DOI: 10.3745/KIPSTD.2009.16D.5.643

마이크로어레이 데이터의 구조적 유사성을 이용한 효율적인 저장 구조의 설계

윤 종 한[†] · 신 동 규[††] · 신 동 일[†††]

요 약

생명정보 대량 획득기술의 하나인 마이크로어레이(microarray)는 DNA와 각종 유전자 연구에 사용되는 도구로 확립되면서, 생명정보학(Bioinformatics)분야의 발전에 크게 기여하였다. 그러나 마이크로어레이는 생명정보학분야의 핵심기술 중 하나로 발전하였음에도 불구하고 실험으로 생성되는 데이터는 형태가 다양하고 매우 복잡한 형태를 갖기 때문에 데이터의 공유나 저장에서 많은 어려움을 겪고 있다. 본 논문에서는 마이크로어레이 데이터의 관리를 원활하게 하기위한 XML 기반의 표준 포맷인 MAGE-ML스키마에서 구조적으로 유사한 엘리먼트가 반복적으로 나타나는 특징과 대다수의 엘리먼트들이 특정 엘리먼트의 자식으로만 온다는 구조적 특징을 이용하여, MAGE-ML의 스키마를 단순화하고 저장구조를 효율적으로 설계하는 방법을 제안한다. 이 방법에서 인라인 기법(Inlining Technique)을 이용한 스키마의 단순화와 새롭게 제시하는 엘리먼트의 구조적 형태를 기준으로 분류하는 기법을 이용한다. 이를 통하여 데이터베이스 스키마는 간략화 되며 테이블조인의 횟수가 줄어들고 성능은 향상된다.

키워드 : 구조적 유사성, MAGE-ML, 인라인 기법, 반구조적 데이터, 마이크로어레이 데이터, 생명정보학

Design of Efficient Storage Exploiting Structural Similarity in Microarray Data

Jonghan Yun[†] · Dongkyu Shin[††] · Dongil Shin[†††]

ABSTRACT

As one of typical techniques for acquiring bio-information, microarray has contributed greatly to development of bioinformatics. Although it is established as a core technology in bioinformatics, it has difficulty in sharing and storing data because data from experiments has huge and complex type. In this paper, we propose a new method which uses the feature that microarray data format in MAGE-ML, a standard format for exchanging data, has frequent structurally similar patterns. This method constructs compact database by simplifying MAGE-ML schema. In this method, Inlining techniques and newly proposed classification techniques using structural similarity of elements are used. The structure of database becomes simpler and number of table-joins is reduced, performance is enhanced using this method.

Keywords : Structural Similarity, MAGE-ML, Inlining Technique, Semi-Structured Data, Microarray Data, Bioinformatics

1. 서 론

생명정보학(Bioinformatics)은 정보학적 관점과 방법론을 통해 생물학을 이해하고 연구하는 과학의 한 분야다. 오늘날 생명공학의 비약적인 발전과 함께 그에 수반되는 데이터 또한 방대해지고 복잡해졌다. 따라서 그 데이터를 다루기 위해 정보학과의 접목이 불가피해지고, 정보학의 역할도 커진 것이 현실이다. 생명공학에서 다루는 데이터를 더욱 체계적이고 편리하게 다루기 위하여 많은 데이터 포맷이 등장하였고, 그 데이터들을 다루는 것이 중요한 역할을 하게 되었다. 그 중에서 마이크로어레이(microarray)는 DNA와 각종 유전자 연구에 사용되는 도구로서 확립되면서, 생명정보학의 발전에 크게 기여하였다[1]. 하지만 마이크로어레이 데이터는 그 형태가 너무 다양하고 복잡한 형태를 갖는다. 그리고 데이터의 양 또한 너무나 방대하기 때문에 관리와 공유에 많은 어려움이 있다. 그래서 MGED(Microarray and

※ 본 연구는 보건복지가족부 보건의료기술진흥사업의 지원에 의하여 지원받았음(과제고유번호: A040163).
† 준 회 원 : 세종대학교 컴퓨터공학과 석사과정
†† 종신회원 : 세종대학교 컴퓨터공학과 교수
††† 종신회원 : 세종대학교 컴퓨터공학과 부교수
논문접수 : 2007년 11월 27일
수 정 일 : 1차 2008년 8월 6일, 2차 2009년 6월 3일, 3차 2009년 6월 22일
심사완료 : 2009년 6월 22일

Gene Express Data)표준화 그룹에서는 마이크로어레이 데이터의 표현에 있어서 표준을 확립하기 위한 노력을 하였다. 그 결과로 마이크로어레이 데이터의 표준 객체모델인 MAGE-OM과 MAGE-OM을 XML의 형태로 표현한 MAGE-ML[2]이 발표되었다. 이것은 상호간의 원활한 데이터 교환을 가능하게 해준다. 그럼에도 불구하고 여전히 그 복잡한 형태와 방대한양 때문에 효율적인 저장과 관리에 어려움을 겪고 있다.

본 논문에서는 MAGE 표준을 따르는 XML기반의 마이크로어레이 데이터의 효과적인 저장 구조에 초점을 맞춘다. 그리고 저장구조를 설계하는 과정에서 핵심이 되는 전략은 기존의 인라인 기법(Inlining Technique)[3]을 이용하여 MAGE-ML의 스키마를 단순화 하고, 이 단순화된 스키마에 정의된 엘리먼트의 구조적 유사성에 따른 분류를 통하여 스키마를 더욱 단순화하는 것이다. 그리고 본문의 핵심인 구조적 유사성에 따른 엘리먼트 분류 과정에서는 의사결정나무를 사용한 알고리즘이 사용된다. 이렇게 단순화된 스키마의 구조는 MAGE-ML 데이터의 저장 시 테이블의 생성과 테이블의 조인연산의 횟수를 줄인다. 그리고 조인연산의 감소와 데이터베이스의 공간복잡도의 감소로 데이터베이스의 성능은 향상된다.

본 논문은 총 6개의 장으로 구성된다. 2장에서는 관련연구, MAGE-ML, 마이크로어레이에 대하여, 3장에서는 인라인 기법을 MAGE-ML에 적용하는 과정을 보이고 본 논문에서 새롭게 제시하는 구조적 유사성 기반의 엘리먼트 분류 기법을 보인다. 그리고 이 기법을 이용하여 MAGE-ML 데이터를 RDBMS에 저장하는 과정을 보인다. 그리고 4장에서는 이렇게 저장된 데이터를 다시 MAGE-ML 문서로 복원하는 과정을 설명한다. 그리고 5장에서는 설계한 데이터베이스의 성능을 평가한다. 그리고 마지막 결론을 내린다.

2. 관련 연구

MGED(Microarray Gene Expression Data) 표준을 따르는 대부분의 저장소는 안정성, 편리한 관리, 그리고 뛰어난 성능을 보이는 RDBMS를 채택하였다 [4, 5]. 연구[4]에서는 객체 모델의 로컬 변화를 통하여 일반 쿼리(query)의 성능을 개선시켰지만, 그러한 변화를 통해 효과적인 데이터베이스 설계 스키마를 기대하긴 어려웠다. 이와 비슷하게 마이크로어레이 데이터베이스에 대한 연구 The Stanford Microarray Database[5]가 있었지만, 역시 명확한 해결책을 보이지는 않았다. 한편, 연구[6]에서는 MAGE(Microarray Gene Expression)객체를 XML 문서로 변환하는 구체적인 매핑 규칙을 발표 하였지만, 그것은 마이크로어레이 데이터를 위한 관계형 데이터베이스의 효과적인 설계를 위한 것이 아니라 단지 관계형 테이블로부터 XML문서를 가져오는데 초점을 맞추었을 뿐이었다.

한편, 반구조적 데이터(semi-structured data)[7]인 XML 문서를 RDBMS(Relational Database Management System)에 저장하는 방식에 대한 연구들이 이루어져왔다[8-10]. 이 방식들은 XML 문서를 객체 트리의 형태로 다루고, 각 객체를 하나의 테이블로 매핑 시키고 테이블간의 조인을 이용하여 계층을 유지한다. 그리고 이것은 상용화 중인 RDBMS (Relational Database Management System)들이 제공하는 방식이다. 하지만 트리 기반의 XML 문서가 표준화된 관계형 데이터베이스 테이블에 저장되어질 경우, 객체 지향과 RDBMS간의 불일치의 문제점[11]이 발생한다. 결국, 객체 지향의 개체 관계를 관계형 데이터베이스의 테이블로 표현을 할 경우 쿼리는 복잡해지고 테이블조인도 늘어나게 된다. 그리고 객체가 많아지고 구조가 복잡해지면 그에 따라 테이블도 늘어나고 테이블조인의 횟수도 늘어난다. 이것은 RDBMS의 성능저하에 결정적인 역할을 한다. 이러한 문제를 극복하기 위한 대안으로 여러 XML 저장 매핑 기술이 연구되었다. 이와 같은 연구들은 테이블간의 결합을 줄임으로써 데이터베이스의 성능을 개선시킬 수 있는 방법들을 제안하였다. 그 중 연구[10]에서는 인라인기법을 제안한다. 이 방법은 DTD를 단순화 시켜 XML 문서와 RDBMS의 매핑을 단순화하여 성능의 향상을 노린다.

2.1 인라인 기법(Inlining Technique)

데이터베이스의 스키마를 생성하기 전에 DTD나 XML 스키마의 복잡도를 줄이는 것은 관계형 데이터베이스의 성능을 향상시키는 방법이다. 인라인 기법은 엘리먼트를 그 부모의 엘리먼트에 포함시키는 방식으로 포함시키는 조건에 따라 기본 인라인(Basic Inlining), 공유 인라인(Shared Inlining), 혼합 인라인(Hybrid Inlining)으로 나눌 수 있다[3].

기본 인라인(Basic Inlining)은 엘리먼트들을 각각의 릴레이션으로 사상하고, 하위 엘리먼트는 사상된 릴레이션의 속성으로 사상된다. 다중 값이나 순환구조를 갖는 엘리먼트는 별도의 릴레이션에 사상된다. 기본 인라인 방법은 과도한 릴레이션이 생성되고 엘리먼트가 여러 릴레이션으로 나누어지기 때문에 질의처리 시 여러 릴레이션을 스캔해야하는 단점을 가진다.

공유 인라인(Shared Inlining)은 그래프에서 2 이상의 진입차수(in-degree)를 가지는 노드와 *에지로 연결이 되는 노드에 대한 릴레이션을 각각 생성한다. 그리고 진입 차수가 1인 노드를 자신의 부모노드에 인라인시킴으로써, 진입차수가 1인 노드에 대한 릴레이션을 따로 생성하지 않는다. 공유 인라인 기법은 한 엘리먼트 노드를 정확하게 한 릴레이션에만 표현한다. 만약 동일한 엘리먼트가 다른 릴레이션에 표현된다면, 해당 엘리먼트를 별도의 릴레이션으로 생성한다. 공유 인라인 기법은 일부 엘리먼트들을 속성으로 취급함으로써 과도한 릴레이션이 생성되거나 중복되는 릴레이션이 생성되는 것을 방지한다.

혼합 인라인(Hybrid Inlining)은 공유 인라인 방식에서 발생하는 단점을 해결하기 위하여 공유되는 릴레이션을 속성으로 사상하여 조인 횟수의 증가를 막는다. 그러나 이 방법은 엘리먼트가 여러 속성으로 사상되기 때문에 여러 릴레

1. $e^+ \rightarrow e^*$.
2. $e? \rightarrow e$.
3. $(e_1|\cdots|e_n) \rightarrow (e_1,\cdots,e_n)$.
4. (a) $(e_1,\cdots,e_n)^* \rightarrow (e^*_1,\cdots, e^*_n)$.
   (b) $e^{**} \rightarrow e^*$.
5. (a) $\cdots,e,\cdots,e,\cdots \rightarrow \cdots,e^*_n,\cdots,\cdots$.
   (b) $\cdots,e,\cdots,e^*_n,\cdots \rightarrow \cdots,e^*,\cdots,\cdots$.
   (c) $\cdots,e^*,\cdots,e,\cdots \rightarrow \cdots,e^*_n,\cdots,\cdots$.
   (d) $\cdots,e^*_n,\cdots,e^*,\cdots \rightarrow \cdots,e^*_n,\cdots$

(그림 1) 인라인 규칙(rules for Inlining)

이션에서 처리된 값들을 합병해야하는 복잡한 과정이 요구된다.

3. Inlning 기법의 적용과 구조적 유사성에 따른 엘리먼트 분류 기법

이 장에서는 MAGE-ML의 스키마에 인라인 기법을 적용하고, 그 스키마에 본 논문에서 새롭게 제시하는 구조적 유사성 기반의 엘리먼트 분류 기법을 적용하는 과정을 보인다. 두 가지 기법을 모두 사용함으로써 MAGE-ML은 더욱 단순화되고, 이것은 XML문서를 RDBMS로 저장할 경우 효율적인 매핑을 가능하게 해준다.

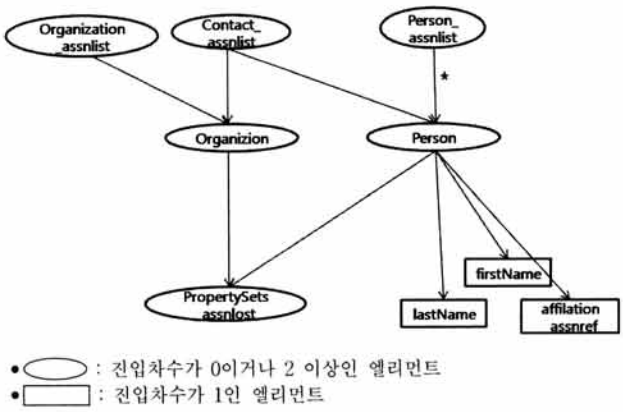
3.1 공유 인라인(Shared Inlining) 기법을 적용한 MAGE-ML

공유 인라인은 인라인 기법 중에서 질의 처리 능력과 저장 공간의 효율면에서 가장 좋은 결과를 보인다[3]. 본 논문에서는 인라인 기법 중 공유 인라인 기법을 이용하여 MAGE-ML을 인라인한다. MAGE-ML의 스키마에 정의되어있는 엘리먼트의 숫자는 총 424개이다. <표 1>은 MAGE-ML의 스키마에 정의되어있는 엘리먼트를 진입차수(in-degree)에 따라 구분한 것이다.

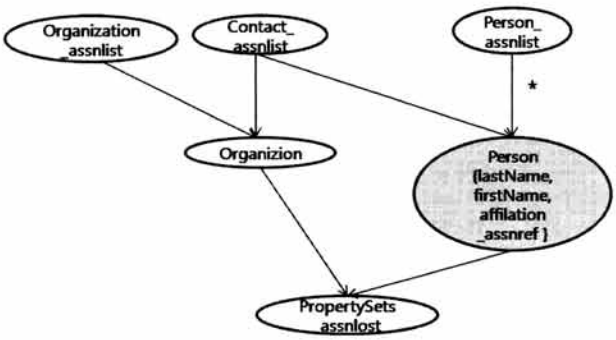
공유 인라인 기법에서 진입차수가 1인 엘리먼트는 인라인(inline) 되므로 테이블이 따로 생성이 되지 않는다. 그러나 진입차수가 1인 엘리먼트 중에서 *로 연결되는 에지라면 공유노드가 되므로 인라인하지 않는다. 참조되어지는 엘리먼트는 <표 1>에서와 같이N MAGE-ML에서 진입차수가 1인 엘리먼트의 숫자는 전체의 278개 이다. 그 중에서 *로 연결되는 엘리먼트는 111개이다. 즉 *로 연결되는 엘리먼트를 제외한 167개의 엘리먼트가 인라인 가능하다는 것을 의미한다. 그리고 진입차수가 0이거나 2 이상일 경우 따로 테이블을 생성한다. (그림 1)의 알고리즘에 따라 구분한 결과 진입차수가 0이거나 2이상인 146개의 엘리먼트들과 *로 연결되는 진입차수가 1인 엘리먼트들은 32개의 그룹으로 구분된다.

〈표 1〉 진입차수에 따라 분류된 엘리먼트의 숫자(the number of elements classified by in-degree)

진입차수	엘리먼트의 숫자
0	9
1	278
2이상	137



(그림 2) MAGE-ML의 부분 그래프



(그림 3) 공유 인라인 적용

(그림 2)는 실제 MAGE-ML 스키마의 일부분을 그래프로 표현한 것이다. 진입차수가 0인 엘리먼트들은 다른 노드로부터 연결될 수 없으므로 테이블을 따로 생성한다. 그리고 진입차수가 2이상인 엘리먼트들 또한 테이블을 따로 생성한다. 그리고 진입차수가 1인 엘리먼트들은 부모엘리먼트에 인라인 된다. 그 결과 (그림 3)과 같은 그래프의 형태가 된다.

3.2 구조적 유사성을 이용한 엘리먼트 분류

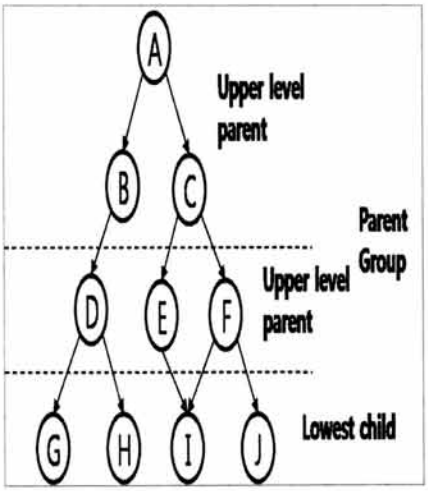
이 장에서는 본 논문에서 새롭게 제시하는 MAGE-ML로부터 구조적 유사성을 가지고 있는 엘리먼트(element)의 분류 알고리즘에 대하여 서술한다. 여기서 사용되는 스키마는 이전 장에서 인라인 기법을 통하여 단순화된 스키마이다. 그리고 그 스키마를 구조적 유사성 기반의 알고리즘을 이용한 엘리먼트 분류를 통하여 단순화 시킨다.

구조적 유사성을 기반으로 하는 엘리먼트 분류 알고리즘의 핵심이 되는 분류 규칙에 대하여 알아보자. 각 엘리먼트들은 형태에 따라 그룹에 속하게 된다. 그 그룹은 계층에서 자식을 갖지 않는 LCG(Lowest Child Group)와 PG(Parent Group)로 구분한다. 그리고 PG는 자식 엘리먼트의 형태에 따라 ULPG(Upper Level Parent Group)과 LPG(Lowest Parent Group)으로 구분한다. 모든 분류가 끝났을 때 엘리먼트의 그룹은 다양하다. 그러나 계층적인 관점에서 각 엘

```
<xsd:element name="A">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="B"/>
      <xsd:element ref="C"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="B">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="D"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="C">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="E"/>
        <xsd:element ref="F"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="D">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="G"/>
      <xsd:element ref="H"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="E">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="I"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="F">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="I"/>
        <xsd:element ref="J"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="G">
  <xsd:complexType>
   <xsd:attribute name="identifier" use="required"/>
   <xsd:attribute name="name"/>
  <xsd:complexTyppe/>
<xsd:element/>
<xsd:element name="H">
  <xsd:complexType>
   <xsd:attribute name="identifier" use="required">
   <xsd:attribute name="name"/>
  <xsd:complexTyppe/>
<xsd:element/>
<xsd:element name="I">
  <xsd:complexType>
    <xsd:attribute name="width" use="required"/>
    <xsd:attribute name="hight" use="required"/>
    <xsd:attribute name="length" use="required"/>
  <xsd:complexTyppe/>
<xsd:element/>
<xsd:element name="J">
  <xsd:complexType>
   <xsd:attribute name="identifier" use="required"/>
   <xsd:attribute name="name"/>
  <xsd:complexTyppe/>
<xsd:element/>
```



(그림 4) XML 스키마의 구조적 표현(structural expression of XML Schema)

리먼트는 LCG, LPG, ULPG로 구분된다. 즉, 같은 계층에 있는 LCG일지라도 최종적인 분류 후에는 자식의 숫자에 따라서 LCG_1, LCG_2, ..., LCG_n로 구분이 된다. 이 과정에서 사용하는 분류알고리즘에서 사용하는 용어들에 대하여 다음과 같이 정의한다.

- *e*: XML 스키마에서 정의된 하나의 엘리먼트
- *E*: 엘리먼트*e* 집합
- *SE*: *e*의 하위 엘리먼트(sub-elements) 집합
- *a*: *e*의 속성
- *A*: *e*의 속성 집합
- *SA*: *e*의 모든 하위 엘리먼트를 위한 속성 집합
- *complexType*: SE의 구성이나 *e*의 A에 대한 구조적 정보
- *Lowest child*: 최하위 엘리먼트(즉, 하위 엘리먼트가 없는 엘리먼트)
- *Lowest parent*: 하나의 최하위 자식 엘리먼트를 하위엘리먼트로 갖는 엘리먼트
- *PG*(parent Group): 최하위 자식의 부모가 될 수 있는 후보 엘리먼트들의 집합
- *LCG*(The Lowest Child Group): 최하위 자식 엘리먼트들의 집합
- *LPG*(The Lowest Parent Group): 최하위 자식의 부모

엘리먼트들의 집합

- *ULPG*(Upper Level Parent Group): 최하위 부모나 최하위 자식이 아닌 엘리먼트를 포함하는 상위 레벨 부모들의 집합

(그림 4)에서 실제 XML 스키마를 그래프로 표현하였다. 그리고 각 엘리먼트가 분류되었을 때 어떤 그룹에 속하는지를 그래프로 표현한다.

XML 스키마에서 정의된 모든 엘리먼트들은 하위 엘리먼트와 속성의 구성을 정의하는 *complexType*로 구성된다. 특정한 형태의 엘리먼트 ele_x 가 *complexType*을 가지고 있고, ele_x의 각 구성 엘리먼트 집합은 다음과 같이 정의되어 있다고 가정한다면, 다음과 같이 표현할 수 있다.

$SE = \{e_1, e_2, \ldots, e_n\}$, $SA_{elex} = \{A_{e1}, A_{e2}, \ldots, A_{en}\}$
$complexType(ele_x) = \{SE, SA_{elex}\}$
이때, SA_{elex}는 ele_x의 SA, A_{en}는 e_n의 속성

일정한 엘리먼트 ele_x의 *complexType*에 있는 *SE* 또는 SA_{elex}가 다른 엘리먼트 ele_y의 *complexType*에 존재하는 *SE* 또는 *SA* 와 정확히 일치할 때, ele_x와 ele_y는 같은 *complexType*을 가지고 있다고 말한다.

3.1.1 분류 규칙

*complexType*의 정의에 따라, 유사 엘리먼트들의 분류를 위한 의사결정나무에서 분기조건으로 사용되는 네 가지 규칙을 정의하였다. 다음은 엘리먼트의 분류를 위하여 본 논문에서 제시하는 핵심 알고리즘의 네 가지 분류 규칙과 pseudo code이다.

- 규칙 1: 만일하나의 엘리먼트가 어떤 하위 엘리먼트도 가지고 있지 않다면 그 엘리먼트(예, Figure 1에서의 노드 G, H, I, J)는 *LCG*의 엘리먼트로 분류 된다. 그렇지 않고 하위 엘리먼트를 가지고 있다면 엘리먼트(예, Figure 1에서의 노드 A, B, C, D, E, F)는 *PG*의 엘리먼트로 분류 된다. 규칙 1은 한 엘리먼트가 *LCG* 그룹이나 *PG* 그룹 어디에 속할 수 있을지를 결정한다.

```
For each ei  E
   if(number of elements in SEi == 0)
      ei is classified into LCG
   end if
   else
    ei is classified into PG
end for
```

- 규칙 2: 규칙1의 *LCG*를 LCG_0라고 하자. LCG_0의 몇 엘리먼트들은 하나 또는 그 이상의 속성들이 정의 되었을 때 같은 *complexType*을 가진 엘리먼트들과 새로운 그룹 LCG_p로 분류 되어 진다($p > 0$). 만일 이미 같은 *complexType*를 가진 LCG_p 그룹이 있다면 그 엘리먼트는 결국 LCG_p 그룹이 된다. 즉, 규칙 2는 *LCG*의 여러 집합을 분류한다.

```
p = 0;
For each ei  LCG0
     Flag=0;
     If (p>0)
                For q=1 to p
        If complexType(ei) = complexType(element in LCGq)
                   ei is classified into  LCGq
                   Flag=1;
        end if
     end if
     If (Flag==0)
       for each ej  LCG0
         if(complexType(ei) = complexType(ej)
              p=p+1
              ei and ej are classified into a new group of LCGp
         end if
       end for
   end if
end for
```

- 규칙 3: 만일*PG*에 특정한 엘리먼트가 오직 *LCG*에 속한 하위 엘리먼트만을 가지고 있다면 그 엘리먼트는 *LPG*의 엘리먼트로 분류 된다. 그렇지 않다면 그 엘리먼트는 *ULPG*의 엘리먼트로 분류 된다. 이것이 *PG* 엘리먼트를 *LPG*와 *ULPG* 이 두 가지 그룹으로 분리하는 규칙 3이다.

```
For each ei  PG
   if(SEi  LCG)
      ei is classified into LPG
   end if
   else
      ei is classified into ULPG

end for
```

- 규칙 4: 규칙 3의 *LPG*를 LPG_0라고 하자. LPG_0의 엘리먼트들이 하나 또는 그 이상의 속성들이 정의되어 있는 *complexType*을 가지고 있다면, 같은 *complexType*을 가지고 있는 그 엘리먼트들은 새로운 LPG_p로 분류 된다.($p>0$) 만약 같은 *complexType*을 가지고 있는 LPG_p 그룹이 이미 있다면 그 엘리먼트는 LPG_p의 그룹이 된다. 즉, 규칙 4는 LPG의 여러 집합을 분류한다.

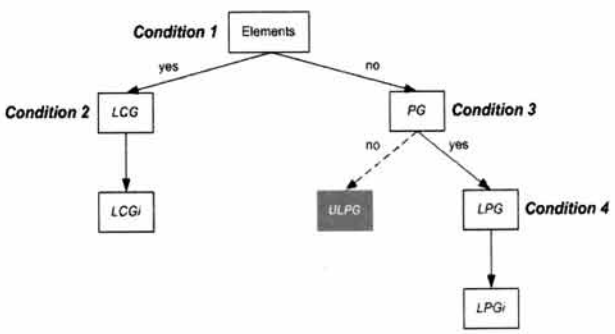
```
p = 0;
For each ei  LPG0
Flag=0;

If (p>0)
For q=1 to p
   If (complexType(ei) is complexType(element in LPGq))
      ei is classified into  LPGq
      Flag=1;
     end if
  end for}
end if

    If (Flag is 0)
    For each ej  LPG0
      if(complexType(ei) is complexType(ej))
      p=p+1
      ei and ej are classified into a new group of LPGp
         end if
      end for
    end if

end for
```

3.1.2 분류를 위한 의사결정나무의 분기 조건

(그림 5)는 3.1장에서 정의한 네 가지 규칙을 기반으로 이진 의사결정트리를 구성한 것이다. 정의된 의사결정나무는 4개의 가지 노드와 3개의 단말 노드를 가지고 있다. 이 트리의 최고 레벨은 3이다. 규칙1의 실행 후, 모든 엘리먼트들은 두 개의 노드들(*LCG*와 *PG*)로 나뉜다.

조건 1, 2, 3, 4는 규칙 1, 2, 3, 4로부터 만들어진다. 규칙 1, 2, 3, 4 적용되어진 엘리먼트들에 초점이 맞춰진다. 규칙을 만족시키는 엘리먼트를 위해 엘리먼트의 *complexType*은 규칙문서에 조건과 정확히 일치 되어야 한다. 이 조건들은 다음에 오는 것들이다.

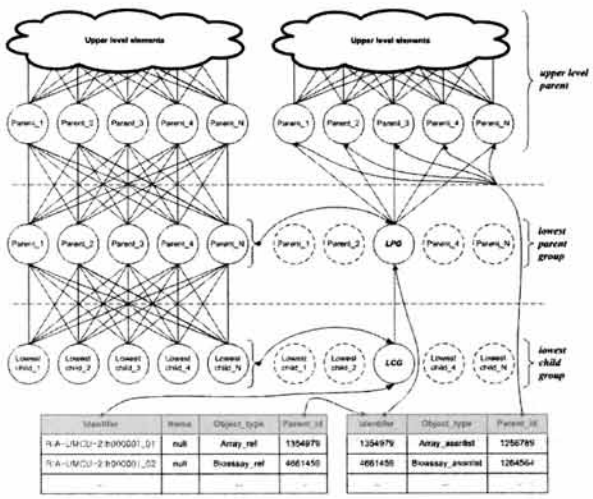
- 조건 1: 규칙 1이 만족된다면, e는 *LCG*로 분류되고, 그렇지 않다면 *PG*로 분류된다.
- 조건 2: 규칙 2가 만족된다면, e_x, 그리고 e_x와 유사한 엘리먼트 e_y는 새로운 *LCG*이다.
- 조건 3: 규칙 3이 만족된다면, e는 LPG 이다. 그렇지 않다면 그것은 *ULPG*이다.
- 조건 4: 규칙 4가 만족된다면, e_x, 그리고 유사한 엘리먼트 e_y는 *LPG* 이다.



(그림 5) 의사결정나무의 간단한 도식(Diagram of decision tree)

3.1.3 분류 알고리즘을 적용한 스키마로부터의 데이터베이스 구조

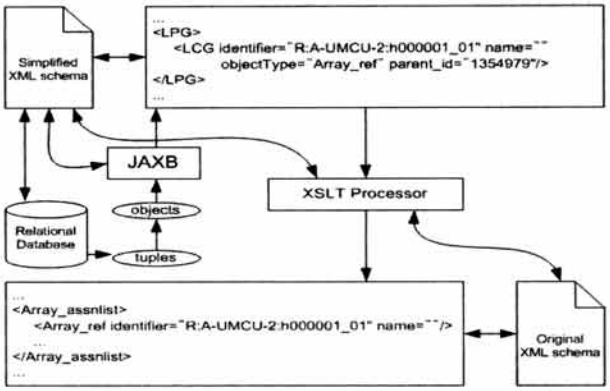
3장에서 설명한 새로운 분류 알고리즘을 이용하여 분류되어진 간략화된 MAGE-ML 데이터를 위한 XML 스키마로부터 실제 생성되는 데이터베이스 테이블을 설명한다. (그림 6)은 공개저장소의 개념적 구조이다 *LCG*와 *LPG*는 N대N의 관계를 갖는다. 이것은 *LPG*와 *LCG*를 위한 최소한의 테이블 숫자를 갖는데 초점을 맞춘다. 이것은 데이터베이스의 복잡도를 개선하는 효과를 가져 온다. 본문에서 *LCG*와 *LPG*가 특수한 데이터이고 반복해서 나타나기 때문에 분류규칙에서 최하위 자식 집단(*LCG*)과 최하위 부모 집단(*LPG*)에 초점을 맞추었다.



(그림 6) 분류되고 조직화된 데이터베이스 테이블(classified and structured database table)

4. XML 문서의 복원

원본 데이터가 분류되어 테이블에 저장될 때, 서로 다른 객체가 같은 테이블에 저장이 되지만, 각 객체는 각자 자신의 identifier를 유지하고 그 identifier를 참조키로 사용하여 외부조인(outer-join)을 하기 때문에 데이터의 내용과 계층을 유지하는 것이 가능하다. 이 방식을 사용하여 기존의 연구[9]에서는 튜플(tuple)들을 복구하기위해 테이블 집합에 접근하고, 그 결과들을 결합하는 것을 보였다. 그리고 그 결과들을 XML 문서의 형태로 바꾸기 위해 정렬된 외부조인(outer-join) 메서드를 사용하여 XML 문서의 형태로 복원한다. 이 작업을 간단히 하기 위해 본문의 실험에서는 관계형 데이터베이스로부터 자동으로 XML 문서를 생성하는 도구(tool)를 사용하였다. JAXB (Java Architecture for XML Binding)[12]는 객체를 XML 문서로 변환하는 기능과 그에 대한 역 변환 또한 제공한다. 우선, 관계형 데이터베이스로부터 튜플들을 검색하고 결과들을 객체에 사상한다. 그리고



(그림 7) XML 문서 재구조화의 흐름도(flow chart of reconstruction of XML Document)

그것들을 JAXB를 사용하여 XML 문서로 변환한다. 이때 데이터베이스가 XML 스키마 중심의 구조이기 때문에 JAXB를 통하여 관계형 데이터베이스로부터 XML 문서를 생성하는 것은 매우 효율적이다. 그러나 이 XML 문서는 완전하지 않다. 왜냐하면 XML 스키마 중심의 데이터베이스 설계 전에 XML 스키마는 이미 단순화되었기 때문이다. 그래서 스키마로부터 완전한 XML 문서를 재구조화하기 위해 XSLT(XML Stylesheet Language Transformations)[13]를 사용 하였다. (그림 6)에서 볼 수 있는 것처럼 각각의 최하위 자식 엘리먼트들은 객체형태(object_type)뿐만 아니라 부모의 ID(parent_id)도 가지고 있다. 따라서 XSLT 처리기는 쉽게 원본 XML 스키마에 명시되어 있는 순서를 알 수 있다. (그림 7)는 이러한 문서 재구조화의 흐름을 나타낸다.

5. 실험 결과

실제 데이터를 본문에서 제안하는 알고리즘을 적용하여 분류한 뒤, 실제 데이터베이스에 저장시켜 보았다. XML문서를 저장하기 위한 데이터베이스는 MySQL을 사용하였다. 그리고 저장되어진 데이터를 원본 XML 문서로 복원하기 위하여 JAXB를 사용하였다. 스키마의 엘리먼트들을 공유 인라인 기법을 사용하여 단순화하고 본 문에서 제시하는 방식을 사용하여 더욱 단순화 시켜 데이터베이스를 디자인하였다. 기존의 엘리먼트 객체와 공유 인라인 기법만을 사용하였을 경우보다 데이터베이스의 복잡도가 줄어든 것을 알 수 있다. 그리고 (그림 6)에서 본 것처럼 실제 데이터들은 기존의 방식처럼 각 객체가 하나의 테이블을 이루는 것이 아니라 구조적으로 같은 객체들이 분류되어 하나의 테이블에 저장이 된다. 그리고 XML문서를 저장하고 JAXB를 이용하여 데이터베이스에 저장된 데이터를 원본 XML문서로 복원하는 실험도 함께 하였다. 이때, 원본 스키마와 인라인된 스키마, 그리고 구조적 유사성기반의 분류 기법을 적용한 스키마를 이용하여 저장된 데이터를 복원하는 과정은 JAXB와 XSLT를 이용하였다.

5.1 공간 복잡도 비교

<표 2>에서 보이는 MAGE-ML의 개선된 스키마(schema)를 도입함으로써 기존의 방식에 비하여 데이터베이스의 XML 데이터 저장의 복잡도를 개선하였다. 분류 규칙을 따르는 테이블을 만든 이후, 클래스는 테이블에 사상되고 클래스와 테이블의 전체 숫자는 같아졌다.

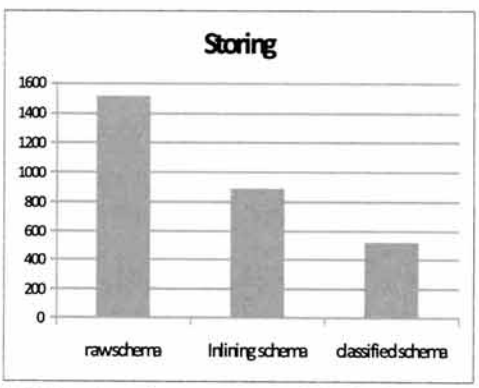
〈표 2〉 데이터베이스 공간 복잡도(space complexity value of database)

	원본 스키마	공유 인라인	구조적 유사성 분류
전체 클래스	455	257	199
전체 테이블	455	257	199
전체 레코드	2012	547	140
전체 DB 크기	710 (Kb)	367 (Kb)	211 (Kb)
전체 테이블 결합	101	74	24

5.2 시간 비교

데이터베이스 공간의 복잡도 감소를 통하여 저장(storing)과 로딩(loading)이 개선되었다. 본 논문에서 사용된 XML 데이터는 1Mbyte의 크기이고 모든 엘리먼트들은 *LCG*와 *LPG*로 분류된다.

(그림 8)은 분류된 스키마와 원본 스키마의 XML 데이터의 저장과 로딩을 위한 처리시간을 보여준다. 이것은 새롭게 제시하는 방식을 통하여 설계한 데이터베이스가 기존의 방식들보다 뛰어난 성능을 보인다는 것을 의미한다.



(그림 8) Stroring 과 Loding 소요 시간 비교(comparison of storing and loading)

6. 결 론

본 논문에서는 생물학적 정보를 위한 XML 스키마를 구조적 특징을 바탕으로 하여 효율적으로 저장하는 새로운 접근방식을 보인다. 기존의 방식들은 XML Schema에 정의된 특정 엘리먼트들을 부모 엘리먼트, 현재 엘리먼트, 자식 엘리먼트 등과 같이 계층별로 구분하여 조상엘리먼트와 자손 엘리먼트 사이에 생략 가능한 엘리먼트를 찾아 제거하는 것

이다. 이러한 방법은 생략 가능한 엘리먼트의 얼마나 많이 정의되었느냐에 의존한다. 본 논문에서 제시하는 방식은 기존의 방식과는 다르게 구조적 유사성에 따라 엘리먼트들을 구분하는 방식으로써 MAGE-OM/ML과 같은 유사한 구조가 반복적으로 나타나는 마이크로어레이 데이터에 더욱 효율적이다. 실험의 결과들은 스키마의 단순화가 XML의 데이터베이스 저장 시 공간의 복잡도를 줄이고 테이블조인을 줄임으로써 성능이 개선되는 것을 보여준다.

참 고 문 헌

[1] Randy Z. Wu, Steve N. Bailey and David M. Sabatini, "Cell-biological applications of transfected-cell microarrays", TrendsinCellBiology12, pp.485-488, 2002,

[2] P. T. Spellman, et al, "Design and implementation of microarray gene expression markup language (MAGE-ML)", GenomeBiol233(9)RESEARCH.1-0046. 9, 2002.

[3] J. Shanmugasundaram, K. Tufte, G. He, C. Zhang, D. Detwitz and J. Naughton, "Relational databases for querying xml documents: Limitations and opportunities", In Proc. Intl. Conf. on 25th VLDB, 1999.

[4] U. Sarkans, H. Parkinson, G. G. Lara, A. Oezcimen, A. Sharma, N. Abeygunawardena, S. Contrino,E. Holloway, P. Rocca-Serra, G. Mukherjee, M. Shojatalab, M. Kapushesky, S. A. Sansone, A. Farne, T. Rayner, A. Brazma, "The ArrayExpress gene expression database: a software engineering and implementation perspective", Bioinformatics. 21(8), pp.495-501, 2005.

[5] A. Catherine Ball, A. B. Ihab. Awad, Janos Demeter, Jeremy Gollub, Joan M. Hebert, Tina Hernandez-Boussard, Heng Jin, C. Matese John , Michael Nitzberg, Farrell Wymore, K. Zachariah, O. Patrick Brown and Gavin Sherlock. "The Stanford Microarray Database accommodates additional microarray platforms and data formats", Nucleic Acids Research, pp.33, 2005.

[6] W. Martin, R.M. Horton, Magebuilder, "A schema translation tool for generating MAGE-ML from tabular microarray data", Bioinformatics Conference, CSB 2003, pp.431-432, 2003.

[7] S. Abiteboul, P. Buneman, D. Suciu, 1st ED. "Data on the web", MorganKaufmann, 2000.

[8] H. Schoning, "Tamino - A DBMS designed for XML", In Proceedings of the 17th International Conference on Data E ngineering 2-6, pp.149-154, 2001.

[9] I. Tatarinov and S. Viglas, "Storing and Querying Ordered XML Using a Relational Database System", In Proceedings of the 2002 ACM SIGMOD international conference on Management of data, pp.204-215, 2001.

[10] K. Runapongsa and J. M. Patel, "Storing and Querying XML Data in Object-Relational DBMS", In EDBT 2002 Workshop on XML-Based Data Management and Multimedia Engineering, LNCS 2490, pp.266-285, 2002.

[11] S. Ambler, D. A. Chapam, "Agile Database Techniques: Effective Strategies for the Agile Software Developer", WILEY, 2003.

[12] JAXB (Java Architecture for XML Binding), http://java.sun.com/xml/downloads/jaxb.html

[13] XSLT (XML Stylesheet Language Transformations), http://www.w3.org/Style/XSL/

윤 종 한

e-mail : jhyoon@gce.sejong.ac.kr

2007년 세종대학교 컴퓨터공학과(학사)

2009년~현 재 세종대학교 검퓨터공학과 석사과정

관심분야 : 데이터베이스, 생물정보학, 데이터마이닝 등

신 동 규

e-mail : shindk@sejong.ac.kr

1986년 2월 서울대학교 계산통계학과 (이학사)

1992년 8월 Illinois Institute of Technology 전산학과(공학석사)

1997년 8월 Texas A&M University 전산학과(공학박사)

1986년 2월~1991년 8월 한국국방연구원 연구원

1997년 8월~1998년 2월 현대전자 멀티미디어연구소 책임연구원

1998년 3월~현 재 세종대학교 컴퓨터공학과 교수

관심분야 : 상황인식 미들웨어, 웹기반 멀티미디어, 멀티미디어 DRM

신 동 일

e-mail : dshin@sejong.ac.kr

1988년 연세대학교 전산과학과(이학사)

1993년 M.S. in Computer Science, Washington State University

1997년 Ph.D in Computer Science, University of North Texas

1997년 9월~1998년 2월 시스템공학연구소 선임연구원

1998년 3월~현 재 세종대학교 컴퓨터공학과 부교수

관심분야 : 상황인식 미들웨어, 무선인터넷, 게임, 지능형 에이전트, HCI