データベースシステム 120－4
（2000. 1. 24）

Mining from Semi-structured Data and Knowledge Integration

KOHEI MARUYAMA † and KUNIAKI UEHARA††

Despite the growing popularity of semi-structured data such as Web documents, most knowledge discovery research has focused on databases containing well structured data. In this paper, we try to find useful information from semi-structured data. In our approach, we begin by representing semi-structured data in a prototype-based approach, then detect the typical structure of object sets. Next, we apply the algorithm of mining association rules to structured layer by using the idea of concept hierarchy. Furthermore, relationships between concepts are defined and data values are not only generalized but also specialized for more flexible knowledge mining.

1. Introduction

World Wide Web has become a huge information storage that is growing rapidly. With the growth of such an on-line data, most of these data has become semi-structured. Therefore, researchers in the field of semi-structured data study the mechanism of manipulating and formulating a query on such data[3)4)6)]. However, most conventional data mining researchers have focused on generating rules within databases containing well-structured data, such as relational database and object-oriented database, where external schema is known in advance. Furthermore, few researchers concentrate on data mining techniques from semi-structured data, because discovering useful rules from collection of unstructured objects is a very challenging task.

That is, there must be more useful information to be discovered in a semi-structured world than in well structured world. Therefore, we try to mine rules from semi-structured data by discovering schema patterns of collection of objects and constructing a structured layer over unstructured objects. We also associate each object with each concept in a concept hierarchy and discover association rules using relationships between concepts. Concept hierarchy means taxonomy ($is-a$ hierarchy) which classifies the concepts into hierarchy based on their level of generality. Concept hierarchy makes us to find hidden informative rules behind the discovered rules which is meaningless at first sight. In dealing with these discovery task, we adopt prototype-oriented approach, which is proposed in the field of object-oriented programming. Prototype-based model is suitable for the idea of semi-structured data.

† Graduate School of Science and Technology, Kobe University
†† Research Center for Urban Safety and Security, Kobe University

2. Prototype-based Model for Semi-structured Data

2.1 Semi-structured data

Semi-structured data is the data that is neither raw data nor very strictly typed as in conventional relational database or object-oriented database. Its structure is irregular, implicit, or partial and the distinction between schema and data is blurred. Semi-structured data is also said to be a "self-describing" in that its structural information is not given in advance but is embeded in the data itself.

For example, bibliography data such as BibTeX files is considered to be a semi-structured data. Figure 1 shows an example of BibTeX file. Data in BibTeX files seems to be relational data, but its structure is not as regular. That is, it contains some degree of irregularity. For example, an attribute appeared in the object such as "**address**" may be missing in other object; some authors' names are recorded separately while others have a single full name or

```
@InProceedings{Object-ID,
  author =      "A.SMITH and M.TOM",
  title =       "XXX",
  booktitle =   "International Conference On Knowledge
                 Discovery and Data Mining",
  editor =      "AAA & BBB...",
  publisher =   "ACM Press, New York",
  address =     "Menlo Park, CA, USA",
  keyword =     "data mining, visualizing"
  month =       "aug",
  year =        "1996",
  pages =       "214--219",
  note =        "ftp://ftp...../**.ps)",
  id =          "ML75",
  list =        "MLO KDD",
}
```

Fig. 1 BibTeX Objects.

a initial name; "publisher" may have "place" attribute like "New York" in some objects; etc.

2.2 Prototype-based model

We adopt a prototype-based model[5] to manipulate semi-structured data. In Prototype-based model, there is no distinction between classes and instances nor between methods and data. New instance objects are created dynamically by copying prototype object. Created objects inherit all the nature from its prototype object (i.e. parent object) and can be modified at local. Furthermore, slots which store the attributes and data value can be added or removed at run time.

Types of data values (int, float, string etc.) are also evaluated dynamically. Figure 2 shows the graphical view of prototype-based model. "slot A" of "obj_2" has a label value "label_2", "slot B" has a method "method_2", and "slot C" has references to its subobjects "obj_21" and "obj_22." Each slot is evaluated dynamically and each prototype can have its own structure.

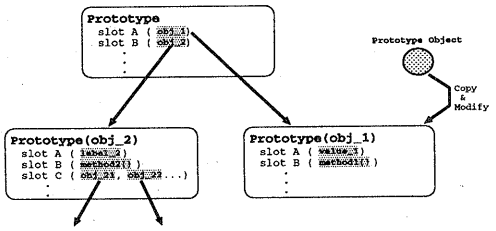

Fig. 2 Prototype-based Model.

Prototype-based model supports dynamic typing and provides flexible environment which is appropriate for modeling semi-structured data which is said to be "self-describing." Prototype-based model has some common characteristics with a conventional object-oriented model and OEM (*Object Exchange Model*)[3][4]. On the other hand, some of differences are as follows:

- **Comparison with OO model**
  - Conventional object-oriented model lacks the ability to support incremental and dynamic evaluation of schema and attributes.
  - Prototype-based model supports dynamic typing and dynamic inheritance, and each object can have a different structure. Such a nature is appropriate for semi-structured data which is said to be "self-describing."
- **Comparison with OEM model**
  - Conventional OEM model only supports object nesting and object identification, and other features such as classes, methods, and inheritance are neglected. This is because OEM is aimed to object integration from highly heterogeneous sources. It means OEM model is too simple to manipulate bibliographic objects whose type of data attributes is eclectic.
  - Prototype-based model is flexible enough and can support method and inheritance like object-oriented model.

In our prototype-based model, each attribute is treated as prototype and its slots have data values of arbitrary types or links to their sub-objects. Each prototype can have method in its slot such as getting its attribute value or name of its sub-object.

We modeled BibTeX objects by tree expression as shown in Figure 3. In OEM model, labels which denote relationships to other objects are attached to edges, but we attach such labels like title and year to nodes and reference between them to edges, which makes it easy to understand intuitively and easy to implement. Each attribute is modeled as prototype, and collection of attributes (i.e. &1 in Figure 3) is also modeled as prototype.

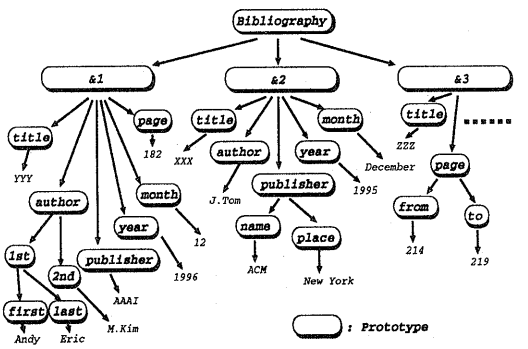

Fig. 3 Bibliography Objects Based on Tree Expression.

3. Discovering Schema Patterns

3.1 Schema Discovery for collection of semi-structured objects

In case of manipulating semi-structured data, we cannot often use standard database access method due to the lack of external schema informaiton. It means we cannot extract attributes used in generating association rules described in Section 4.2. To formulate meaningful queries and mine rules on semi-structured data, first of all, we need to discover the information of their structures. Such operation is referred to as schema discovery.

Schema discovery process is executed by moving from an object to its sub-objects and by keeping track of labels of the object reference in the case of single object[9]. This technique was extended to consider multiple objects like those in Figure 3[6]. We apply this algorithm to discovering typical structural pattern of a given collection of objects to our case.

In the BibTeX data, there are useless attributes which do not appear so frequently, such as "url" or "abstract" attributes. These attributes make mining task inefficient. By discovering schema patterns, these attributes can be pruned. We use these discovered schema patterns for building a structured layer over semi-structured objects to enable mining association rules from those objects.

A summary of the algorithm is as follows: Note that the original algorithm[6] is a little more complex because it deals with cyclic graph model but our model deals only acyclic graph.

- **Definitions**
  - *tree expression* : we regard schema as labeled tree representation of the object. $\perp$ is a *tree expression* of every object and te_i are tree expressions of objects id_i. Let p_i denotes *path expression* which is a path representation from root node to leaf node. A *$k-$tree expression* is a tree-expression containing k leaf nodes and can be represented by a sequence $p_1 ... p_k$.
  - $MINSUP$: Consider a tree-expression te. The *support* of te is the number of the root document d such that te is "weaker than" d. Intuitively, if all structural information of te_1 is found in te_2, te_1 is weaker than te_2. $MINSUP$ denotes user-specified minimum support and te is frequent if the *support* of te is not less than $MINSUP$. te is *maximally frequent* if te is frequent and is not weaker than other frequent tree-expressions. The discovery problem is to find all maximally frequent tree expressions.

For example in Figure 3, te_2 = { title : $\perp$, publisher : { place : $\perp$ } } is a tree expression of &2 but not a tree expression of &1. Therefore, if we assume that all objects are only &1 and &2, *support* of te_2 is 50%. If $MINSUP$ is not less than 50%, te_2 is considered to be frequent.

- **Algorithm**
  (1) $MINSUP$ is specified by the user.
  (2) For all frequent 1-tree expressions, F_1 are found in the form of pass-expressions.
  (3) Every frequent k-tree expressions $p_1 ... p_{k-1} p_k$ is constructed by two frequent $(k-1)$-tree expressions $p_1 .... p_{k-2} p_{k-1}$ and $p_1 .... p_{k-2} p_k$. We represent all frequent k-tree expressions $p_1 ... p_{k-1} p_k$ as F_k
  (4) The actual frequent k-tree expressions in F_k are found. This step prunes all non-maximally tree-expressions.

Figure 4 shows an example of step 3. In this example, frequent 3-tree expression $p_1 p_2 p_3$ is constructed from two frequent 2-tree expressions $p_1 p_2$ and $p_1 p_3$.

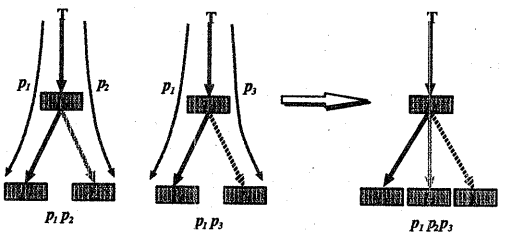

Fig. 4 Constructing $(k+1)$-frequent Tree Expression.

In the schema discovery phase, schema patterns which satisfy the specified number ($MINSUP$) of objects in all the objects are found.

Figure 5 shows the discovered pattern of BibTeX object set we got from Web. In this object set, *"author"* is composed from two elements:

"first" and *"second"*. *"editor"* is composed from three elements: *"first"*, *"second"* and *"third"*. However, in another object set, *"address"* may not appear and *"author"* may be composed from only one element, etc. If $MINSUP$ is specified more higher, the attributes like *"book title"* and *"city"* and *"state"* of *"address"* can be pruned. We consider this discovered schema pattern to be a structured layer.

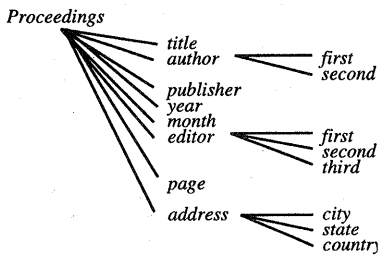

Fig. 5 Discovered Pattern (Structured Layer).

3.2 Knowledge Integration

Our purpose is to integrate the technique of schema discovery and mining association rules described in next section. It means the results of schema discovery phase affect the types of generated rules in association rule mining phase.

We show some examples. Figure 6 shows two schema patterns generated in schema discovery phase.

Pattern A shows an example of discovered schema pattern of bibliographies on technical reports. From this, we can find that there are attributes like "url" which means the existence of on-line information about the paper. Because style of technical reports are usually defined in advance, all objects have similar schema and the change of $MINSUP$ does not affect the discovered pattern dramatically.

In the case of pattern B which is a schema pattern of bibliographies on "spatial reasoning" we got from web, we can find that there are attributes like "journal" and "keywords" etc. So the generated rules are as follows:

- { journal = *"IEEE"*, year = *"1986"* } ⇒
  { keywords = *"geometric modeling"*,
  keywords = *"vision"*}

Unlike the technical report, most bibliographies have different structures individually. Therefore, the change of $MINSUP$ affect the

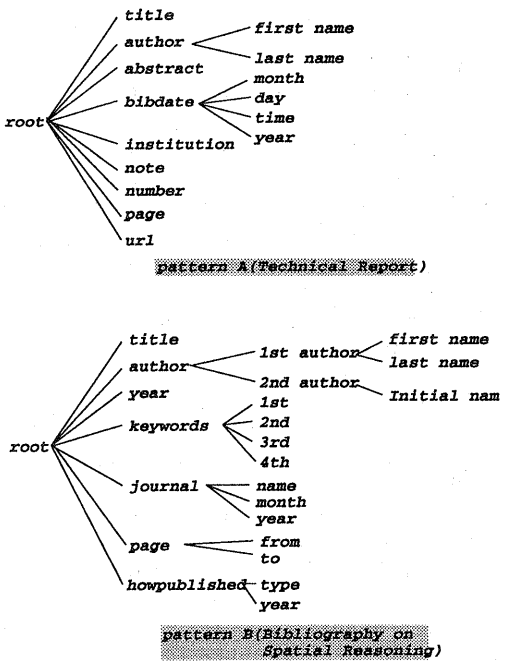

Fig. 6 Variety of Discovered Pattern.

discovered pattern and the generated rules. For example, if the attribute "journal" is not found in schema discovery phase, generated rules may be as follows:

- { year = *"1986"* } ⇒
  { keywords = *"geometric modeling"*,
  keywords = *"vision"*}

That is, items appeared in generated rules depend on the collection of bibliography objects we get. If we select another types of bibliography objects, resulting rules may be changed. Furthermore, generated rules also depend on $MINSUP$. Change of $MINSUP$ affect the items appeared in generated rules.

4. Mining with Concept Hierarchy

4.1 Prototype-Oriented Concept Hierarchy

Once a structured layer is built after schema discovering phase, we can apply algorithm of mining association rules to BibTex data by using discovered attibutes as item-set.

Association rules are powerful abstractions to understand a large amount of data by finding interesting regularities. However, the problem is that the number of discovered rules satisfying given thresholds of *support* and *confidence* are

often very large and contain many ambiguous rules.

Mining at a single concept level means that items in a rule are always concrete, and we cannot discover conceptually higher rules. To find more interesting and informative rules, it can be considered to use concept hierarchies as the representation of background knowledge.

For example, assume that a rule between *"author"* attribute and *"title"* attribute like *"A. Smith ⇒ association rule : 65%"* is found. Then we can know *"A. Smith"* wrote some papers about *"association rule."* But without knowledge about relationship among concepts like Figure 7, we cannot know that *"A. Smith"* wrote papers about *"data mining."* It means that we cannot generate rules relating with multiple concepts.

By integrating background knowledge with data mining algorithm, we are able to find more interesting rules and patterns. Some researchers have already proposed the use of a concept hierarchy during data mining process[10)11)], but in most of these cases, the concept hierarchy is in the form of a tree and attribute values exist only at the leaf node level of the tree. Moreover, these concept hierarchies are used to only generalize data value to a particular level in the hierarchy to represent resulting rules in more generalized form. For example, numerical value such as *"657-0024"* in *"zipcode"* attribute can be generalized to more abstract form by using a concept hierarchy, but cannot be specialized any more.

It is true that if data values of the database are numerical values or symbolic values, there is no need for specialization. However that is not case with the text value which contains some concepts itself. For example, text value such as *"association rule"* in *"title"* attribute can be generalized to its higher level *"data mining"* and be specialized to its lower level *"multi-level association rule."*

In the case of knowledge discovery from BibTeX data, the most useful attribute is *"title"* attribute. It contains important words which denote author's main interest. In our approach, each word is associated with a concept in a concept hierarchy.

We adopt prototype-based approach to construct the concept hierarchy. Our approach is different from conventional approaches as described below:

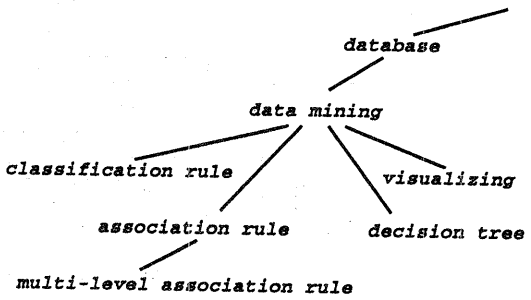

Fig. 7 Example of a Concept Hierarchy.

- Each concept is described as a prototype which is created by copying another concept object. When a new concept is created, we can define an inheritance relationship between the new concept and prototype concept. We distinguish three types of relationships: *Parent*, *Child*, and *Similar*. *Parent* means more general or broader concept, *Child* means more specific or narrower concept, and *Similar* means synonymous concept.
  For example in Figure 7, *"visualizing"* is a *Child* concept of *"data mining"* and *"data mining"* is *Parent* concept of *"visualizing."* *Similar* relationship is like a relationship between *"data mining"* and *"knowledge discovery."* Once these relationships are defined, the word *"knowledge discovery"* is regarded as the word *"data mining"* in discovering the association phase. This means we have no need for awareness of the words which have a similar meaning.

- A concept hierarchy might not be a tree structure. It can be an arbitrary graph structure and which may not have a unique root. As described above, conventional concept hierarchy has data values at a leaf node and each values are generalized only to mine multi-level association rules. But in our approach, each node in the concept hierarchy will also have data, and it can be both generalized and specialized.
  In our concept hierarchy, attributes are generalized or specialized based on neighboring relationship. It means resulting association rules are *Child* or *Parent* rules and there is no need to be concerned with over generalization. *Child* rule resembles a rule which contains descendant concepts in taxonomies ($is-a$ hierarchies)[2)]. But note

Table 1 Resulting Rules.

original rules	Conf
AUTHOR = *"A.Smith"*, YEAR = *"1998"* ⇒ TITLE = *"data mining"*	81.3%
AUTHOR = *"A.Smith"*, YEAR = *"1999"* ⇒ TITLE = *"data mining"*	85.7%
child rules	**Conf**
AUTHOR = *"A.Smith"*, YEAR = *"1998"* ⇒ TITLE = *"association rule"*	37.5%
AUTHOR = *"A.Smith"*, YEAR = *"1998"* ⇒ TITLE = *"visualizing"*	0.0%
AUTHOR = *"A.Smith"*, YEAR = *"1999"* ⇒ TITLE = *"association rule"*	0.0%
AUTHOR = *"A.Smith"*, YEAR = *"1999"* ⇒ TITLE = *"visualizing"*	28.6%

that we consider only *Child* concepts and not all descendant concepts are included in generating association rules.

- Prototype-based model supports dynamic evaluation of a concept hierarchy. That is to say, the new item can be included into the concept hierarchy and system can detect them at run time. For example, if a user wants to add new item into a concept hierarchy, he can add it by selecting a prototype item which has close meaning to it and copying it with relationship definition (*Parent*, *Child* or *Similar*).

4.2 Mining Association Rules

Association rule has the form

$$X \Rightarrow Y \ (X, Y \subset Itemset, X \cap Y = \emptyset)$$

It satisfies two given threshold values. The *support* is defined as the probability that a transaction contains both X and Y, and *confidence* is the probability that any transaction containing X also contains Y, so defined as

$$confidence(X \Rightarrow Y) = \frac{support\ (X \cup Y)}{support(X)}$$

An efficient algorithm of generating association rules is reported by Agrawal et. al[1]. However, because we are interested in using relations between concepts, their algorithm for mining association rules is modified as follows:

(1) By using a concept hierarchy, *Parent* concept of each item is identified.
(2) Find all item-sets which satisfy the user specified minimum support. These itemsets are called large item-sets. First of all, we find all large 1-item-sets, L_1. *Parent* concepts are also concerned at each step.
(3) In $k \geq 2$, the candidate itemsets of size k are generated from large $(k-1)$-itemsets, L_{k-1}.
(4) Search the transaction database, and compute the support of candidate itemsets.
(5) Large k-item-sets L_k, which satisfies the minimum support, is found.
(6) For each large item-set, find all the rules which have greater than a specified minimum confidence.
(7) Based on the relationships between concepts, *Child* rules for resulting original rules are generated. *Child* rules have the same items as original rules.

We call rules that are discovered without using a concept hierarchy as original rules. We can generate not only original rules but also their *Parent* and *Child* rules in this approach. For example, assume that following rules are found.

- { AUTHOR = *"A.Smith"*, YEAR = *"1998"* } ⇒ { TITLE = *"data mining"* }
- { AUTHOR = *"A.Smith"*, YEAR = *"1999"* } ⇒ { TITLE = *"data mining"* }

Of course, because concepts like *"decision tree,"* *"visualizing,"* etc. are defined as children of *"data mining,"* these conceps are generalized and such rules can be found. From these rules, we can know that *"A. Smith"* wrote papers mainly about the field of *"data mining"* recently. However, we cannot discover any knowledge from these rules. But using our concept hierarchy described above, we can discover other rules as follows: For example, we generate *Child* rules from discovered two original rules as shown in Table 1.

Generated rules show that in *"1998"* *"A. Smith"* wrote some papers about *"association rule"* on the data mining field, but in *"1999"* he wrote about *"visualizing"* several times while no papers were written about *"association rule."* This means that his interest has been changed from *"association rule"* to *"visualizing"* recently. Such knowledge can not be discovered by original rules alone.

As shown in Table 1, by using the relationship

between concepts, we can generate more informative rules for discovered original rules. For example, we may find the important keywords which are frequently appeard recently but not discovered because of the largeness of the itemsets we choose or of the height of *support* or *confidence*. One of our purpose is to find these hidden informative rules behind the discovered rules which is meaningless at first sight. It is often said that an algorithm of mining association rules may find many uninteresting rules. However, it is possible enough that many informative knowledges is hidden under these uninteresting rules.

5. Related Works

Some researchers use concept hierarchy in generating multi-level association rules[10)11)]. But in these cases, the concept hierarchy is used to only generalize association rules. In addition, these researchers do not assume mining rules from semi-structured data sets. We adopt the concept hierarchy which has an arbitrary graph structure and use it for not only generalizing the association rules.

Some reseachers attempt to discover knowledge using semi-structured data, but most of them regard doncument collection as semi-structured or their purpose is to classify semi-structured data.

L. Singh et. al[7)8)] have attempted to discover knowledge from semi-structured data. However, in their approach, semi-structured data means unstructured components of the text documents and stored separately from structured components. Our approach differs from them since we regard semi-structured data as data which has its own schema rather than text documents. Therefore, we had to begin with discovering their schema patterns to manipulate semi-structured data. It means that our approach has the capability of extension use to other semi-structured data such as XML whose DTDs have different structure depending on users.

6. Conclusion and Future Works

We have presented a framework for mining association rules from semi-structured data, based on a prototype-based concept hierarchy. The main contribution of this paper are as follows: First, we modeled the semi-structured data in a prototype-based approach. This approach provides a dynamic environment which is approproate for a semi-structured world. Second, we applied a schema discovering technique for collection of semi-structured objects. Discovered schema patterns are useful for mining task. Finally, we construct a concept hierarchy and define the relationsip between them. By integrating resulting association rules and concept hierarchy, more useful knowledge was found.

To discover association rules, we queried on web database of bibliography and got data in which we are interested. However, these object sets are not so large. We have to test our approach for larger datasets to discover various and useful association rules.

In addition, the efficiency which is important problem in the field of mining association rules is not considered in this paper. We have to examine this problem by applying our approach to larger datasets, and if necessary, it has to be improved.

We believe that there still remains a lot of works to be done in a semi-structured world, especially from knowledge discovery and the data mining point of view. Our work on semi-structured association rule mining must be continued.

7. Acknowledgement

This work was partially supported by the grant-in-aid for scientific research on priority area "Discovery Science" from the Japanese Ministry of Education, Science, Sports and Culture.

References

1) R. Agrawal and R. Srikant. "Fast Algorithms for Mining Association Rules," Proc. of 20th International Conference of Very Large Databases, pp.487-499 (1994).

2) R. Srikant and R. Agrawal. "Mining Generalized Association Rules," Proc. of 21st International Conference of Very Large Databases, pp.407-419 (1995).

3) Y. Papakonstantinou, H. Garcia-Molina and J. Widom. "Object Exchange Across Heterogeneous Information Sources," In Proc. of 11th International Conference on Data Engineering, pp.251-260 (1995).

4) S. Abiteboul. "Querying Semi-Structured Data," Proc. of 6th International Conference

on Data Engineering, Lecture Notes in Computer Science, Vol.1186, pp.1-18, Springer-Verlag (1997).
5) G. Blascheck, "Object-Oriented Programming with Prototypes," Springer-Verlag (1994).
6) K. Wang and H. Liu. "Discovering Typical Structures of Documents: A Road Map Approach," Proc. of 21st Annual International ACM SIGIR Conference on Research and Development in Information, pp.146-154 (1998).
7) L. Singh, P. Scheuermann and B. Chen. "Generating Association Rules from Semi-Structured Documents Using an Extended Concept Hierarchy," Proc. of 6th International Conference on Information and Knowledge Management, pp.193-200 (1997).
8) L. Singh, B. Chen, R. Haight, P. Scheuermann and K. Aoki. "A Robust System Architecture for Mining Semi-Structured Data," Proc. of 4th International Conference on Knowledge Discovery and Data Mining, pp.329-333 (1998).
9) S. Nestorov, J. Ullman, J. Wiener and S. Chawathe. "Representative Objects: Concise Representations of Semistructured Hierarcal Data," Proc. of 13th International Conference on Data Engineering, pp.79-90 (1997).
10) S. Fortin, L. Liu and R. Goebel. "Multi-Level Association Rule Mining: An Object-Oriented Approach Based on Dynamic Hierarchies," Technical Report TR 96-15, Dept. of Computing Science, University of Alberta (1996).
11) J. Han, A. Nishio, H. Kawano and W. Wang. "Generalization-based Data Mining in Object-oriented Databases Using an Object Cube Model," Data and Knowledge Engineering, Vol.25, Nos.1-2, pp.55-97 (1998).
12) K. Maruyama and K. Uehara. "Mining Association Rules from Semi-structured Data," Proc. of 20th ICDCS International Workshop of Knowledge Discovery and Data Mining in the World-Wide Web (to appear).