A Database System Integrating Structured Documents and Objects*

5 F－4　Hiroyuki KATO, Masatoshi YOSHIKAWA, Hiroko KINUTANI †
Graduate School of Information Science, Nara Institute of Science and Technology(NAIST) ‡

1. Introduction

By managing documents in databases, it is possible to associate character strings in documents with other data in databases. The association in most of such efforts is based on pattern matching with strings.

Our research focuses on semantics denoted by strings in documents, and provides link mechanism based on this semantics to develop deep integration between document data and other data in databases[4), 6), 5)]. The semantics denoted by strings is that i).data being managed by databases and; ii).data possible to be managed by databases. We will call, in this paper, such data as *legacy data*. By utilizing the link mechanism, there will be the following advantages.

- In addition to traditional pattern matching facilities provided by Information Retrieve Systems(IRS), it becomes possible to retrieve documents based on relationships among legacy data denoted by character strings.
- Furthermore, it becomes possible to utilize database facilities such as functions and indices of some types.

Considering the foregoing link mechanism, we will model documents as "text of which some substrings have associated legacy data" instead of "a linear sequence of character strings". Our approach is the introduction of a new ADT named *paratext* into extensible database. Paratext has a simple data structure with functions having efficient expressive power. The paratext has following features:

- The links are created over instances in databases.
- The links are created by authors of documents when documents are created.
- Legacy data and paratext data are integrated, that is, both are managed in the "same" database.

The integrated database proposed in this paper can answer, for example, the following query:

Q1: *Retrieve articles in which something related to Canada is mentioned, and their ranking scores.*

In section 2., we will describe the paratext ADT and other ADTs incorporated into our database. Section 3. illustrate the query language in our database. We will also sketch a novel sort *dizzy path*.

2. ADTs Incorporated into Database

We introduce new ADTs into extensible databases to handle paratext data.

2.1 preliminary

We assume that *integer*, *string*, *float*, *boolean*, *date* types are built-in types in our extensible database. There are following ADTs introduced into our database to define the ADT to handle paratext data.

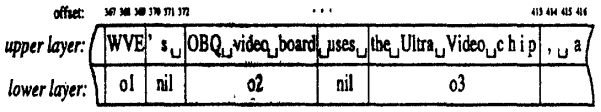

Figure 1 A sample paratext

top type

We introduce *top* type as root of type lattice in our databases to manage legacy data.

2.1.1 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 occurrence 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.

Using combined with these predicates, one can construct expressions with equivalent expressive power of region algebra[2)]. Each region is simply represented as a pair of integers (a, b) such that $a \leq b$, where a is not always 1 origin.

2.1.2 text type

We introduce *text* type to manage character strings. The text type has following built-in functions.

- *set of text* `extract(text, arg2)`
  For a given text data and `arg2`, it results set of text partitioning by `arg2`. If `arg2` is associated with text type, it extract a set of text, over a given text data such that, each of which is `arg2`. If `arg2` associated with region type, it extract text over given text partitioning by `arg2`.
- *text* `get_region(text)`
  For a given text data, it results region value corresponding to the beginning and end of the text over original text.

There are also variable predicates 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[1)].

2.1.3 structured-text type

Structured-text type is, subtyping of text type, able to manage SGML documents. Literals of values of this type are represented as SGML documents. There is a built-in function.

---

*構造化文書とオブジェクトの統合データベース管理システム

†加藤 弘之　吉川 正俊　網谷 弘子
‡奈良先端科学技術大学院大学 情報科学研究科

company

	name	country	emps
o1	World Video Electronics Ltd.	o9	1250
o4	FastCircuit Inc.	o8	327
	...	...	...

country

	name	capital	population
o8	Japan	Tokyo	120,000,000
o9	Canada	Ottawa	26,000,000
	...	...	...

Figure 2 Legacy data stored in relations.

- *set of structured-text*
  `extract(structured-text,<tag-name>)`
  For a given SGML document and a given tag name, it results a set of structured documents such that each of which is the corresponding tag name.

There are variable researches for managing structured documents in DBMS, for example [3].

2.2 paratext type

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

Definition 1: Given a database instance D, a *paratext data* P over D is a 3-tuple(S, R, τ), such that

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

Figure 1 shows a sample value of paratext type. There are following *paratext* type specific functions in addition to appropriate overloading functions defined in the super types mentioned above.

- *set of paratext* `extract(paratext, top)`
  For a given paratext instance and a given legacy data associated with top type, this function results set of paratext instance such that the legacy data is placed in "lower" layer of each of which.
- *text* `get_text(paratext)`
  For a given paratext instance, it returns a value of text type is placed at "upper" layer.
- *set of top* `get_ref(paratext)`
  For a given paratext instance, it results set of legacy data in "lower" layer.

2.3 structured-paratext type

A value of *structured-paratext type* is that for 3-tuple(S, R, τ), S is structured-text data. This type has appropriate overloading functions defined in the super types of the type.

There are some cases where data in "upper" layer of the result of `extract()` is no longer associated with structured-text type. Such data, then, migrates to be associated with paratext type. Figure ?? shows the class hierarchy about text data.

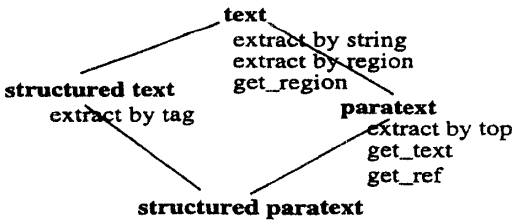

Figure 3 Class hierarchy

3. The Query Language by Eample

We assume that relation named *news* has a attribute named *article* which associated with a paratext instance, and that there are legacy data in Figure 2 . The sample query **Q1** expressed by SQL style is as follows:

```
SELECT  get_text(a.article), sim(o Dpath_p)
FROM    a in news, b in country
WHERE   o IN get_ref(a.article)
AND     b.name='Canada'
AND     o Dpath_p = b
```

In this query, the variable o plays the role of top type variable. The top type variable is range restricted coming from comparison(equality, membership or containment) with variables. The range restriction is useful for optimization of queries.

dizzy path(sketch)

We introduce a novel sort named *dizzy path*. A value of dizzy path denotes a path through complex objects/values with permitting inversion in the whole path. One can express similarity about semantics between objects/values using the dizzy path.

In above query, the `Dpath_p` is a variable of dizzy path. The function `sim()` computes ranking score based on similarity between begining and end tipped objects/values in dizzy path.

For the query described above, our database, thus, will answer articles contain the paratext data in Figure 1. Because, there are valuation from `Dpath_p` to `.country`, and from `o` to `o1`.

References

1) 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.

2) M.P. Consens et al. Algebras for querying text regions. In *Proc. ACM Symp. on Principles of Database Systems*, pp. 11–22, May 1995.

3) M. Volz et al. Applying a flexible oodbms-irs-coupling to structured document handling. In *Proc. of IEEE 12th Intl. Conf. on Data Engineering*, Feb.-Mar. 1996.

4) M. Yoshikawa et al. Amalgamating sgml documents and databases. In *Proc. of the 5th Intl. Conf. on Extending Database Technology (EDBT'96), LNCS, No. 1057, Springer-Verlag*,March 1996.

5) H. Kinutani et al. A cross-reference mechanism between database objects and document context for integrated document databases.(In Japanese) 55th Annual Conventions IPSJ, 5F-03, September 1997.

6) H. Kato et al. Querying structured documents with object links. IPSJ SIG Notes, 97-DBS-113. July 1997.