Efficient Storage for Semi-Structured Microarray Data
Exploiting Structural Similarity

Jongil Jeong¹, Dongkyoo Shin²∗ and Dongil Shin²

¹Industry Development Division, Korea Internet Security Agency
Daedong B/D, Garak-Dong 78, Songpa-Gu, Seoul 138-803, Korea
jijeong@kisa.or.kr

²Department of Computer Engineering, Sejong University
98, Gunja-Dong, Gwangjin-Gu, Seoul 143-747, Korea
{shindk, dshin}@sejong.ac.kr

Abstract. In this paper, we propose an alternative mapping strategy to mine the structural similarity and an advanced mapping rule from the algorithm. Object-relational mapping technique is used for extracting advanced storage design schema for microarray data and structural similarity of elements is evaluated for efficient storage construction. The mapping strategy reduced the number of relational tables remarkably.

Keywords: structural similarity, decision tree, microarray database, XML, schema mining

1 Introduction

Even though public repositories have been constructed successfully, enough consideration has not been given to the design of storage structure for the hierarchy of microarray data. For the purpose of managing semi-structured data, commercial RDBMS (Relational DataBase Management System) provides the object-relational mapping methods [3, 6, 7]. Commonly, these methods treat XML documents as a tree form of objects and transform it into several relational tables. This makes queries to XML data set have a number of joins. As a result, the performance of a database adopting this method is potentially deteriorated [3]. To overcome this problem, several studies have proposed alternative XML storage mapping techniques: DTD-dependent [3], Edge-dependent [4], and data mining-dependent [5]. The research [4] stores graph and edges in a single Edge table to handle all graph and edge information of XML data. The research [5] proposed STORED system adopted a data-mining algorithm [8] to extract relations from XML data and then transform it to relational database schema. In managing the structural variety of an XML instance, simplifying DTD would be the fundamental measure. The research [3] proposed in-line technique focusing on simplifying DTD prior to object-relational mapping. This technique eliminated omissible elements from DTD.

* Corresponding author
This research is supported by Seoul R&BD Program (SS110008).
Many MGED (Microarray Gene Expression Data) standard-compliant repositories have adopted RDBMS, which enables the repositories to take advantage of the RDBMS engine for its stability, convenient management, and good performance [1, 2]. The goal of the research is to implement storage following the MGED standard.

2. Design of Decision Tree for Establishing Classification Rules

Every element has a complexType that defines components such as sub-element(s) and attribute(s). Each component can be expressed as a part of a set with properties. Assuming that a certain element \( x \) has a complexType, each complement set of \( x \) can be defined as follows:

\[
SE_x = \{e_1, e_2, \ldots, e_n\}, \quad SA_x = \{A_{e_1}, A_{e_2}, \ldots, A_{e_n}\}
\]

\[
\text{complexType}(x) = (SE_x, SA_x)
\]

To help to understand the above summary, we defined some terminologies as follows:

- \( e \): an element defined in XML schema
- \( E \): an elements set of \( e \)
- \( SE \): a sub-elements set of \( e \)
- \( a \): an attribute of \( e \)
- \( A \): an attributes set of \( e \)
- \( SA \): an attributes set for all sub-elements of \( e \)
- \( \text{complexType} \): Structural information that consists of \( SE \) and (or) \( A \) of \( e \).
- \( \text{Lowest Child} \): an element without any sub-element.
- \( \text{LCG (The Lowest Child Group)} \): a set of lowest child elements
- \( \text{PG (Parent Group)} \): a set of elements with sub-element.
- \( \text{Root} \): an element without parent element.

Whether one complexType is similar to another complexType will depend on the comparison of sequence and attribute set between two complexType. For example, when \( SE_x \) and (or) \( SA_x \) in the complexType of a certain element \( x \) exactly match \( SE_y \) and (or) \( SA_y \) in the complexType of another element \( y \), \( x \) and \( y \) have the structural similarity.

---

**Fig. 1.** The decision tree from the training set of the XML schema for example
Figure 1 shows the decision tree shaping idea for classifying elements with structural similarity. There are two branching points in the decision tree. As an individual set of similar elements, each branching point has a condition by which an element is classified into a specific target node.

From each of branching points in the decision tree, we extracted rules for classifying elements with the structural similarity. Each of rules is expressed into IF-THEN statements respectively. Table 2 presents IF-THEN statements at each node.

Table 1. The rules for classifying LCGs, PGs, and root element.

<table>
<thead>
<tr>
<th>No</th>
<th>If</th>
<th>Then</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>hasChild=yes</td>
<td>Go to node No4</td>
</tr>
<tr>
<td>3</td>
<td>hasChild=no</td>
<td>LCGs</td>
</tr>
<tr>
<td>4</td>
<td>hasChild=yes AND hasParent=yes</td>
<td>PGs</td>
</tr>
<tr>
<td>5</td>
<td>hasChild = yes AND hasParent = no</td>
<td>root</td>
</tr>
</tbody>
</table>

Note that Node No3 and 4 are leaf nodes, which are sets of elements classified by the conditions at No1 and 2, respectively. Each set of elements might be separated into several groups for parent elements and lowest child elements with structural similarity. Thus, rules are required to evaluate the structural similarity of elements in No3 and No4.

References