データベースシステム 113－49
（1997. 7. 16）

オブジェクトリンクを有する構造化文書に対する問合せ

加藤弘之 吉川正俊 植村俊亮

奈良先端科学技術大学院大学 情報科学研究科

本稿では，オブジェクト識別子や原子値のリテラルが対応付けられた，構造化文書の新しい論理モデルを提案する．この文書のことを “paratext” と呼ぶ．paratext は論理的に二つの平行な層から構成されているものとみなす．表の層には通常の文書 (すなわち，一次元文字列) が格納され，裏の層にはオブジェクト識別子やリテラルが表の層の文字列と対応付けられて保持されているものとする．この文書モデルに対して，定義域固有のいくつかの関数を設計した．これらの関数を用いることで，二つの層を自由に移動することができ，たとえば「記事の中に日本の企業に関する記述のある新聞記事」を検索する 問合せを記述することが可能となる．

Querying Structured Documents with Object Links

Hiroyuki KATO, Masatoshi YOSHIKAWA and Shunsuke UEMURA

Graduate School of Information Science
Nara Institute of Science and Technology (NAIST)

In this paper, a new logical model for structured documents enriched with OIDs and atomic value literals is presented. Such enriched text is named as “paratext”. Paratexts are logically viewed as consisting of two parallel layers. On the “upper” layer, ordinary text (i.e. a linear sequence of character strings) is placed, while the “lower” layer holds an array of OIDs and literals. Each OID or literal on the lower layer is associated with a contiguous substring of the upper layer text, and represents the semantics of the associated substring. We have also designed domain-specific functions for this document model. Using the functions, we can express queries which go back and forth between the two text layers. For example, a query language with those functions can express a query such as “Retrieve news articles which mention Japanese companies in its body.”

1. Introduction

By storing and managing text in database, the following tasks become possible: reuse of part of documents; and concurrency control and access control in simultaneous editing by multiple authors. Furthermore, with structured documents such as SGML, documents' logical structures can be used for the unit of above functionalities, also it becomes possible to retrieve documents based on logical structures and string pattern matching. For these reasons, research on document databases (especially on structured document databases) has come to be important [SDAMZ94][BYN96].

Another merit in managing documents in database is that by associating text data and the rest part of data in database systems (Throughout the paper, we will call data in databases other than text as 'legacy data'.) we can use them in integrated manner. That is, it becomes possible to retrieve text data based on legacy data and vice versa. Such association is a goal of several document database systems proposed so far[YA94][BCD+95]. However, in these systems, character string in text data and string data in legacy data is integrated based on string pattern matching. Therefore, documents in these systems can be retrieved based on pattern matching with strings of legacy data. Such retrieval functionality is essentially within those of conventional IR systems.

We need to develop deeply integration between text data and legacy data, based on not pattern matching with strings but semantics denoted by strings. This integration concretely is concretely that it is to construct links between strings, in documents, denoting objects or values stored in databases, and it is to couple strings, in documents denoting values not stored in databases but possible to be managed in databases, to literals in databases.

Utilizing this links and couplings for queries yields following advantages.

- It is possible to retrieve documents using semantics denoted by strings. One can find, for example, documents based on similarity between legacy data denoted by strings
- It is possible to utilize not only legacy data but also database services like functions and indices for specific types.

In this paper, we describe a conceptual database model for documents possible to be managed with additional information about links between arbitrary strings in documents and legacy data being managed by databases.

There are many variant to be able to model such links and couplings. We choose most simple model in conceptual level for it's easily handling. Our approach is that a ADT named *paratext* which have simple data structure with function having efficient expressive power is introduced into extensible databases. This ADT materialize a notion of "text of which some substrings have associated database objects or values" The notion of paratext is a generalization for of text in the natural manner.

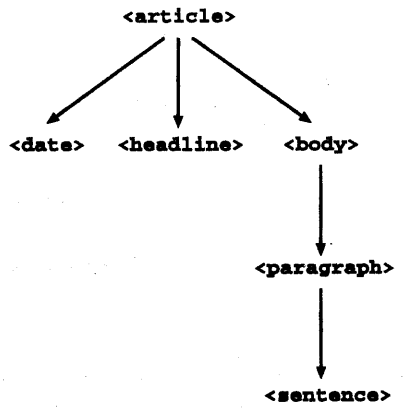

Figure 1 DTD for newspaper

In Section 2., we will give an illustrative example to show the usefulness of the notion of paratext. Section 3. describes the paratext ADT. Domain specific functions for this ADT is explained, and some example SQL queries using such functions are presented.

2. Paratext

In databases for structured documents, one can express queries based on documents' logical structure and character string pattern matching. For example, let us consider newspaper articles written in SGML conforming to the DTD[1] in Figure 1. An example SGML document instance is shown in Figure 2. (We are not concerned here with the character strings under underline. And, the numbers at left end of each line represent offset of the first character in the line.) In almost every structured document database sytem proposed so far, one can express queries like the following one:

Q1: *Find the headline of articles of which body contains 'WVE', and dated on September 26, 1996.*

To process this query, pattern matching is made between the content of **<date>** logical element in each document and the character string "September 26, 1996".

Users often wish to issue queries which have more complex conditions on date, rather than simply retrieving articles on a specific date. This is especially true in newspaper article databases. Examples of such complex queries include **Q2** and **Q3** below.

Q2: *Find the date and headline of articles of which body contains 'WVE', and dated on or*

[1]Following [YIU96], we use directed graphs to depict DTDs.

```
  1    <article>
 11    <date>
 18    Thursday, September 26, 1996
             DATE '1996-09-26'
 47    </date>
 55    <headline>
 66    WVE asks developers to write software for video board
        o1                                          o2
120    </headline>
132    <body>
              ...

360    <para>
367    WVE's OBQ video board uses the Ultra Video chip,
       o1          o2                  o3
416    an integrated circuit developed jointly with FastCircuit Inc.,
             o3                                      o4
479    a Japanese firm. The board will sell for less than U.S.$ 400.
            o4          o2
              ...

819    </para>
              ...

996    </body>
1004   </article>
```

Figure 2 A sample document.

within two weeks from September 26, 1996.

Q3: *Find the date and headline of articles of which body contains 'WVE', and dated on or within two weeks from the date of an article of which body contains both 'WVE' and 'FastCircuit'.*

Q2 retrieves articles in a period from a specific date, while in **Q3**, the beginning of periods is not fixed but is specified by conditions on another logical element.

In structured document database systems proposed so far, content of documents' logical elements is simply modeled as text (i.e. character strings). Hence, there are the following limitations:

i) Some complex queries concerned with the semantics of date become very complicated, and some others are impossible to express declaratively.

   For example, to express the query **Q2**, one need to specify a selection condition that sais the content of **<date>** element matches one of the fourteen character strings: "September 26, 1996", "September 27, 1996", ... and "October 9, 1996". Even worse, one cannot express the query **Q3** declaratively.

ii) Text indices on **<date>** element do not recognize subfields (i.e. year, month and day) of dates. Thus, processing of queries concerned with combinations of such subfields tend to be inefficient.

company

	name	country	emps
o1	World Video Electronics Ltd.	Canada	1250
o4	FastCircuit Inc.	Japan	327
	...	...	...

product

	name	company	year
o2	OBQ Video Board	o1	1985
o3	Ultra Video Chip	o4	1984
	...	...	...

Figure 3 Legacy data stored in relations.

Recent database systems support many rich data domains. For example, the semantics of contents of the logical element **<date>** is naturally modeled as **DATE** values in SQL-92[DD93]. If such semantics (i.e. a **DATE** value) of the content of each **<date>** element in text is available, database systems can easily utilize functions in **DATE** domain (e.g. arithmetic operations and **CURRENT_DATE**). Also, with adequate physical implementation of those **DATE** values, query processors can utilize indices on **DATE** values. These eliminate the limitations mentioned above.

DATE is a system built-in type. The same story applies to user-defined types. Note that company and product names appear in **<headline>** and **<body>** logical elements in Figure 2. Let us assume the database which stores this newspaper article documents also keeps legacy data on companies and products (for example, in relations shown

in Figure 3)[2]. By associating character strings in text (e.g. "WVE") and corresponding objects in legacy databases (e.g. o_1), the system can enjoy fully inegrated power of databases for structured documents and for legacy data.

For example, the integrated database can answer the following query:

Q4: *Find articles in which headline a company with more than 1000 employees is mentioned, and dated on September 26, 1996.*

In fact, similar queries are expressible also in other systems [YA94][BCD+95]. However, the integration of text data and legacy data in those systems relies on string pattern matching. Hence, in their systems, the processing of the query **Q4** first retrieves the names of companies with more than 1000 employees from legacy databases. Then, string pattern matching of those names is carried out against text in `<body>` elements. This approach cannot handle the situations where:

- different character strings represent the same real world entity; or
- different real world entities are represented by the same character strings.

For example, **Q4** cannot retrieve the newspaper article in Figure 2 under the approach of [YA94][BCD+95], because the character string (i.e. "WVE") in `<headline>` representing the company does not match with the company's name (i.e. "World Video Electronics Ltd.") stored in legacy database.

The concept, generalized above, of "text of which some sbstrings have associated database objects or values" is captured by the notion of *paratext*.

Figure 2 including the literals and the object identifiers under underlines using `DATE` type and user-defined relations in Figure 3 shows an example of paratext.

3. Basic Approach in Conceptual Level

In this section, we describe basic approach in conceptual level taking following points into consideration to satisfy the requirements mentioned previous section.

- It must be integrated between legacy data and paratext data, that is, both is managed in the "same" database.
- It is required, at conceptual level, the notion abstracted from "text of which some substrings have associated database objects or values".

We introduce new ADTs into extensible databases to handle paratext data. This approach has following advantages.

i) It is not necessary to change syntax of SQL.

ii) It keeps extensibility adding incrementally new ADTs into databases for another purpose.

[2]We believe this assumption is not unrealistic for wide variety of today's organizations including newspaper companies and manufacturers.

3.1 ADTs incorporated into extensible databases

We assume that *integer*, *string*, *float*, *boolean*, *date* types are built-in types in our extensible database.

3.1.1 preliminary

There are following ADTs introduced into our database to define the ADT to handle paratext data.

region type

We consider a document as a set of one's substrings. Each substring is defined by a pair of positions in the document corresponding to the beginning and end of the occurence of the substring. We call such substring *region*.
Now, following region type specific predicates are defined.

- `inclusion(region, region)`: holds iff. former region includes later region.
- `precedense(region, region)`: holds iff. former region precedes later region.

Each region is simply represented as a pair of integers (a, b) scht that $a \leq b$, where a is not always 1 origin.

text type

We introduce *text* type to manage character strings. The text type has following buit-in functions. There is a function that for a given text instance and a given substring it results a set of region such that each region over the text instance represents the substring. There is also a function that vice versa.

And, there are variable predicates, built in text type, to represent sophisticated pattern matching facilities for selecting documents according to their content relying on full text indexing, like IRS(Information Retrieval System) providing. For example, `contain(text,text)` holds iff. former text contain later text.

Indeed, There are many researches for integration between DBMS and IRS([ElM96]).

structured-text type

Structured-text type is, subtyping of text type, able to manage SGML documents. Literals of values of this type are represeneted as SGML documents. There is a buit-in function that for a given SGML document and a given element name, it results a set of corresponding region of the document.

There are variable researches for managing structured documents at DBMS. ([YA94, CDY95, VAB96]).

3.1.2 paratext type

We introduce *paratext* type to manage data such as "text of which some substrings have associated

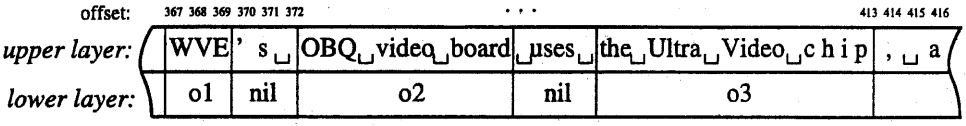

Figure 4 A sample paratext

database objects or values". Values of *paratext* type are, as mentioned above, logically viewed as consisting of two parallel layers. On the "upper" layer, a linear sequences of character strings with it's structure, that is value of *structured-text* type, is placed, while the "lower" layer holds an array of OIDs stored in advance and literals, each of which is associated with a contiguous substrings of the upper layer text. Figure 4 shows a sample value of paratext type.

Now, we give a formal definition to paratext data.
Definition1: For a given database instance D, a *paratext data* P is a 3-tuple(S, R, τ), such that

- S is a document(i.e. a linear sequence of character strings)
- R is a set of region (with no restriction on overlaps) over S.
- τ is a mapping from a region $r \in R$ to set of a d. The d is an element of a finite set $\mathbf{L}$ such that $\mathbf{L} = \mathbf{oid}_{fin} \cup \mathbf{lit}$, where $\mathbf{oid}_{fin}$ is a finite set of object identifiers occurring in D; $\mathbf{lit}$ is a finite set of literals possible to be managed in D.

There are following *paratext* type specific functions. A function to extract paratext data is, first of all, introduced.

- `set of paratext extract(paratext,arg2)`
  For a given paratext instance and a given `arg2`, this function results set of paratext instance partitioning by `arg2`. If `arg2` is associated with text type, this function extract a set of paratext, over a given paratext instance, such that "upper" layer of each of which holds `arg2`. If `arg2` is OID(s) or literal(s), this function results set of paratext instance such that `arg2` is placed at "lower" layer of each of which. And If `arg2` is a <element name>, this function results set of paratext such that "upper" layer of each of which is a corresponding element.

Next, we introduce functions return values, namely at "upper" or "lower" layer, consisting *paratext* instances.

- `text get_text(paratext)`
  For a given paratext instance, it returns a value of text type is placed at "upper" layer.
- `set of basic type get_ref(paratext)`
  For a given paratext instance, it results set of literals or OIDs at "lower" layer.
- `flatten(set)`
  To extract element of a set, like unnest operator at nested relation.

3.2 The language combined various domain specific functions/predicates

Figure 5 shows sample queries expressed by SQL style, mentioned previous section using natural languages. We assume that relation named *news* has a attribute named *article* which associated with a paratext instance conforms to DTD in Figure 1

4. Conclusion and Future Works

We present a novel way to manage structured document in databases. Now, we are implementing a prototype of our database model.

Acknowledgements

We wish thank members of Uemura-Lab at NAIST for many fruitful discussions during the early stages of this work.

References

[BCD+95] G. E. Blake, M. P. Consens, I. J. Davis, P. Kilpeläinen, E. Kuikka, P. -Å. Larson, T. Snider, and F. 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.

[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.

[CDY95] S. Chaudhuri, U. Dayal, and T. Yan. Join queries with external text sources: Execution and optimization techniques. In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 410–422, May 1995.

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

[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.

[SDAMZ94] Ron Sacks-Davis, Timothy Arnold-Moore, and Justin Zobel. Database systems for structured documents. In *Proc. of the International Symposium on Advanced*

Q1: *Find the headline of articles of which body contains 'WVE', and dated on September 26, 1996.*

```
SELECT  extract(article, <headline>)
FROM    news
WHERE   flatten(get_ref(flatten(extract(article, <date>)))) = DATE '1996-09-26'
AND     contain(get_text(flatten(extract(article, <body>))), "WVE")
```

Q2: *Find the date and headline of articles of which body contains 'WVE', and dated on or within two weeks from September 26, 1996.*

```
SELECT  extract(article, <date>), extract(article, <headline>)
FROM    news
WHERE   flatten(get_ref(flatten(extract(article, <date>)))) >= DATE '1996-09-26'
AND     flatten(get_ref(flatten(extract(article, <date>))))
                <= (DATE '1996-09-26' + INTERVAL '14' DAY)
AND     contain(get_text(flatten(extract(article, <body>))) 'WVE' )
```

Q3: *Find the date and headline of articles of which body contains 'WVE', and dated on or within two weeks from the date of an article of which body contains both 'WVE' and 'FastCircuit'.*

```
SELECT  extract(news2.article, <date>), extract(news2.article, <headline>)
FROM    news news1, news news2
WHERE   contain(get_text(flatten(extract(news1.article, <body>))), 'WVE')
AND     contain(get_text(flatten(extract(news1.article, <body>))), 'FastCircuit')
AND     flatten(get_ref(flatten(extract(news1.article, <date>))))
                <= flatten(get_ref(flatten(extract(news2.article, <date>))))
AND     flatten(get_ref(flatten(extract(news2.article, <date>))))
                <= flatten(get_ref(flatten(extract(news1.article, <date>))))
        + INTERVAL '14' WEEK
AND     contain(get_text(flatten(extract(news2.article, <body>))), 'WVE')
```

Q4: *Find articles in which headline a company with more than 1000 employees is mentioned, and dated on September 26, 1996.*

```
SELECT  a.article
FROM    a in news , b in company
WHERE   flatten(get_ref(flatten(extract(a.article, <date>)))) = DATE '1996-09-26'
AND     b.oid IN get_ref(extract(a.article, <headline>))
AND     b.emps <= 1000
```

Figure 5 Sample queries by SQL like expressions

Database Technologies and Their Integration, pp. 272–283, October 1994.

[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.

[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.