０３のｘは、一時表ｘを意味し、ｘ．ｉｄは一時表ｘのｉｄカラムを意味する。ＳＱＬ式４０３による検索の結果として、クエリ実行制御モジュール１０３にはｉ．ｉｄとして“ｒ０２”というデータが返る。
【００４４】
このようにしてタグ付き構造化文書再構成処理４０２は、抽出したＩＤを持つｉ要素を関係表２０１～２０３および構造検索エンジン１０６の格納イメージ２０４のデータから再構成する。このため、タグ付き構造化文書再構成処理４０２は、スキーマ間マッピング定義１０９を参照して関係表よりデータを抽出するＳＱＬ式４０５～４０７を作成し、これらのＳＱＬ式をＲＤＢ１０５に対して発行する（丸付き数字３）。これらは、関係表間の外部参照関係に基づき、抽出したＩＤ“ｒ０２”を持つｉ要素の全子孫要素を抽出するものである。ここでＳＱＬ式４０５のｉｉｄには“ｒ０２”が代入される。ｊｉｄには何も代入されず、結果的にはＳＱＬ式４０７の結果は返らない。本例の場合にはＳＱＬ式４０７がなくても構わない。一方、ｉ要素は一部に構造検索エンジン１０６に保存された格納イメージ２０４のデータも含むため、タグ付き構造化文書再構成処理４０２は、それを取得するためのクエリ４０８を、構造検索エンジン１０６に対して発行し（丸付き数字３’）し、格納イメージ２０４を取得する。タグ付き構造化文書再構成処理４０２は、ＸＭＬ文書構造定義１０８、スキーマ間マッピング定義１０９および関係表スキーマ定義１１０を参照し、抽出したデータと格納イメージ２０４からＸＭＬ文書を再構成し、結果１１３を得る（丸付き数字４）。
【００４５】
図５を用いて、先に関係表２０１，２０２に対する条件でデータを絞る場合について説明する。この場合、ＵＤＦ分離処理４０１は、ＵＤＦを含むＳＱＬ式１１２を、ネイティブＳＱＬ式５０２のようなクエリに分離する。クエリ実行制御モジュール１０３は、このＳＱＬ式をＲＤＢ１０５に発行し（丸付き数字１）、関係表２０１，２０２に対する条件でデータを絞る。その際に、関係表ｊ（２０２）のカラムｗの値も同時に抽出する。クエ

リ実行制御モジュール１０３は、ｉ．ｉｄとして“ｒ０２”、ｊ．ｗとして“ｘｉ－ａ”という値を受け取る。構造判定処理５０１は、ＳＱＬ式５０２の結果を受け取り、格納イメージＩＤ“ｘｉ－ａ”とＸＰａｔｈ式３０２を構造検索エンジン１０６に送り（丸付き数字２）、これらの条件に合う格納イメージが構造検索エンジン１０６に登録されているか否かを判定する。構造検索エンジン１０６に該当するデータがあれば、格納イメージＩＤ“ｘｉ－ａ”をタグ付き構造化文書再構成処理４０２に渡す。以降の処理は、図４の場合と同一である。

【００４６】

なお、以上の説明では、構造検索式変換モジュール１０２とクエリ実行制御モジュール１０３を区別して説明したが、これらは一つのモジュールとして実現されていても構わない。その場合は、ＵＤＦを含むＳＱＬ式１１２を生成せずに、構造検索式１１１から直接ＳＱＬ式４０３、５０２に変換するような実施例もあり得ることは自明である。

【００４７】

以降、本実施例におけるマッピング定義チューニングモジュール１０４が行う具体的なマッピング定義改善の処理手順について、図６、図７（ａ）、図７（ｂ）、図８を用いて説明する。

【００４８】

図６を用いて、ＸＭＬ文書構造定義で明示的に定義されていない部分データについての検索頻度が所定数を越える場合の、マッピング定義改善処理について説明する。

【００４９】

システムは、タグ付き構造化文書に対する構造検索式の発行履歴を図示しない検索履歴データベースに記録する。マッピング定義チューニングモジュール１０４は、検索履歴データベースを参照し、同一のＵＤＦについての構造検索式の発行頻度を計数する。図２～図５の例における構造検索式１１１中のｏ要素についての条件指定のように、ＸＭＬ文書構造定義に登場せず、従って明示的にスキーマ間マッピングを定義していない部分に対して検索が頻出する場合、マッピング定義チューニングモジュール１０４は、この部分を格納する関係表とマッピング定義を自動的に生成する。

【００５０】

ＲＤＢの検索処理性能は、長年の改良の結果、一般的な構造検索エンジンに比べ高速であり、また他の関係表データに対するのと同時に条件指定することを考慮した場合、データは、ＲＤＢ外部の構造検索エンジンではなく、可能な限り関係表で管理した方が効率的に優れるため、このようなマッピングの変更は性能改善に繋がる。

【００５１】

本例では、マッピング定義チューニングモジュール１０４は、ＲＤＢ１０５上に関係表ｏ（６０１）を新規に作成し、関係表ｊ（２０２）との間に外部参照関係を規定する。この時、関係表スキーマ定義１１０は、関係表スキーマ定義６０３に変更される。関係表ｏは、カラムｐｉｄ，ｕ，ｖ，ｉｄの４つのカラムを持つと定義される。同時に、スキーマ間マッピング定義１０９は、スキーマ間マッピング定義６０４に変更される。マッピング定義チューニングモジュール１０４は、マッピング定義１０９の１０行目にあった未定義部分を構造検索エンジンにマッピングすることを表す記述を削除し、新たに１０行目に関係表ｊと関係表ｏの外部参照関係を表す記述、および１１，１２行目に、ｏ要素の各属性と関係表ｏの各カラムとの対応を表す記述を追加する。またマッピング定義チューニングモジュール１０４は、ＸＭＬ文書構造定義１０８の｛ＡＮＹ｝を＜ｏ　ｕ＝“ｓｔｒ”ｂ＝“ｓｔｒ”／＞×０…ｎに変更する。

【００５２】

マッピング定義チューニングモジュール１０４が実行する処理手順の詳細は次の通りである。マッピング定義チューニングモジュール１０４は、スキーマ間マッピング定義１０９の各定義レコードをたどり、ＸＭＬ文書構造定義１０８に定義されていない要素の部分内容を見つける。次にマッピング定義チューニングモジュール１０４は、関係表スキーマ定義１１０を参照してその部分内容に定義された関係表とカラムの識別子を取得する。次

にマッピング定義チューニングモジュール１０４は、ＲＤＢ１０５に対してＳＱＬ検索式を送付し、その関係表とカラム位置の属性値を取得する。その属性値が構造検索エンジン１０６のファイルＩＤを示しているので、マッピング定義チューニングモジュール１０４は、構造検索エンジン１０６からその格納イメージ２０４を取得する。次にマッピング定義チューニングモジュール１０４は、ＲＤＢ１０５を介してその記憶領域内に関係表ｏの記憶領域を確保する。次にマッピング定義チューニングモジュール１０４は、上記のデータ変換モジュール１０１の処理手順に従って格納イメージ２０４からｏ要素を取り出し、ＲＤＢ１０５を介して関係表ｏを作成する。次にマッピング定義チューニングモジュール１０４は、関係表スキーマ定義１１０に関係表ｏの定義を追加し、関係表スキーマ定義１１０を関係表スキーマ定義６０３に更新する。次にマッピング定義チューニングモジュール１０４は、スキーマ間マッピング定義１０９に関係表ｏについてのマッピング定義を追加し、スキーマ間マッピング定義１０９をスキーマ間マッピング定義６０４に更新する。次にマッピング定義チューニングモジュール１０４は、ＸＭＬ文書構造定義１０８の定義文をたどり、未定義の要素を見つけ、ｏ要素の定義に置き換える。次にマッピング定義チューニングモジュール１０４は、構造検索エンジン１０６から格納イメージ２０４を削除する。

【００５３】

以上の変更が加えられたマッピング定義においては、構造検索式１１１は、構造検索式変換モジュール１０２によって、ＳＱＬ式６０２に変換されることになる。このＳＱＬ式は、構造指定ＵＤＦを含まないため、ＲＤＢ１０５で処理するのに望ましい形となっている。

【００５４】

図７（ａ）及び図７（ｂ）を用いて、再帰的な構造を持つＸＭＬデータ管理の改善を実現する処理手順について説明する。図７（ａ）に示すように、ＸＭＬ文書７０１は、ＸＭＬ文書構造定義７０２に妥当である、自己再帰的な構造を持つ。すなわちｊ要素の子要素としてｊ要素自身が複数出現する。このようなＸＭＬ文書の関係表への格納方法は、スキーマ間マッピング定義７０３、および関係表スキーマ定義７０４によって定義される。マッピング定義７０３の３行目は、ｊ要素の親はｉ要素かｊ要素であり、その区別を関係表ｊのカラムｐｒｌの値（“ｉ”または“ｊ”）で表現することを意味している。ＸＭＬ文書７０１の格納先となる関係表は、関係表ｉ（７０５）および関係表ｊ（７０６）の２つで、関係表ｉと関係表ｊの間の外部参照関係、および関係表ｊ内部での自己参照関係が規定されている。

【００５５】

一方、構造検索式７０７は、属性ａの値が“ｘｘ０１”であるｉ要素の子孫要素として任意の階層に出現する、属性ａの値が“ｘｘ１８”であるようなｊ要素を抽出することを要求している。このことをＳＱＬ式で表現するには、再帰クエリを利用する必要がある。構造検索式７０７は、構造検索式変換モジュール１０２によって、ＳＱＬ式７０８に変換される。このＳＱＬ式は、再帰的に関係表ｊ（７０６）の自己参照関係を辿って、一時表ｔｍｐに、ｉ要素の全ての子孫を抽出して行く再帰クエリである。

【００５６】

しかし、一般的にＲＤＢの再帰クエリは効率の悪い処理であり、このような構造検索式が頻出する場合には、上記のようなマッピング定義は好ましくない。

【００５７】

これに対し、再帰構造を持つＸＭＬ部分データを、敢えて構造検索エンジン１０６に格納することで改善を図る。一般的に構造検索エンジンは、階層の深いデータに対しても妥当な性能で検索処理が可能であるように設計されているため、関係表で管理するよりも効率が良い場合がある。

【００５８】

図７（ｂ）に示すスキーマ間マッピング定義７０９は、上記のスキーマ間マッピング定義７０３における３～６行目のｊ要素を関係表ｊ（７０６）に対応付けている記述を削除

し、新たに３行目に、ｉ要素の子孫を全て構造検索エンジン１０６に格納する記述を追加している。関係表スキーマ定義７１０は、関係表ｉ（７０５）に構造検索エンジン１０６での格納イメージのＩＤを格納するカラムｗを追加している。

【００５９】

以上のマッピング定義においては、ＸＭＬ文書７０１は、関係表７０５および構造検索エンジン１０６の格納イメージ７１１，７１２に分解して格納される。また構造検索式７０７は、構造検索式変換モジュール１０２によって、ＵＤＦを含むＳＱＬ式７１３に変換される。ＳＱＬ式７１３は、ＳＱＬ式７０８と比較して再帰を含まないシンプルなクエリとなっており、ＲＤＢ１０５と構造検索エンジン１０６の適切な使い分けが成される。

【００６０】

なお、以上のようなＲＤＢ１０５での管理が非効率的であるＸＭＬ文書を、敢えて構造検索エンジンに格納するように変更する改善手法は、再帰構造を持つＸＭＬ文書以外でも適用可能である。例えば、階層の深いＸＭＬ文書を関係表に格納する場合は、多数の関係表を定義してその間の外部参照関係を規定することになるが、このような関係表に対して構造検索をかける場合は、外部参照関係の条件を全てＳＱＬ式に加えなくてはならない。このような条件は、ＲＤＢにおいては検索コストの高いジョイン操作として処理されるため効率が悪い。このような場合に対しても、図７（ｂ）のようなマッピング定義チューニング手法を適用することによって、検索効率を改善することが可能である。

【００６１】

階層の深いＸＭＬ文書のマッピング定義の改善には、構造検索エンジンを用いない別の手法もある。図８を用いてこれを説明する。構造検索式８０１は、ｉ要素、その子要素であるｊ要素、さらにその子要素であるｋ要素に関する条件を指定するクエリである。スキーマ間マッピング定義１０９を用いる場合には、この構造検索式は構造変換モジュール１０２によってＳＱＬ式８０３に変換されることになる。このＳＱＬ式には、二つのジョイン操作、“ｉ．ｉｄ＝ｊ．ｐｉｄ”、および“ｊ．ｉｄ＝ｋ．ｐｉｄ”の条件が含まれることになる。これに対し、関係表ｋ（２０３）を関係表ｋ（８０２）のように、関係表ｉ（２０１）のカラムａと関係表ｊ（２０２）のカラムｃの値もレコードに含むように更新することによって、同じ構造検索式を関係表ｋ（８０２）のみに対するクエリとして実行することが可能となる。

【００６２】

関係表スキーマ定義１１０、およびスキーマ間マッピング定義１０９は、それぞれ関係表スキーマ定義８０５、スキーマ間マッピング定義８０６に更新されることになる。スキーマ間マッピング定義８０６の１行目は、ｉ要素の属性ａの値を関係表ｋ（８０２）のカラムｉａにも格納することを表現している。６行目も同様である。以上のマッピング定義においては、構造検索式８０１は、構造検索式変換モジュール１０２によって、ＳＱＬ式８０４に変換される。該検索式はジョイン操作を含まないため検索コストが低い。

【００６３】

複数の構造検索式の効率化を目的とする場合は、全ての構造検索式のパスの和を取って、上記と同様のマッピング定義改善手法を適用することが可能である。例えば、以下の構造検索式全てに関して効率化を図る場合：

・　／ｘ／ｉ［@ａ＝“．．”］／／ｋ［@ａ＝“．．”］
・　／／ｊ［@ｃ＝“．．”］／ｋ［@ｂ＝“．．”］
・　／／ｉ［@ａ＝“．．”　ａｎｄ　@ｂ＝“．．”］／ｊ［@ｃ＝“．．”］／ｋ

ｉ要素の属性ａ，ｂ、およびｊ要素の属性ｃの値を含むように関係表ｋを更新する。

【００６４】

このようなマッピング変更は、関係表の正規化を崩すことにあたり、一つの値を複数のカラムで管理することになるため、データの更新時にはオーバヘッドとなる。マッピング定義チューニングモジュール１０４は、更新クエリの発行履歴も併せて参照し、参照系クエリと更新系クエリの発行頻度の兼ね合いに応じて、このマッピング定義改善手法を適用

するか否かを決定する。

【００６５】

なお、以上の説明で用いたスキーマ間マッピング定義の記法は実施例を限定するものではなく、同様の意味を表現し得る定義仕様であれば、どのような記法でも適用可能である。また、以上は、ＸＭＬ文書の管理方法として説明したが、本実施例における方法は、ＳＧＭＬ、ＨＴＭＬに代表されるタグ付き構造化文書一般の管理方法としても適用可能であることは自明である。

【図面の簡単な説明】

【００６６】

【図１】実施例の全体構成図である。

【図２】実施例のＸＭＬ文書のデータ変換機能に関する部分の構成図である。

【図３】実施例のクエリリライト機能に関する部分の構成図である。

【図４】実施例のクエリ実行機能に関する部分の構成図である。

【図５】実施例のクエリ実行機能に関する部分の構成図（続き）である。

【図６】実施例のスキーママッピング改善例を説明する図である。

【図７（ａ）】実施例のスキーママッピング改善例を説明する図（続き）である。

【図７（ｂ）】実施例のスキーママッピング改善例を説明する図（続き）である。

【図８】実施例のスキーママッピング改善例を説明する図（続き）である。

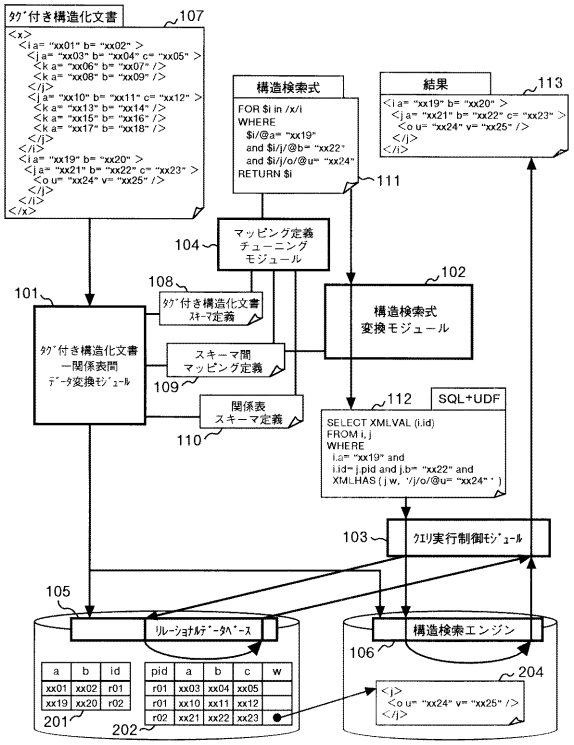
【符号の説明】

【００６７】

１０１．．．タグ付き構造化文書－関係表間データ変換モジュール，１０２．．．構造検索式変換モジュール，１０３．．．クエリ実行制御モジュール，１０４．．．マッピング定義チューニングモジュール，１０５．．．リレーショナルデータベース，１０６．．．構造検索エンジン，１０７／７０１．．．タグ付き構造化文書，１０８／７０２．．．タグ付き構造化文書スキーマ定義，１０９／６０４／７０３／７０９／８０６．．．スキーマ間マッピング定義，１１０／６０３／７０４／７１０／８０５．．．関係表スキーマ定義，１１１／７０７／８０１．．．構造検索式，１１２／６０２／７０８／７１３／８０３／８０４．．．リライト結果のクエリ，１１３．．．（構造検索式の）結果，２０１～２０３／６０１／７０５／７０６／８０２．．．関係表，２０４／７１１／７１２．．．（構造検索エンジンに対する部分ＸＭＬ文書の）格納イメージ

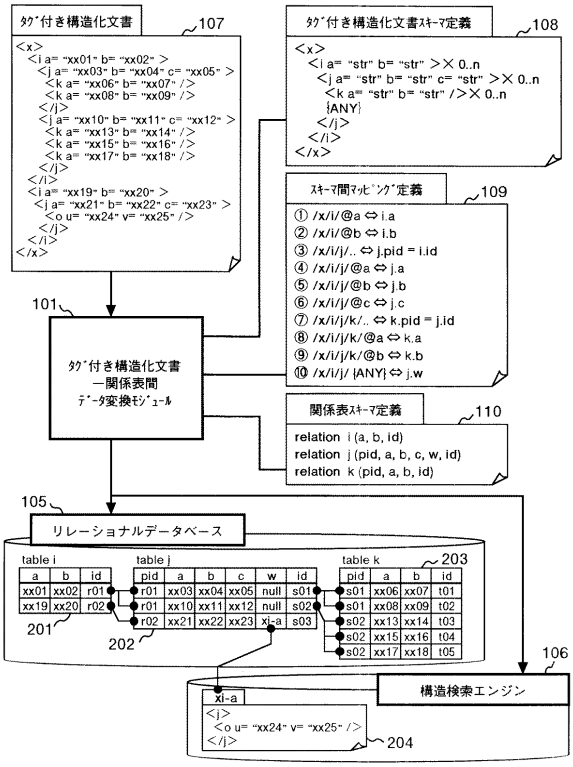
【図１】

図 1



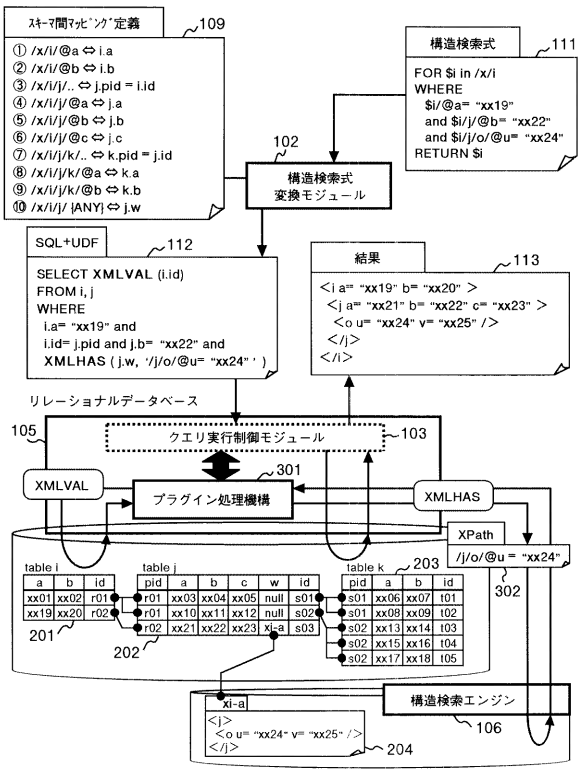
【図２】

図 2



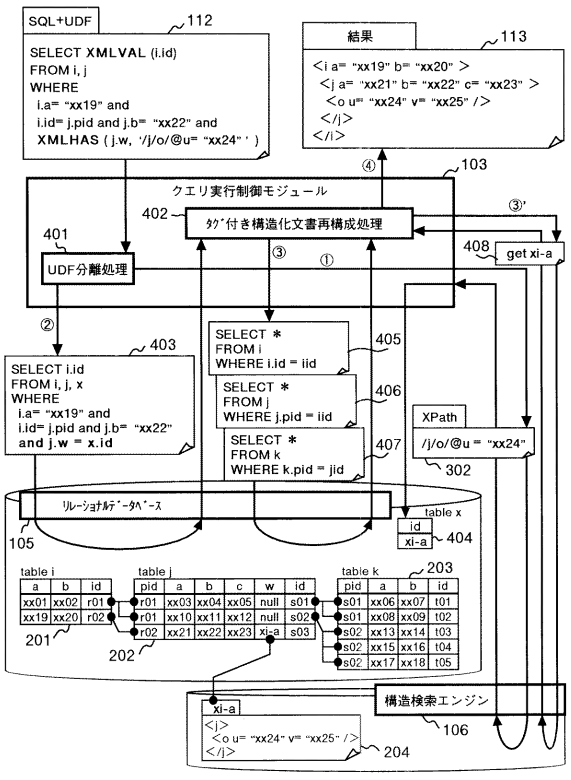
【図３】

図 3



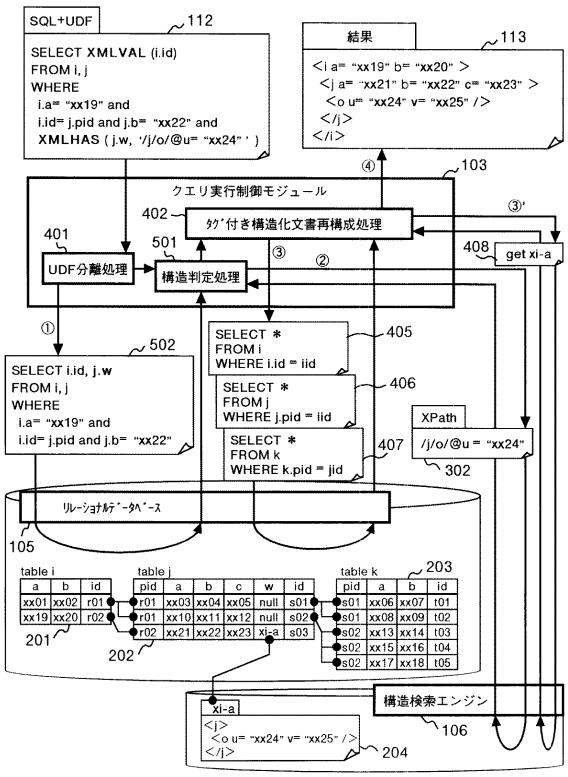
【図４】

図 4



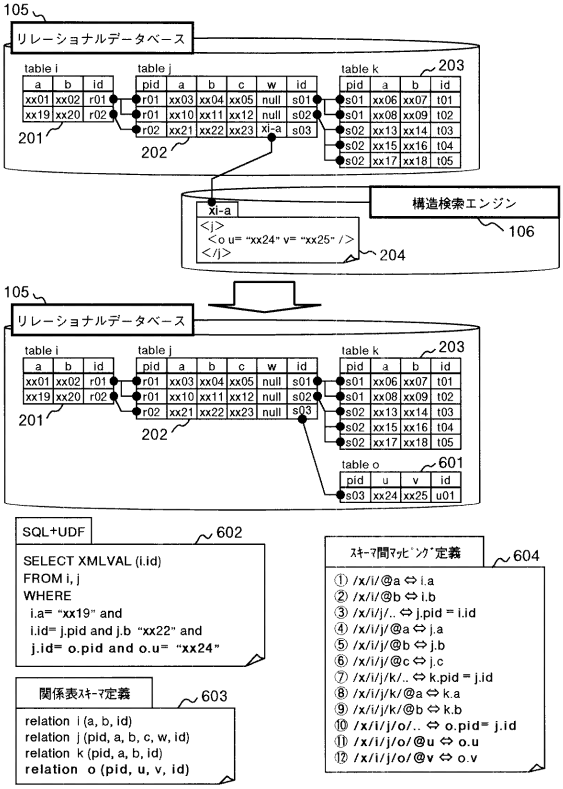
【図５】

図 5



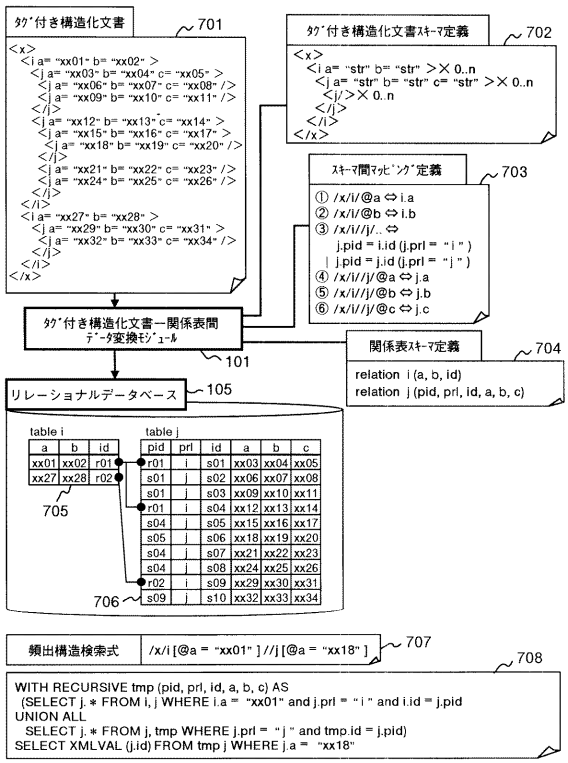
【図６】

図 6



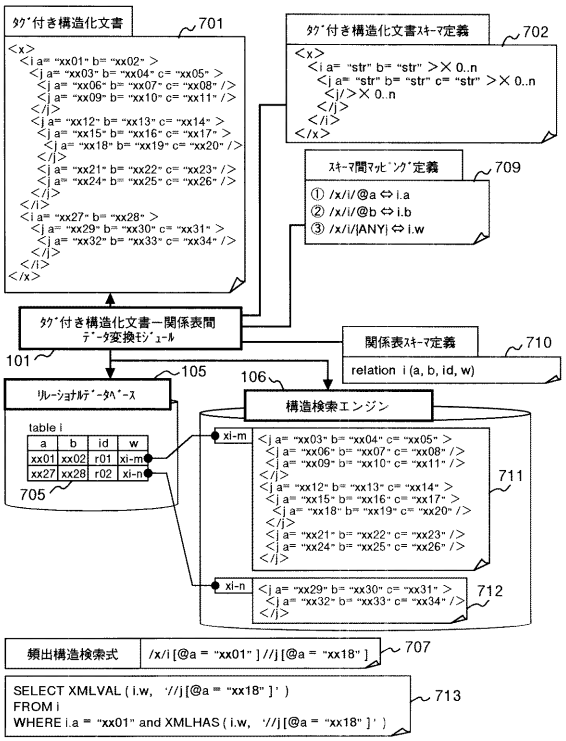
【図７（ａ）】

図 7（a）



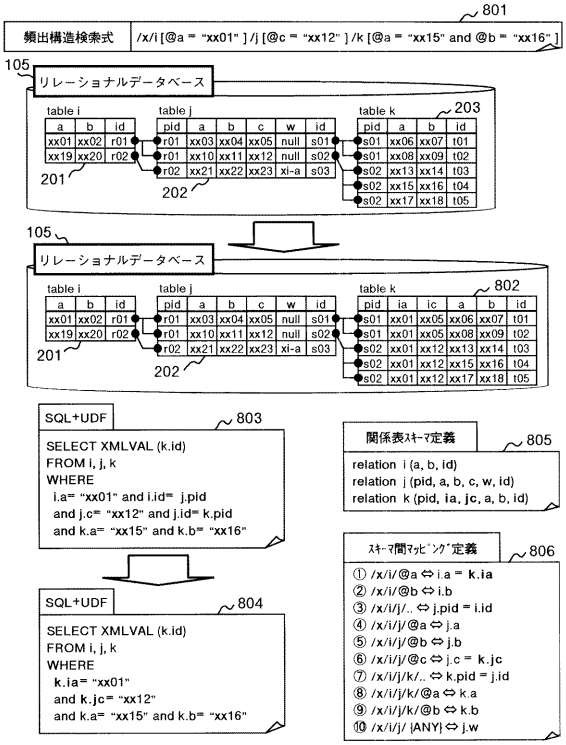
【図７（ｂ）】

図 7（b）



【図８】

図 8


801
頻出構造検索式
/x/i [@a = "xx01"] /j [@c = "xx12"] /k [@a = "xx15" and @b = "xx16"]
105
リレーショナルデータベース
table i
a b id
xx01 xx02 r01
xx19 xx20 r02
table j
pid a b c w id
r01 xx03 xx04 xx05 null s01
r01 xx10 xx11 xx12 null s02
r02 xx21 xx22 xx23 xi-a s03
table k
203
pid a b id
s01 xx06 xx07 t01
s01 xx08 xx09 t02
s02 xx13 xx14 t03
s02 xx15 xx16 t04
s02 xx17 xx18 t05
201
202
105
リレーショナルデータベース
802
table i
a b id
xx01 xx02 r01
xx19 xx20 r02
table j
pid a b c w id
r01 xx03 xx04 xx05 null s01
r01 xx10 xx11 xx12 null s02
r02 xx21 xx22 xx23 xi-a s03
table k
pid ia ic a b id
s01 xx01 xx05 xx06 xx07 t01
s01 xx01 xx05 xx08 xx09 t02
s02 xx01 xx12 xx13 xx14 t03
s02 xx01 xx12 xx15 xx16 t04
s02 xx01 xx12 xx17 xx18 t05
201
202
SQL+UDF
803
SELECT XMLVAL (k.id)
FROM i, j, k
WHERE
i.a= "xx01" and i.id= j.pid
and j.c= "xx12" and j.id= k.pid
and k.a= "xx15" and k.b= "xx16"
SQL+UDF
804
SELECT XMLVAL (k.id)
FROM i, j, k
WHERE
k.ia= "xx01"
and k.jc= "xx12"
and k.a= "xx15" and k.b= "xx16"
関係表ｽｷｰﾏ定義
805
relation i (a, b, id)
relation j (pid, a, b, c, w, id)
relation k (pid, ia, jc, a, b, id)
ｽｷｰﾏ間ﾏｯﾋﾟﾝｸﾞ定義
806
① /x/i/@a ⇔ i.a = k.ia
② /x/i/@b ⇔ i.b
③ /x/i/j/.. ⇔ j.pid = i.id
④ /x/i/j/@a ⇔ j.a
⑤ /x/i/j/@b ⇔ j.b
⑥ /x/i/j/@c ⇔ j.c = k.jc
⑦ /x/i/j/k/.. ⇔ k.pid = j.id
⑧ /x/i/j/k/@a ⇔ k.a
⑨ /x/i/j/k/@b ⇔ k.b
⑩ /x/i/j/ {ANY} ⇔ j.w

フロントページの続き

Ｆターム(参考) 5B075 ND35 PP23 QT06
5B082 CA11 FA11 GA03 GA06 GA07 GA08