5F－9

ファイルマップを特徴とした構造化テキストデータベースのデータ格納と質問言語 *

金子 邦彦，権藤 夏男，牧之内 顕文
九州大学 大学院システム情報科学研究科 †

1 はじめに

構造化テキストにおいても，テキストデータベース[2]は，重要な研究テーマとなっている．テキストデータベースの主な課題の１つは，質問言語の設計と実装である．構造化テキストの質問言語の実現には，「構造情報を持ったテキストをデータベース形式に変換する」[2]ことが必要である．つまり，ファイルとして格納された構造化テキストを走査(parse)し，構造情報を取り出すことである．その課題に対し，[2]では，*structuring schema* が提案されていた．この方式の基本的な問題点は，「データベース中に走査木(parse tree)を格納する」ため，「不必要なオブジェクトが多数作成されることになる」ことだが，[2]では，質問の最適化によりこの問題の解決を試みている．

我々はテキストを前もってデータベース形式に変換するのでなく，質問の実行時に必要なだけ変換するという方式「動的抽出法」（*dynamic extraction*）を提案する．本方式では，構造情報がデータベース中に格納されないので，その管理の手間を省くことができる．すなわち，（1）データベースサイズが大きいときや，（2）テキストの更新頻度が大きいときに有効であると考えられる．以下，本方式の実装について述べる．

2 テキスト格納法

本研究は，我々の研究室で開発されたデータベースサーバ「わかし」[3]および永続プログラミング言語「いなだ」の研究成果を基盤としている．「わかし」と「いなだ」は，並行処理制御，ログ，リカバリなどのデータベース機能を提供し，オブジェクトデータベース標準ODMG-93[1]のC＋＋バインディングに準拠している．

「わかし」の基本アイデアは，「データベースファイルにマップされた分散共有メモリ（図1）」である．すなわち，テキストファイルは分散共有メモリにマップされ，遠隔サイトに供給される．分散共有メモリとは、ネットワーク上の複数の計算機が１つのメモリ空間を共有したものである．「わかし」の実装では，ネットワーク上の各計算機に分散共有メモリの複製が作成され，互いの通信により各複製のメモリイメージの同一性が維持される．

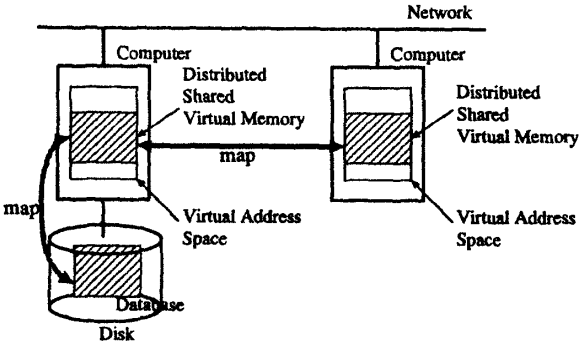

図 1: 分散共有メモリ

3 OQLによる質問

オブジェクトデータベース標準ODMG-93で規定された検索言語 Object Query Language(OQL)[1]と述語containの組み合わせにより，全文検索と構造検索との両方が可能である．例えば，OQLによる全文検索は次のように書くことができる．

```
select t
from t in Texts
where t.contains('database')
      and t.contains('text')
```

構造検索の例として，「タイトルに『datebase』と『text』を含むようなテキストを検索」する場合を考えると，この質問は，OQLと述語containにより次のように記述できる．

```
select t
from t in Reports
where t.title().contains('database')
      and t.title().contains('text')
```

*Data Storage and Query Processing for Structured Document Databases on File-Map Architecture, Kunihiko KANEKO, Akifumi MAKINOUCHI, Taiyong JIN

†Graduate School of Information Science and Electrical Engineering, Kyushu University, 6-10-1 Hakozaki, Higashi-Ku, Fukuoka 812-81, Japan

4 スキーマ定義

テキストのデータベース化のために，利用者は，テキストを扱うためのスキーマを定義する必要がある．我々のスキーマ定義方式の特徴は，（1）テキストは構造を持ったオブジェクトのように見せることができること，（2）すべてのテキストのベースクラスとなるTextクラスの導入により，利用者によるクラス定義の労力の軽減をねらっていることの2点である．Textクラスの定義は次の通り．

```
class Text {
   LargeObject* tp;
   d_Long offset;
   d_Long size;
   boolean contain( d_String keyword );
   d_List<Text>* parse( d_String tag );
}
```

ここで，メンバ変数 tp は，テキストファイルそのものへのポインタを表現する．メソッド contain は，与えられたキーワードがテキスト中に含まれるかを調べ，メソッドparse は，与えられたキーワードが存在する位置を調べる．

タイトル，著者，所属，あらすじ，章，節，参考文献の構成要素を持つような種類のテキストArticle のクラス定義は次の通りである．

```
class Article : public Text {
    Title* title();
    d_List<Author>* authors();
    d_List<Affiliation>* affiliations();
    Abstract* abstract();
    d_List<Section>* sections();
    Bibliography* bibliography();
};
```

ここで，Articleを構成するすべての要素はメソッドとして表現されることが特徴である．このことは，構成要素に関する情報がメソッド呼び出し時に取り出し可能なことを意味する．

テキストの構成要素も Textクラスのサブクラスとして定義される．例えば，「章」に対応するクラスの定義は次の通り．

```
class Section : public Text {
    Title* title();
    d_Lost<Body>* bodies();
    d_List<SubSection>* subsections();
}
```

「章」は，タイトル，本文，節からなるが，これら要素もArticleの場合と同様，メソッドとして表現される．

5 質問言語処理

Text クラスには構造情報を取り出すためのメソッドである'parse' が次のように定義されている．

```
d_List<Text>* parse( d_String tag_name );
```

構造の始まりは<タグ名> で，構造の終りは</ タグ名> で表現されているから，構造情報の取り出しは，タグの始めとタグの終わりでテキストを切り出すことで可能である．ここにメソッド parse の使用例を示す．

```
Title* Article::title()
{
 return (Title*)this->
  parse("title")->retrieve_first_element();
}

d_List<Affiliation>* affiliations()
{
 return (d_List<Affiliations>*)
  this->parse("title");
}
```

OQLでの質問は，前処理によりC++プログラムコードに変換される．例えば次の通り．

```
select s
from s in t.sections()
where s.contains('database')

d_Iterator<Section> iter
    = t.sections()->create_iterator();
Section s;
while( iter.next(s) ) {
   if ( s.contain( 'database' ) {
     query_result( s );
   }
}
```

6 まとめ

今回示した方式は，HTMLなどの構造化テキストのファイルを「ファイルマップ」によりそのままの形でデータベース化する．質問はOQLによって記述され，構造情報の抽出は質問処理時に行われる．

参考文献

[1] R.G.G.Cattel, "The Object Database Standard: ODMG-93 Release1.2," Morgam Kaufmann, 1996.

[2] Serge Abiteboul, Sophie Cluet, and Tova Milo, "Querying and Updating the File," Proceedings of the 19th VLDB Conference, pp. 73-84, 1994.

[3] G.Yu, K.Kaneko, G.Bai, A.Makinouchi, "Transaction Management for a Distributed Object Storage System WAKASHI – Design, Implementation and Performance," 12th Int'l Conf. on Data Engineering, 1996.