이질의 생물 정보원 통합을 위한 랩퍼 시스템에서의 XML 질의 처리 시스템

박 은 경[†] · 강 동 완[††] · 정 채 영[†††] · 김 현 주[††††] · 배 종 민[†††††]

요 약

분산된 생물 정보원을 물리적으로 혹은 가상적으로 통합하기 위해서는 정보원들의 이질성을 해결해야 하며, 사용자의 다양한 요구를 수용할 수 있도록 범용성과 적응력이 뛰어나야 한다. 본 논문에서는 데이터는 물론 응용 프로그램의 통합을 위한 미들웨어를 설계함에 있어서 랩퍼 시스템에서의 융통성 있는 범용 XML 질의 처리 엔진을 제시한다. 제시된 질의 처리 엔진은 사용자 정의 XML 뷰를 지원함으로서 융통성 있는 통합 질의 구성을 가능하게 한다. 질의 처리 과정은 뷰 합성, 지역 정보원에 대한 질의 변환, 그리고 결과 문서 생성과정을 동반하는데, 이를 위해 XML 뷰와 XML 질의어를 뷰 트리로 표현하는 XML 뷰 트리 기반의 질의 처리 모델을 제시한다. 그리고 제시된 질의 처리 모델의 범용성을 확인하기 위해 관계형 데이터베이스와 웹 정보원, 그리고 응용 프로그램에 대하여 정보원의 질의 결과 형이 관계형 튜플과 XML 문서인 경우에 대한 질의 처리 방법을 제시한다.

키워드 : 생물 정보원 통합, XML, XML 뷰, 사용자 정의 XML 뷰, XML 질의 처리, 랩퍼

An XML Query System in a Wrapper System for Integrating Heterogeneous Biological Databases

Eun-Koung Park[†] · Dong-Wan Kang[††] · Chai-Young Jung[†††] · Hyun-Ju Kim[††††] · Jong-Min Bae[†††††]

ABSTRACT

In order to integrate distributed biological information sources physically or virtually, it is necessary to overcome the heterogeneity of information sources and support a superior generality and adaptation in order to satisfy user's various demands. In this paper, we present a flexible and general XML query engine of a wrapper system in designing the middleware system to integrate data as well as application programs. Since this query engine applies user-defined XML view, it is possible to composite flexible integrated query. The query processing in a wrapper requires view composition, query translation into local sources, and generation of XML documents from local query results. We present a query processing model based on the view tree, where the XML views and the XML queries are represented by the view tree. Moreover, to confirm the generality of our query processing model, we present a methodology of query processing for relational databases, web sources, and application programs whose return types of query results are relational tuples or XML documents.

Key Words : Intregration of Biological Information Sources, XML, XML View, User-defined XML View, XML Query Processing, Wrapper

1. 서 론

생물학적 실험환경의 급격한 발전으로 인하여 많은 생물정보원이 개발되고 또한 방대한 생물데이터를 분석하고 처리하기 위한 소프트웨어 도구들이 개발되었다. 특히 생물정보원들은 역사적인 이유로 인하여 이질성이 매우 높다. 이에 따라 정보원 통합에 관한 연구는 그 역사가 오래되었음에도 불구하고, 생물정보원의 통합문제는 새로운 이슈가 되었다[14]. 생물정보원의 데이터는 관계형 모델, 객체형 모델, 혹은 플랫(flat) 파일로 관리되기도 하고 웹 데이터베이스로 제공되기도 한다. 또한 정보원에 접근하기 위해서는 정보원이 제공하는 프로그래밍 인터페이스를 활용할 수도 있고, SQL과 같은 표준 질의어를 사용할 수도 있으며, 웹 검색엔진을 사용할 수도 있고, 폼(form)기반의 인터페이스를 사용할 수도 있다. 그리고 질의에 대한 결과는 관계형 튜플(tuple)이나 객체일 수 있고, XML 혹은 HTML 문서일 수도 있으며, ASN.1과 같은 데이터교환 양식일 수도 있다[2]. 분

※ 본 연구는 학술진흥재단(KRF-2004-002-D00380)의 지원으로 수행되었음.
† 준 회 원 : 경상대학교 컴퓨터과학과 대학원
†† 정 회 원 : 미디어코러스(주)
††† 정 회 원 : 경상대학교 컴퓨터과학과 대학원
†††† 정 회 원 : 진주산업대학교 컴퓨터공학부 조교수
††††† 종신회원 : 경상대학교 컴퓨터과학부/컴퓨터정보통신연구소 교수(교신저자)
논문접수 : 2005년 1월 21일, 심사완료 : 2005년 6월 20일

산된 생물정보원을 물리적으로 혹은 가상적으로 통합하기 위해서는 이와 같은 정보원들의 이질성을 해결해야 한다.

이러한 이질성을 해결하기 위한 방법으로서 모든 정보원을 XML 정보원으로 보는 관점을 제공하면 이질의 정보원들의 모델에 독립적인 질의가 가능하다. 이질의 정보원들을 XML 정보원처럼 사용하기 위해서는 각 정보원을 XML 관점으로 변환하는 XML 기반의 랩퍼(wrapper)가 필요하다. 랩퍼는 각 정보원의 스키마를 XML 뷰(view)로 변환하여 사용자에게 제공한다. XML 뷰는 정보원에 저장된 데이터를 임의의 XML 구조로 사상한 가상적인 XML 문서이다. 사용자는 모든 정보원을 가상적인 XML 정보원으로 간주하여 XML 질의어를 이용하여 질의한다. 이때 랩퍼는 사용자의 XML 질의어를 개별 정보원의 질의어, 예를 들면 SQL, OQL, 웹 URL 등으로 변환해서 정보원에 질의하고, 정보원으로부터 받은 다양한 양식을 가진 질의결과를 XML 질의어의 양식에 맞도록 XML 문서로 변환하여 사용자에게 전달한다.

한편, 생물정보원은 그 특성상 사용자의 요구가 매우 다양하다. 사용자의 다양한 요구를 수용하는 하나의 방법으로서 본 논문에서는 범용성과 적응력이 뛰어난 통합 미들웨어 시스템을 개발한다. 이를 위하여 관계형 데이터베이스에서 사용자 정의 뷰를 지원하는 것과 마찬가지로, 가상적인 XML 정보원에 대해서도 사용자 정의 XML 뷰를 지원한다. 따라서 랩퍼가 지역 정보원에 대한 하위 수준의 XML 뷰를 자동적으로 생성하여 제공하면, 사용자는 이것에 대하여 사용자 정의 XML 뷰를 정의할 수 있어야 한다. 또한, 사용자 정의 XML 뷰를 기반으로 또 다른 사용자 정의 XML 뷰 정의를 지원하면 정보원에 대한 더 높은 추상화가 가능하다.

여기서는 XML 뷰와 질의를 표현하는 언어로서 W3C에서 제안한 XQuery[19]를 사용한다. 사용자가 XML 뷰를 정의하거나 질의를 하기 위해서는 XML 뷰에 대응되는 스키마가 필요한데, 이를 위해서 랩퍼는 XML 뷰에 대한 스키마를 자동으로 생성하여 사용자에게 제공한다. XML 스키마를 표현하기 위한 언어로는 W3C 표준안인 XML Schema[16, 17, 18]를 사용한다.

사용자 정의 XML 뷰를 정의할 수 있으면 사용자 정의 뷰를 기반으로 질의를 할 수 있다. 이 질의를 처리하기 위해서는 사용자 정의 뷰 기반의 질의를 각 정보원에 대한 스키마를 직접적으로 XML 스키마로 사상한 결과인 기본 XML 뷰에 대한 질의로 변환되어야 한다. 이 과정을 질의어 합성이라 한다. 합성된 질의로써 지역 정보원의 질의로 변환한다. 그리고 그 결과는 XML 문서로 변환되어 반환된다. 따라서 질의 처리 과정은 질의어 합성, 변환, 결과 문서 생성의 세 가지 과정을 거친다.

본 논문은 각 정보원의 이질성을 극복하여 데이터는 물론 응용 프로그램 통합을 위한 미들웨어로서의 융통성 있는 범용 질의 처리 시스템을 제시한 다음, 관계형 데이터 모델과 웹 정보원에 대하여, 질의결과가 튜플(tuple) 스트림이나 XML 문서인 경우에 대한 질의처리방법을 보인다. 제시된 모델에서는 정보원의 스키마를 가상의 XML 스키마로 표현한 기본 XML 뷰, 기본 XML 뷰를 기반으로 사용자의 관점에서 가상적인 정보원을 구성할 수 있는 사용자 정의 XML 뷰, 그리고 가상의 XML 정보원에 대한 질의어를 모두 개념적으로 동일시하여 그 표현은 가상적 혹은 실제적 XQuery로써, 그리고 내부 모델은 XML 뷰 트리로써 정의한다.

질의 처리는 뷰의 경로식에 대한 탐색을 통해서 이루어진다. XML 뷰 트리는 임의의 XQuery 표현식으로부터 구성되며, 가상의 XML 문서의 구조를 그대로 반영한다. 제시된 모델은 뷰에 대한 XQuery 경로식의 탐색을 통하여 합성 과정을 쉽게 수행하고, XML 뷰 트리를 질의 결과로 생성되는 XML 문서를 구성하기 위한 템플릿으로 활용할 수 있게 하며, XML 뷰에 대응되는 스키마 생성을 쉽게 한다. 개념적으로 추상화된 XML 뷰와 질의는 동일한 내부 모델로 표현되기 때문에 각 정보원의 이질성을 처리할 때 구현의 중복이 제거되어 최소한의 노력으로 새로운 정보원을 추가하여 통합시스템에 포함시킬 수 있으며, 다단계로 정의되는 XML 뷰를 쉽게 지원한다. 그리고 정보원의 모델에 독립적이기 때문에 정보원의 이질성을 극복하는 골격을 제공한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해 살펴보고, 3장에서는 XML 뷰에 대해서 예를 들어 설명한다. 4장에서는 추상화된 XML 뷰의 트리 모델을 제시하고, 5장에서는 XML 뷰 트리를 이용한 합성 알고리즘을 제시한다. 6장, 7장에서는 각각 XML 뷰 트리를 이용한 질의 변환과 결과 문서 생성 메커니즘을 논하고, 8장에서는 각 정보원 별로 질의 처리 시간에 대한 시험 및 분석 결과를 제시하고, 마지막으로 9장에서는 결론을 보인다.

2. 관련 연구

정보원과 응용프로그램 통합을 위해서는 이들에 대한 XML 뷰를 정의하고, 이를 기반으로 XML 질의처리를 해야 한다. 정보원과 응용프로그램에 대한 기본의 뷰 정의 방법은 OPM(Object Protocol Model) 기반의 뷰 정의 모델[1, 6], ODL(Object Definition Language)이용한 뷰 정의 모델[15], 검색 뷰(search view) 기반의 뷰 정의모델[20], 등 다양하게 있다.

XML 뷰에 관한 기존 연구로서, SilkRoute[10, 11]는 관계형 데이터베이스에 대한 XML 뷰를 정의하기 위해 자체적으로 정의한 선언적 질의어인 RXL(Relational to XML translation Language)를 제공한다. RXL은 SQL 부분과 XML-QL의 조합으로 이루어진 질의어로 블록 구조, 중첩 질의, 그리고 스콜렘 함수 등을 지원한다. 사용자는 RXL을 이용하여 XML 뷰를 기술하고, XML-QL를 이용하여 질의를 한다.

XPERANTO[7, 8, 9]는 관계형 데이터베이스에 대한 XML 뷰와 사용자 질의를 XQuery를 사용하여 표현하며, 사용자 정의 XML 뷰를 지원한다. 이는 [5]에서 정의된 XML 기반의 뷰 정의 모델의 바탕이다. 그런데, 관계형 데이터를

XML 문서로 사상하는 방법에는 여러 가지가 있을 수 있을 수 있는데, 데이터베이스 전체를 하나의 XML 문서로 보는 XPERANTO와는 달리, [5]에서는 하나의 테이블을 하나의 XML 문서로 간주한다. 이는 관계형데이터 뿐 아니라 웹 데이터 등 일반적인 모델에 대한 뷰 정의에 적합하다. 따라서 본 논문에서는 [5]에서 정의된 XML 기반의 뷰 정의 모델에 따른다.

다음으로 랩퍼에서의 질의처리에 관한 기존 연구 중에서 XML 기반의 질의처리에 대하여 논한다. SilkRoute[10, 11]와 XPERANTO[7, 8, 9]은 관계형 데이터베이스의 내용을 XML 문서로 출판하는 미들웨어 시스템이다. 이들은 사용자가 정의하는 XML 뷰를 이용하여 정의하는 가상의 XML 문서를 기반으로 동작한다. 이때 가상의 XML 문서는 실체화되지 않으며 가상의 XML 문서에 대한 사용자 질의는 실제 관계형 데이터베이스에서 사용하는 질의어인 SQL로 변환하는 과정이 필요하다. 이를 위해 XML 뷰와 사용자 질의간의 합성 과정을 거친다. 관계형 데이터베이스를 기반으로 하는 이들 시스템들은 합성 방법이 관계형 데이터베이스에 종속적이다.

SilkRoute에서는 가상의 XML 문서에 대한 문서구조를 유지하는 템플릿과 문서의 각 노드의 생성 규칙을 가지는 스콜렘(Skolem) 함수로 뷰 트리를 구성하며 템플릿에 대한 패턴 매칭과 스콜렘 함수에 대한 질의 재작성을 통해 합성 과정을 수행한다. 합성을 통해 새로운 RXL 질의를 생성하며 이를 실행 가능한 질의(executable query)라 한다. 합성 결과로부터 중복되는 프레디키트와 조인들로 여러 가능성 있는 다수의 질의 플랜을 생성한다. 이때 최적의 질의 플랜을 생성하는 최적화 알고리즘은 NP-Complete이기 때문에 greedy 알고리즘을 사용한다. 선택한 질의 플랜에 의해 생성된 SQL을 수행하여 생성된 결과는 템플릿을 이용하여 통합한다.

XPERANTO는 XML 뷰와 사용자 질의를 XQuery를 사용하여 표현하며, 관계형 모델에서 사용하는 QGM을 확장한 XQGM(XML Query Graph Model)을 내부 질의 모델로 이용한다. XQuery로 기술된 XML 뷰와 사용자 질의는 XQGM으로 변환되어 합성되며 합성 후 새로운 XQGM을 생성한다. 합성 과정에서 사용자 질의의 프레디키트를 XML 뷰로 내려보낸 후, 사용자 질의에 나타난 XML 뷰에 대한 탐색을 제거하고, 사용자 질의에 포함되지 않은 부분은 XML 뷰에서 제거한다. 합성된 XQGM은 SQL로 변환되는 부분과 결과 XML 문서를 생성하는 부분으로 분리되어 각각 질의 변환과 태깅에 이용된다. Sorted Outer Union 방법을 이용하여 하나의 최소화된 SQL을 생성하며 이를 수행한 결과를 이용하여 태깅을 수행한다.

본 논문은 XQuery를 이용하여 XML 뷰와 사용자 질의를 정의하고, XML 뷰와 사용자 질의를 추상화하여 가상적인 XML 문서의 구조를 그대로 반영하는 트리를 구성한다. 합성은 사용자 질의에 나타나는 경로식을 XML 뷰 트리에서 탐색하여 바인딩된 노드에 대한 포인터 연결을 유지하는 것으로 이루어진다. 본 논문에서는 SilkRoute와 달리 기본 XML 뷰를 지원하며 XML 뷰 정의어와 사용자 질의어가 동일하다. 또한 뷰 합성을 통해 하나의 최소화된 질의 플랜을 만들어 낸다. 그리고 XPERANTO와 달리 합성 과정에서 경로식에 대한 추적을 통해 XML 뷰 트리의 질의 정보를 사용자 질의 트리로 모은다. 이는 사용자 질의에 포함되지 않은 부분을 XML 뷰 트리에서 제거하는 과정이 필요하지 않고 한번 생성된 XML 뷰 트리를 다른 사용자 질의와의 합성에 그대로 이용할 수 있다. 이 방법은 관계형에 종속적인 SilkRoute와 XPERANTO와는 달리, 관계형, 객체관계형, 웹 정보원과 같은 기반 데이터모델과는 상관없이 이루어진다. 또한 제안하는 XML 뷰 트리는 합성뿐만 아니라 질의 변환, 결과 문서 생성에 그대로 사용되는 기본 프레임워크 역할을 한다.

3. XML 뷰 정의

지역 정보원에 저장되어 있는 데이터를 XML 문서 관점으로 표현한 뷰를 기본 XML 뷰라고 한다. 그리고 기본 XML 뷰를 기반으로 사용자 관점의 가상의 XML 문서구조를 정의할 수 있는데 이를 사용자 정의 XML 뷰라고 한다. XML 질의처리엔진은 XML 뷰에 대하여 동작하기 때문에, 질의 처리 시스템을 논하기 전에 먼저 관계형 데이터와 웹 데이터, 그리고 응용 프로그램에 대한 XML 뷰를 간단히 보인다. 보다 상세한 내용은 [5]에 있다.

3.1 관계형 데이터베이스에 대한 XML 뷰 정의

(그림 1)은 단백질 발현에 관련된 정보가 저장되어 있는 관계형 데이터베이스의 일부이다.

(그림 2)는 (그림 1)의 데이터를 기본 XML 뷰의 관점에서 본 가상적인 XML 문서를 나타낸 것이다. (그림 2)에서는 보는 것처럼 한 테이블을 하나의 XML 문서로 간주하므로 기본 XML 뷰는 다수의 가상적인 XML 문서의 집합으로 표현된다. 그리나, 이는 랩퍼가 생성하는 기본 XML 뷰에

Patients

id	name	sex
P01	Kang	M
P02	Park	F

Gelspots

id	sspno	chartno	quantity
1001	2468	P01	40
1002	1357	P02	500
1003	1479	P02	50000

Spots

p_id	g_id	normalQty
P01	1001	35
P02	1002	400
P02	1003	45000

(그림 1) 관계형 데이터베이스의 예

<?xml version="1.0"?>
<Patients>
  <tuple>
    <id>P01</id>
    <name>Kang</name>
    <sex>M</sex>
  </tuple>
  <tuple>
    <id>P02</id>
    <name>Park</name>
    <sex>F</sex>
  </tuple>
</PatientGel>

<?xml version="1.0"?>
<GelSpots>
  .... similar to <Patients>
</GelSpots>

<?xml version="1.0"?>
<Spots>
  .... similar to <Patients>
</Spots>

(그림 2) 기본 XML 뷰에 대한 XML 인스턴스

```
<?xml version="1.0" encoding="euc-kr"?>
<PatientGel>
  <Patient ID="P01">
    <Name>Park</Name>
    <Spots>
      <GelSpot>
        <SSPno>1357</SSPno>
        <Quantity>500</Quantity>
        <NormalQty>400</NormalQty>
      </GelSpot>
      <GelSpot>
        <SSPno>1479</SSPno>
        <Quantity>50000</Quantity>
        <NormalQty>45000</NormalQty>
      </GelSpot>
      <Sex>F</Sex>
    </Spots>
  </Patient>
  <Patient ID="XX">
    ....
  </Patient>
</PatientGel>
```

(그림 3) 사용자 정의 XML 뷰에 의해 정의된 가상의 XML 문서 인스턴스

```
1: <PatientGel> {
2:   FOR $patient IN document("Patients")/tuple
3:   WHERE $patient/sex = "F"
4:   RETURN
5:     <Patient ID={$patient/id/text()}>
6:       <Name> $patient/name/text() </Name>
7:       <Spots> {
8:         FOR $gel IN document("Gelspot")/tuple,
9:             $spots IN document("spots")/tuple
10:        WHERE $spots/p_id = $patient/id AND $spot/g_id =
           $gel/id
11:        RETURN
12:          <GelSpot>
13:            <SSPno> $gel/sspno/text() </SSPno>
14:            <Quantity> $gel/quantity/text() </Quantity>
15:            <NormalQty> $spot/normalQty/text() </NormalQty>
16:          </GelSpot>
17:      }
18:      </Spots>
19:      <Sex> $patient/sex/text() </Sex>
20:    </Patient>
21: }
22: </PatientGel>
```

(그림 4) 사용자 정의 XML 뷰의 예

의해 생성되는 가상의 XML 문서이지 실제로 이 문서가 명시적으로 생성되는 것은 아니고, 문서의 구조가 내부적으로 뷰 트리로 유지된다.

이 가상의 XML 데이터베이스를 기초로 사용자가 자신의 관점으로 가상의 데이터베이스를 다시 정의하는 사용자 정의 XML 뷰를 정의할 수 있다. 예를 들어 (그림 1)의 데이터베이스에 대해 사용자 관점의 XML 문서가 (그림 3)과 같다고 가정하자. 이것은 환자들 중, 성별이 여자인 사람들의 이름과 각 Spot 단백질 발현량, 그리고 그들의 성별을 나타내는 XML이다.

이와 같은 가상의 데이터베이스를 구성하기 위해서는 사용자는 (그림 4)와 같이 XQuery를 이용하여 사용자 정의 XML 뷰를 정의한다. XQuery에서 FLWR (For-Let-Where-Return) 표현식의 FOR과 LET절은 바인딩된 변수의 튜플 시퀀스를 생성하는데, 이것을 튜플 스트림이라고 부른다. (그림 4)에서 2번째 행의 FOR절은 기본 XML 뷰 "Patients"의 모든 <tuple> 엘리먼트에 바인딩되는 변수 $patient을 정의하며, 이 변수는 질의 결과를 만드는데 사용된다. WHERE절은 튜플 스트림에 대하여 어떤 튜플을 포함시키거나 배제시키는 필터링을 제공하는데, (그림 4)의 3번째 행에서는 성별이 여자인 튜플만을 포함시키며, 10번째 행에서는 환자와 Gel, 그리고 Spot 간의 조인을 수행하고 있다. RETURN절은 FLWR 표현식의 결과를 구성하는데, WHERE절에 의해서 걸러진 튜플 스트림의 모든 튜플을 한 번씩 차례대로 엘리먼트에 적용한다. XML 엘리먼트를 나타내는 XML 표기는 엘리먼트 생성자인데, 이것은 질의 결과에 해당 엘리먼트를 만들어낸다.

3.2 웹 데이터에 대한 XML 뷰 정의

웹 데이터에 대하여 XML 뷰를 정의하기 위해서는 웹 정보원에 대한 스키마를 알아야 한다. 그러나 웹 자원은 관계형 모델처럼 데이터가 저장되어 있는 스키마를 명시적으로 제공하지 않는다. 그러나, 생물 정보원은 대부분 검색결과에 대한 출력 양식이 일정하게 정해져 있으며, 많은 경우 검색

결과를 일정한 형태의 XML 문서로 제공하거나 플랫 파일(flat file) 형태로 제공한다. 이 사실을 이용하면 검색 결과의 문서 양식을 그 정보원이 관리하고 있는 데이터에 대한 스키마로 간주할 수 있다. 따라서 검색결과 문서양식으로부터 XML 스키마를 생성하여 기본 XML 뷰로 정의한다. 이와 같이 기본 XML 뷰가 정의된 이후에는 사용자 정의 XML 뷰 정의 방법이 앞에서 논한 관계형 정보원에서의 처리방법과 동일하게 이루어지기 때문에 전체적으로 일관된 모델로 질의를 처리할 수 있다.

3.3 응용 프로그램에 대한 XML 뷰 정의

응용 프로그램에 대하여 XML 뷰를 정의하기 위해서는 응용 프로그램도 데이터를 가지는 하나의 정보원으로 간주하는 관점이 필요하다. 그런데 응용 프로그램도 대개 출력결과의 양식이 정해져 있기 때문에 웹 정보원처럼 프로그램의 출력결과에 대한 양식으로부터 XML 스키마를 추출할 수 있다. 그러나 응용 프로그램의 경우에는 입력 데이터를 기반으로 프로그램이 수행되며, 입력 데이터가 반드시 출력 데이터 양식의 엔트리로 포함된다는 보장이 없다. 따라서 응용 프로그램을 하나의 정보원으로 간주할 때 입력 데이터도 응용 프로그램이 관리하는 데이터로 간주해야 나중에 질의변환이 가능하다. 따라서 응용 프로그램의 입력 양식과 출력 양식이 바로 응용 프로그램이 제공하는 데이터에 대한 뷰가 된다. 이 뷰를 XML 스키마로 간주할 때 그것을 기본 XML 뷰로 정의하여, 응용 프로그램을 XML 문서로 구성된 XML 정보원으로 간주할 수 있다.

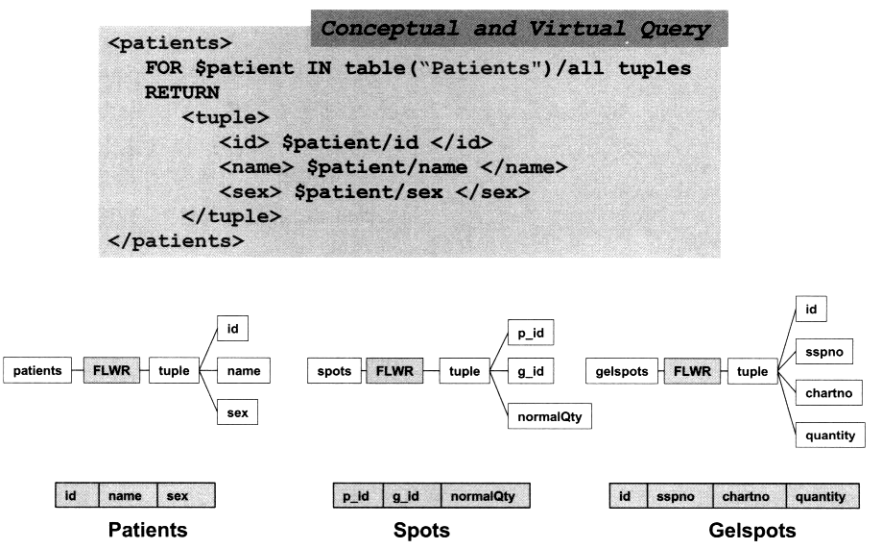
4. XML 뷰 트리

앞에서 언급한 바와 같이, XML 뷰와 질의어는 뷰 트리로 표현된다. 뷰 트리의 구조는 대응되는 가상의 XML 문서의 구조를 그대로 반영하도록 구성된다. 그런데, 기본 XML 뷰는 사용자 정의 XML 뷰나 사용자 질의어와는 다르게 XQuery로 정의할 필요는 없다. 그러나, XML 뷰에 대한 개념적인 추상화를 통하여 모든 XML 뷰를 동일한 데이터 모델로 표현하면 시스템을 유연하게 구성할 수 있으며 구현이 쉽고 구현의 중복을 피할 수 있다. XML 뷰는 가상적인 XML 문서이며, 이 가상적인 XML 문서에 대한 질의 결과 역시 XML 문서이다. 따라서 모든 XML 뷰와 질의어는 개념적으로 통일될 수 있어서 동일한 모델로 표현될 수 있다. 즉, 기본 XML 뷰와 사용자 정의 XML 뷰, 사용자 질의는 컨텍스트(context)에 따르는 용어일 뿐 개념적으로 모두 동일하며, 동일한 모델로 표현될 수 있다. 이 경우, 모든 XML 뷰가 하나의 내부 모델로 표현되기 때문에 구현의 중복이 제거되며, 여러 단계에서 정의된 XML 뷰에 대한 합성을 구현하기 쉬워진다. 또한, 이러한 XML 뷰에 대한 합성은 가상적인 XML 문서에 대해 이루어지므로 지역 정보원의 모델과는 상관없이 독립적으로 수행된다.

4.1 기본 XML 뷰 트리의 생성

기본 XML 뷰는 언급했듯이 XQuery로 정의되지 않는다. 본 통합 미들웨어 시스템에서는 관계형 데이터베이스인 경우, 3.1에서 설명한 기본 XML 뷰의 맵핑 관점을 적용하여 기본 XML 뷰를 구성할 수 있는 개념적이고 가상적인 질의를 설정하고, 이것을 바탕으로 기본 XML 뷰 트리를 랩퍼의 초기화 작업의 일부로서 자동적으로 구성하는 도구를 개발하였다. (그림 5)는 (그림 1)의 단백질 발현량 데이터베이스의 테이블에 저장된 모든 튜플에 대해 바인딩되는 변수를 정의하고, 이 변수를 이용하여 (그림 2)와 같은 XML 문서를 구성하는 개념적이고 가상적인 XQuery 질의와 생성된 기본 XML 뷰 트리를 생성한 결과이다.

(그림 5)에서 보는 것처럼 기본 XML 뷰 트리는 XQuery의 엘리먼트 생성자로부터 만들어지는 엘리먼트의 정보를



(그림 5) 가상적인 XQuery와 대응되는 기본 XML 뷰 트리

저장하는 노드와 FLWR(For-Let-Where-Return) 표현식의 질의 정보를 저장하는 노드로 구성된다. 엘리먼트 노드는 엘리먼트의 이름과 데이터 타입, 그리고 엘리먼트의 텍스트 데이터와 연결되는 경로식이 저장된다. FLWR 노드에는 FLWR절의 변수와 프레디키트, 그리고 정렬 조건에 대한 정보가 저장된다. FLWR 노드의 하위에는 FLWR절에 의해서 구성되는 엘리먼트 노드들이 추가된다.

웹 데이터와 응용 프로그램의 경우는 3.2와 3.2에서 정의한 뷰 정의 모델을 기반으로 기본 XML 뷰 트리를 구성한다. 이때 XML 문서로부터 XML 스키마를 도출해야 하는데 이는 현재 많은 연구가 되고 있는 주제이다. 본 연구에서는 XML 문서로부터 XML 스키마를 수작업으로 구성한 다음, XML 스키마로부터 뷰 트리를 생성시키는 도구를 개발하였다. XML 스키마로부터 뷰 트리를 생성시키는 알고리즘은 여기서는 생략한다.

4.2 사용자 정의 XML 뷰 트리와 질의 트리 생성

사용자 정의 XML 뷰와 사용자 질의는 XQuery로 정의되기 때문에 사용자 정의 XML 뷰 트리와 질의 트리는 XQuery로부터 생성된다. 주어진 XQuery로부터 뷰 트리를 생성하기 위해 XML 문서의 SAX[13] 파서와 유사한 역할을 하는 이벤트 기반의 XQuery 파서를 설계하였다. 이것은 XQuery에서 FLWR 표현식의 FOR, LET, WHERE, RETURN, SORTBY 절과 엘리먼트 생성자가 나타날 때마다 각각에 대해 특정 이벤트를 발생시켜준다. 뷰 트리 생성기는 이러한 이벤트를 감지하여 뷰 트리를 구성하는 일련의 연산을 수행한다. (그림 6)은 이벤트 기반의 XQuery 파서를 이용하여 뷰 트리를 생성하는 알고리즘이다.

(그림 8)은 (그림 4)에서 정의된 사용자 정의 XML 뷰에 대하여 생성된 뷰 트리이다. (그림 8)의 아랫부분은 (그림 5)의 기본 XML 뷰 트리이고, 윗부분은 (그림 4)에서 제시한 사용자 정의 XML 뷰를 트리로 구성한 것이다. (그림 8)에서 보는 것처럼 (그림 4)의 XQuery에서 처음에 나타나는 엘리먼트 생성자인 <PatientGel>에 의하여 뷰 트리에 이름이 PatientGel인 엘리먼트가 추가되고, 그 하위에 나타나는 FLWR 표현식에 의해 FLWR 노드가 추가된다. FLWR 노드에는 변수와 프레디키트 정보들이 함께 저장된다. 계속해서 FLWR 노드의 하위에는 RETURN 절에 나타나는 엘리먼트 생성자와 FLWR 표현식에 의해 해당 노드들이 추가된다. (그림 8)에서 사용자 정의 XML 뷰 트리의 노드에서 아래의 기본 XML 뷰 트리의 노드들로 연결된 선은 합성을 대비한 것인데, 그 과정은 다음 장에서 논한다.

```
generateViewTree(event) {
    switch (event) {
        case: Element Constructor occurs then
            add an element node to the view tree;

        case: FOR or LET clauses occur then
            add a FLWR node to the view tree;

        case: WHERE clause occurs then
            add a predicate to the related FLWR node;

        case: SORTBY clause occurs then
            add an order condition to the related FLWR node;

        case: Path Expression occurs then
            trace the view tree according to the path expression
            and get the bound node;
    }
}
```

(그림 6) 이벤트 기반 파서를 이용한 XML 뷰 트리 생성 알고리즘

5. 사용자 질의와 XML 뷰의 합성

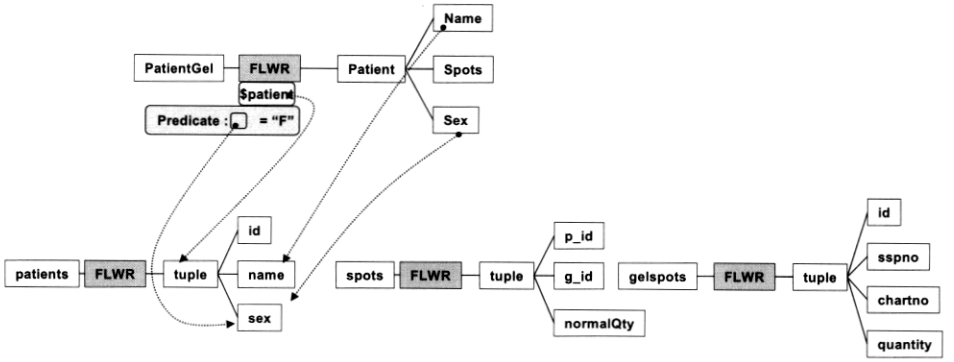
사용자 질의가 사용자 정의 XML 뷰를 바탕으로 이루어졌을 때, 이는 결국 기본 XML 뷰에 대한 질의로 변환해야 되기 때문에 합성 과정이 필요하다. 이를 위해서는 사용자 정의 XML 뷰 트리를 생성할 때 사용자 정의 뷰와 기본 XML 뷰 트리 사이의 관계를 미리 구성한다. 이를 위하여 사용자 정의 뷰 트리에 노드가 추가될 때 그 노드에 연관된 경로식에 대한 탐색이 이루어지고, 탐색 결과로 바인딩 된 노드에 대한 포인터가 유지된다. 이와 같은 과정은 사용자 정의 뷰 트리가 구성되는 과정에서 동시에 이루어진다.

(그림 7)은 (그림 4)의 사용자 정의 뷰에 대한 트리를 구성하면서 (그림 5)의 기본 뷰 트리와 사상되는 모습의 일부이다. (그림 7)의 사용자 정의 뷰 트리에서 변수 $patient의 경우에 주어진 경로식 document("Patient")/tuple을 기본 XML 뷰 트리에서 탐색하게 되고, 그 결과로서 기본 XML 뷰 트리의 tuple 노드에 대한 포인터로서 관계를 유지하는 것이다. 이렇게 변수에 대해 바인딩된 노드는 향후 변수로부터 출발하는 경로식의 항해에서 출발 지점으로 활용된다. 예를 들어 프레디키트 $patient/Sex = "F"의 경우에 $patient에 바인딩된 노드인 tuple 노드에서 항해가 시작된다. 엘리먼트이 값을 나타내는 경로식인 $patient/name/text() 역시 항해를 통해 기본 XML 뷰 트리의 name 노드와 바인딩 된다.

계속해서 (그림 8)은 Spots 엘리먼트 노드의 하위 노드들이 생성되고 기본 XML 뷰 트리에 대해서 사상된 결과이다. (그림 8)에서 보는 것처럼 Spots 엘리먼트 노드의 하위에 있는 FLWR 노드에는 변수 $gel과 $spot이 기본 XML 뷰 트리의 노드와 연결되며, WHERE 절의 조인에 사용된 경로식 역시 탐색을 통해 기본 XML 뷰 트리의 노드에 연결된다. 이 때, 조인에 사용된 경로식의 변수 $patient가 상위 FLWR 절에서 정의된 변수이므로 하위 FLWR 노드는 상위 FLWR 노드와 연관된다. 새로 추가된 FLWR 노드의 하위에는 RETURN 절에 나타난 엘리먼트 생성자에 의해서 해당 엘리먼트들이 추가되어 있음을 확인할 수 있다.

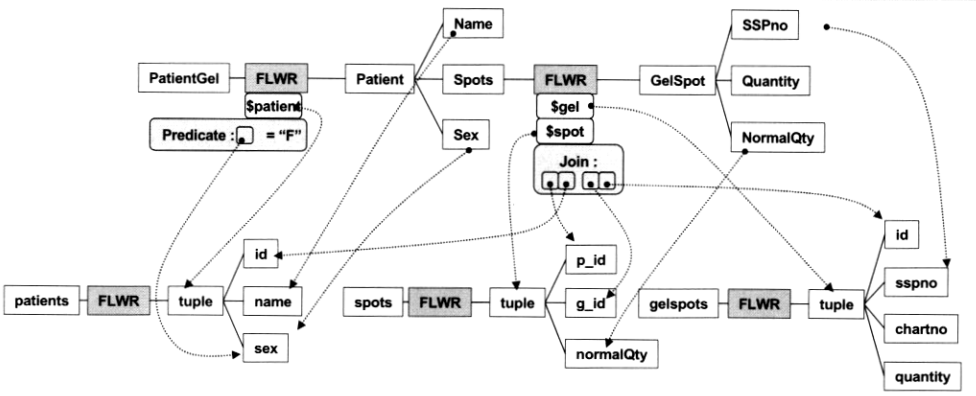
뷰 트리에 대한 XQuery 경로식의 항해는 뷰 트리가 가상적인 XML 문서의 구조를 유지하기 때문에 뷰를 실체화하지 않더라도 가능하다. 원칙적으로 XQuery에 나타나는 경로식에 대한 항해는 실체화된 XML 문서에 대해 이루어

```
<PatientGel> {
  FOR $patient IN document("Patients")/tuple
  WHERE $patient/Sex = "F"
  RETURN
   <Patient ID=$patient/id/text()>
     <Name> $patient/name/text() </Name>
     <Spots> { ... } </Spots>
     <Sex> $patient/sex/text() </Sex>
   </Patient>
}
</PatientGel>
```



(그림 7) 사용자 정의 XML 뷰와 기본 XML 뷰간의 사상 과정 1

```
<Spots> {
      FOR $gel IN document("GelSPots")/tuple,
          $spot IN document("Spots")/tuple
      WHERE $spot/p_id = $patient/id AND $spot/g_id = $gel/id
      RETURN
        <GelSpot>
          <SSPno> $gel/sspno/text() </SSPno>
          <Quantity> $gel/quantity/text() </Quantity>
          <NormalQty> $spot/normalQty/text() </NormalQty>
        </GelSpot>
    }
</Spots>
```



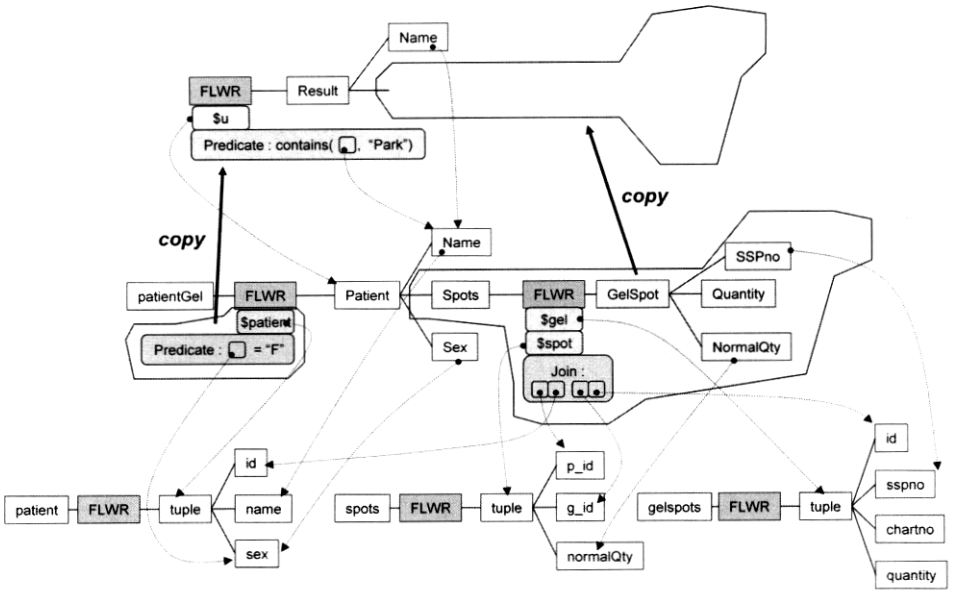
(그림 8) 사용자 정의 XML 뷰와 기본 XML 뷰간의 사상 과정 2

지는 것이지만, XML 뷰 트리는 가상적인 XML 문서의 메타 데이터이기 때문에 스키마 수준에서 탐색이 이루어지는 것이다. 결국, 뷰 트리에 대한 탐색의 결과는 실체화된 XML 문서에 포함된 엘리먼트의 메타 데이터가 된다.

(그림 9)는 XQuery로 표현된 사용자 질의의 예이다. 이 질의는 (그림 4)의 사용자 정의 XML 뷰를 기반으로 작성한 질의이며, 사용자 정의 XML 뷰가 PatientGel 이라는 이름으로 정의되어 있을 때, 문자열 "Park"이 포함된 이름을 가지는 사람들의 이름과 단백질 발현량 목록을 반환하는 질의이다. (그림 9)의 6번째 행의 생략된 경로식은 해당 엘리먼트와 그 하위 엘리먼트를 모두 질의 결과에 포함시킨다.

```
1 : FOR $p IN document("PatientGel")/Patient
2 : WHERE contains($p/Name, "Park")
3 : RETURN
4 :       <Result>
5 :               <Name> $p/Name/text() </Name>
6 :               { $p/Spots }
7 :       </Result>
```

(그림 9) 사용자 질의



(그림 10) 사용자 질의와 기본 XML 뷰의 합성

(그림 10)은 (그림 9)의 사용자 질의를 뷰 트리로 구성하고, 사용자 정의 XML 뷰 트리를 통해 기본 XML 뷰 트리에 대해서 합성한 결과이다. 사용자 질의에 나타난 경로식은 사용자 정의 XML 뷰 트리에 대한 포인터로 유지된다. (그림 9)의 질의에서 6번째 행의 숨은 경로식은 해당 엘리먼트와 그 하위 엘리먼트를 모두 질의 결과에 포함시키는 것이므로 경로식에 바인딩된 노드와 그 하위 노드를 복사하여 트리에 추가한다. (그림 10)에서 사용자 질의의 변수 $p가 사용자 정의 XML 뷰 트리의 Patient 엘리먼트 노드에 바인딩되고 이 엘리먼트 노드는 사용자 정의 XML 뷰의 $patient로부터 파생되므로 $p는 $patient의 프레디키트에 영향을 받는다. 이것을 질의 트리에 반영하기 위하여 $patient의 프레디키트를 질의 트리로 복사하여 $p의 프레디키트에 추가한다. 즉, 하위 뷰의 프레디키트를 상위의 질의로 끌어올린 것이다.

본 논문에서 사용된 합성 메커니즘은 합성의 결과로 모든 질의 정보가 최상위의 질의 트리에 모이게 된다. 반대로 상위 질의의 프레디키트를 하위 뷰로 내려 보내어 질의 정보를 하위의 뷰로 모으는 방법도 가능한데, 이 경우 최종 질의에 포함되지 않는 뷰의 요소를 합성된 뷰 트리로부터 제거해야 하며, 뷰 트리를 다른 질의와 공유할 수 없는 단점이 있다. 본 논문의 방법은 질의에 의해 구성되는 엘리먼트 노드들만 끌어 올려지기 때문에 질의에 필요하지 않은 뷰의 요소는 처음부터 포함되지 않는다. 그리고 하위의 뷰 트리는 다른 상위의 질의들과 공유될 수 있다.

(그림 10)에서 보는 것처럼 합성의 결과, 질의 트리는 노드간의 포인터 연결을 통해 기본 XML 뷰 트리에 연결되어서 기본 XML 뷰에 대한 질의가 된다. 이렇게 합성된 질의 트리는 스키마 생성, 질의 변환, 결과 문서 생성에 이용된다.

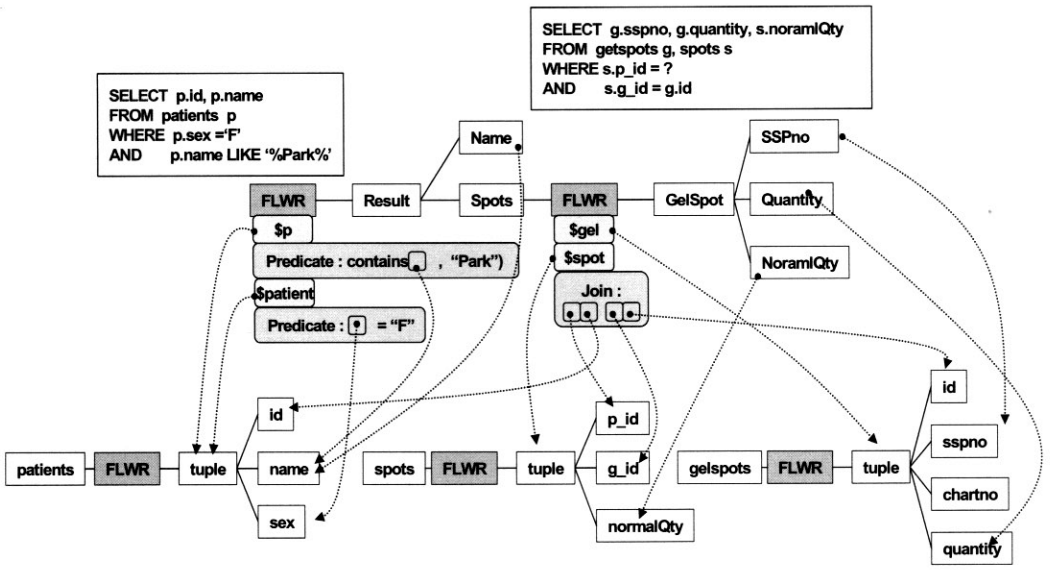
경로식의 탐색과 뷰 트리의 노드간의 포인터 연결을 이용하는 합성 메커니즘은 포인터 연결을 따라 쉽게 기본 XML 뷰의 정보에 접근이 가능하며, 더 높은 추상화를 위하여 사용자 정의 XML 뷰에 대해 정의된 XML 뷰에 대한 합성도 동일한 방법으로 이루어지므로 뷰 정의를 쉽게 지원한다. 또한, 사용자 정의 XML 뷰와 기본 XML 뷰에 대한 질의를 하나의 질의에서 표현하는 것과 같이 여러 단계에서 정의된 XML 뷰에 대해서 그 레벨에 관계없이 질의가 가능하다. 그리고 잘못 작성된 경로식에 대한 검출을 쉽게 만든다.

6. 질의 변환

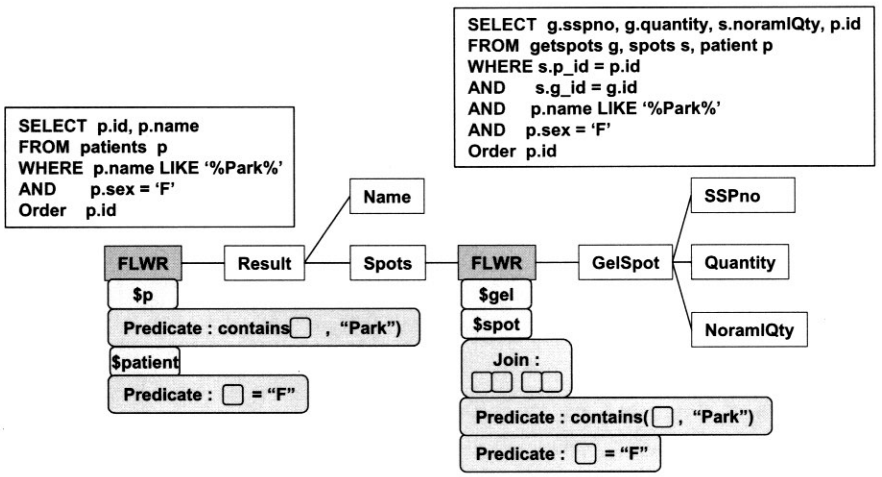
합성된 질의는 지역 정보원의 질의로 변환되어 정보원 엔진에 의해 수행된다. 예를 들어 관계형 데이터베이스의 경우 SQL로 변환하는 과정이 필요하며, 응용 프로그램일 경우 명령어 라인으로, 그리고 웹 자원일 경우 URL로 변환하는 과정이 필요하다. 그러나 이는 변환될 질의의 형태만 다를 뿐 개념적으로는 동일한 과정으로 질의 변환이 가능하다.

6.1 관계형 데이터베이스에 대한 질의변환

본 절에서는 관계형 데이터베이스 질의어인 SQL로 변환하는 과정을 제시한다. 먼저 합성된 질의 트리를 순회하며 변환에 필요한 정보들을 FLWR 노드에 모으고, 이를 이용하여 정보원에서 수행될 질의어 형식에 맞게 변환하다. 관계형 데이터베이스의 경우 SQL의 SELECT절과 FROM절을 구성하는 정보는 FLWR 노드의 하위 엘리먼트 노드에 바인딩된 기본 XML 뷰 트리의 노드로부터 얻고, 관련된 프레디키트는 SQL의 WHERE절을 구성한다. (그림 11)은 합성된 질의 트리를 기반으로 생성된 SQL 질의이다. 그림에서 질의 트리와 기본 XML 뷰 트리 사이의 사용자 정의 XML 뷰 트리는 생략되었다. (그림 11)에서 보는 것처럼 각각이 FLWR 노드에 연관된 질의 정보가 SQL로 번역된다.



(그림 11) 호스트 변수를 이용한 질의 변환



(그림 12) 디코릴레이션된 질의 트리와 생성된 SQL

(그림 11)에서 상위의 FLWR 노드와 하위의 FLWR 노드는 서로 조인관계이다. 이렇게 조인된 질의를 처리하는 가장 직관적인 방법은 (그림 11)에서 번역된 SQL과 같이 호스트 변수를 사용하는 것이다. 이 방법은 상위의 SQL을 수행시키고 그 결과 튜플을 하나씩 하위 SQL의 호스트 변수에 적용시킴으로서 조인 연산을 수행하게 한다. 결국, 조인 연산을 랩퍼에서 수행하게 하며, 하위의 SQL이 여러 번 수행하게 함으로서 성능의 저하를 초래할 수 있다. 물론 2개의 SQL을 하나의 SQL로 합쳐서 수행시킬 수도 있는데, 이 때는 중복되는 데이터로 인해 결과 문서를 생성하는데 어려움이 있다.

이러한 성능 저하를 해소하기 위해서는 가능하면 지역 데이터베이스의 질의 처리 능력을 최대한 활용하는 것이 중요하다. 관계형 데이터베이스에서 조인으로 상호 연관된 질의를 디코릴레이션 시켜서 독립적으로 수행하게 하는 것은 널리 알려진 방법이다. 본 논문에서 이것은 상위의 FLWR 노드의 프레디키트를 하위의 FLWR 노드로 내려 보내고 호스트 변수를 조인으로 대체함으로서 이루어진다. 이때, 태거에서 결과 XML 문서를 수월하게 구성할 수 있도록 질의를 변환하는 것 또한 중요한데, 이를 위해서 연관된 SQL 질의를 같은 조건으로 정렬한다.

(그림 12)는 디코릴레이션된 질의 트리와 이를 기반으로 생성된 SQL이다. 그림에서 상위 FLWR 노드의 프레디키트가 하위 FLWR 노드로 내려 보내진 것을 확인할 수 있다. 그리고 하위 FLWR 노드에 생성된 SQL에 상위에서 받은 프레디키트가 반영되어 있고, 두 개의 SQL은 같은 정렬 조건을 가지고 있다. 두 개의 SQL은 각각 독립적으로 수행되고, 결과 튜플이 같은 조건으로 정렬되어 있으므로, 태거에 의해 단일 패스로 XML 문서로 변환될 수 있다.

6.2 웹 자원에 대한 질의변환

웹 자원의 경우, 변환될 질의 형식이 SQL이 아니라 URL

```
1 : <Result> {
2 : FOR $n IN document("Nucleotide")//GBSeq
3 : WHERE $n/GBSeq_primary_accession="X60070"
4 :     OR $n/GBSeq_primary_accession="AF164272"
5 : RETURN
6 :     <GBRef>
7 :         $n/GBSeq_references
8 :     </GBRef>
9 : </Result>
```

(그림 13) Entrez 웹 자원에 대한 사용자 질의

형식이라는 것 이외는 관계형과 동일한 과정으로 질의변환이 가능하다. 제시된 모델은 정보원에서 요구하는 질의어를 구성하는 부분만 정보원에 따라 다를 뿐이고 질의어가 변환되는 메커니즘은 정보원의 성격에 관계없이 동일하게 수행될 수 있다.

예를 들어 웹 자원인 Entrez[4]에 대한 질의 변환을 생각해보자. (그림 13)은 Entrez 랩퍼에 전달된 XQuery로 작성된 사용자 질의이다. 사용자 질의를 처리하기 위해서 랩퍼에서는 내부적으로 질의 트리를 생성하고 동시에 Entrez의 기본 XML 뷰 트리에 대한 합성을 수행한다. 합성된 질의 트리는 태거에 의해서 루트부터 순회를 시작한다. 이때 FLWR 노드를 방문하게 되면 질의에 필요한 정보를 모으게 되고, 이를 바탕으로 Entrez에서 수행될 URL을 구성하게 된다.

(그림 14)는 합성된 질의 트리로부터 URL형식의 질의어로 변환하는 과정을 나타낸다. 우선 기본 URL이 명시되고, FOR절에 바인딩된 XML 문서의 이름을 통해 Entrez에서 검색할 데이터베이스 이름이 정해진다. 그리고 관련된 프레디키트는 Entrez의 URL에거 제공하는 파라미터에 맞게 사상시켜 완전한 하나의 URL을 구성한다.

이와 같이 정보원에서 수행될 질의어로 변환하는 과정은 합성된 질의 트리를 순회하면서 FLWR 노드를 만나게 되면 각 정보원에서 수행하게 될 질의 형식을 구성하고 변환된 질의를 수행하면 된다. 정보원이 BLAST[3]와 같은 응용 프로그램일 경우도 마찬가지이다. 즉, 정보원의 이질적인 특징에 상관없이 뷰 트리를 이용한 질의 변환 메커니즘은 동일하다.

7. 결과 문서 생성

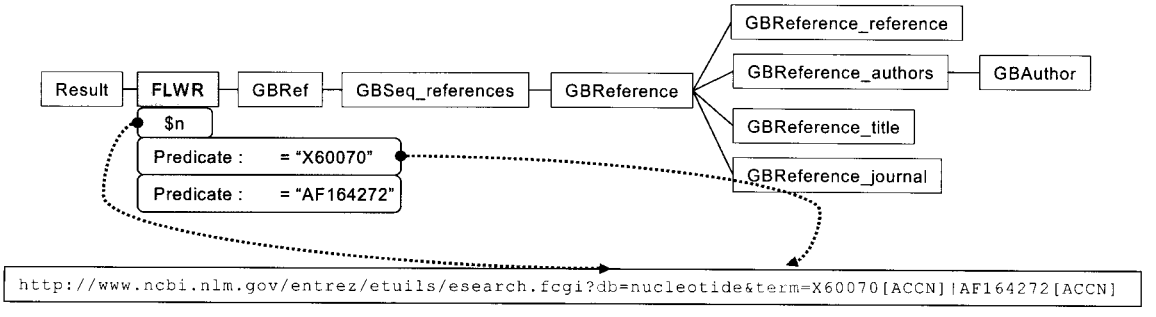
질의 트리가 결과 XML 문서의 템플릿 역할을 하기 때문에 태거가 결과 XML 문서를 생성하는 것은 직관적이다. 태거가 질의 트리를 순회하면서 방문하는 노드가 FLWR 노드인 경우에는 정보원에서 수행될 질의어를 만들고, 변환된 질의어를 정보원에서 수행될 수 있도록 한다. 그리고, 정보원에서 수행된 결과 집합의 개수만큼 하위 노드를 순회한다. 태거가 방문하는 노드가 엘리먼트 노드일 경우에는 상위의 FLWR 노드에서 실행될 질의 결과를 적용하여 XML 문서를 구성한다.

7.1 관계형 튜플로부터 결과문서 생성

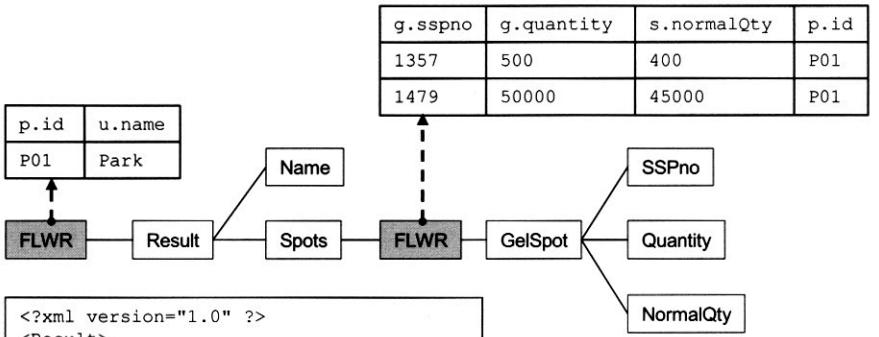
(그림 15)는 정보원이 관계형 데이터베이스일 경우 (그림 12)에서 생성된 질의를 수행한 결과와 태거에 의해서 생성된 결과 XML 문서를 나타낸다. (그림 15)에서 최상위 노드가 FLWR 노드이므로 변환된 SQL을 실행시키고, 결과 집합이 하나의 튜플을 가지고 있으므로 하위 노드를 한번 순회하게 된다. 따라서 하위의 Result와 Name, 그리고 Spots 엘리먼트가 차례대로 결과 XML 문서에 추가되는데, Name 엘리먼트의 경우에는 FLWR 노드의 결과 튜플이 적용된다.

Spots 엘리먼트 노드의 하위 FLWR 노드는 상위의 FLWR 노드와 조인 연산이 적용되었으므로 결과 집합에 속한 튜플 중 현재 적용중인 상위 FLWR 노드의 결과 튜플과 연관된 튜플들만 하위의 노드에 적용된다. 결국, 두 연관된 결과 집합이 외부 조인이 이루어진 것인데, 각각의 결과 레코드 셋이 정렬되어 있기 때문에 싱글 패스로 결과 XML 문서를 생성할 수 있다. (그림 15)에서 하위 FLWR 노드의 결과 집합에 속한 두 개의 튜플이 모두 현재 적용중인 상위 FLWR 노드의 결과 튜플과 연관되므로, GelSpot 엘리먼트 노드 이하는 두 번 순회하게 된다.

이와 같은 질의 변환 및 결과 문서 생성 메커니즘은 결과적으로 XPERANTO 연구에서 이용된 Sorted Outer Union(SOU)[10] 방법과 유사한 면이 있으나, SOU는 연관된 질의를 디코릴레이션하여 정렬하고 이것을 외부 조인으로 합치는 하나의 SQL 질의를 구성하여 관계형 데이터베이스 엔진에서 수행하게 하는 방법인데 결과 튜플의 길이가 길어지는 단점이 있다. 본 논문의 방법은 디코릴레이션된 질의가 각각 독립적으로 관계형 데이터베이스에서 수행되고 태거에 의해서 외부 조인이 이루어지며, 각각의 결과 튜플에는 관련된 컬럼들만 포함되게 된다.



(그림 14) 합성된 질의 트리로부터 URL 형식으로 변환하는 과정




```
<?xml version="1.0" ?>
<Result>
    <Name> Park </Name>
    <Spots>
        <GelSpot>
            <SSPno>1357</SSPno>
            <Quantity>500</Quantity>
            <NormalQty>400</NormalQty>
        </GelSpot>
        <GelSpot>
            <SSPno>Helicopter</SSPno>
            <Quantity>50000</Quantity>
            <NormalQty>45000</NormalQty>
       </GelSpot>
    </Spots>
</Result>
```

(그림 15) SQL 수행 결과와 변환된 XML 문서

아울러, 생성되는 결과 XML 문서의 크기가 큰 경우를 대비하여 커서를 이용할 수 있도록 한다. 변환된 질의 수행 결과로 얻은 결과 집합 전체를 한번에 하나의 XML 문서로 구성하지 않고, 결과 집합의 튜플을 하나씩 XML 단편으로 구성하여 커서에 바인딩시킨다. 그런데, XML 문서의 임의의 내포구조는 커서의 범위를 설정하는데 혼란을 야기하기 때문에, 본 논문에서는 XQuery에서 최상위 수준의 FLWR 절, 즉 질의 트리에서 최상위의 FLWR 노드의 결과 집합을 커서에 바인딩 시킨다. (그림 15)의 결과 문서의 경우에는 하나의 XML 단편이 커서에 바인딩된다. 만약 최상위 FLWR 노드의 결과 집합이 3개의 튜플을 가졌다면 3개의 XML 단편이 커서에 바인딩 될 것이다.

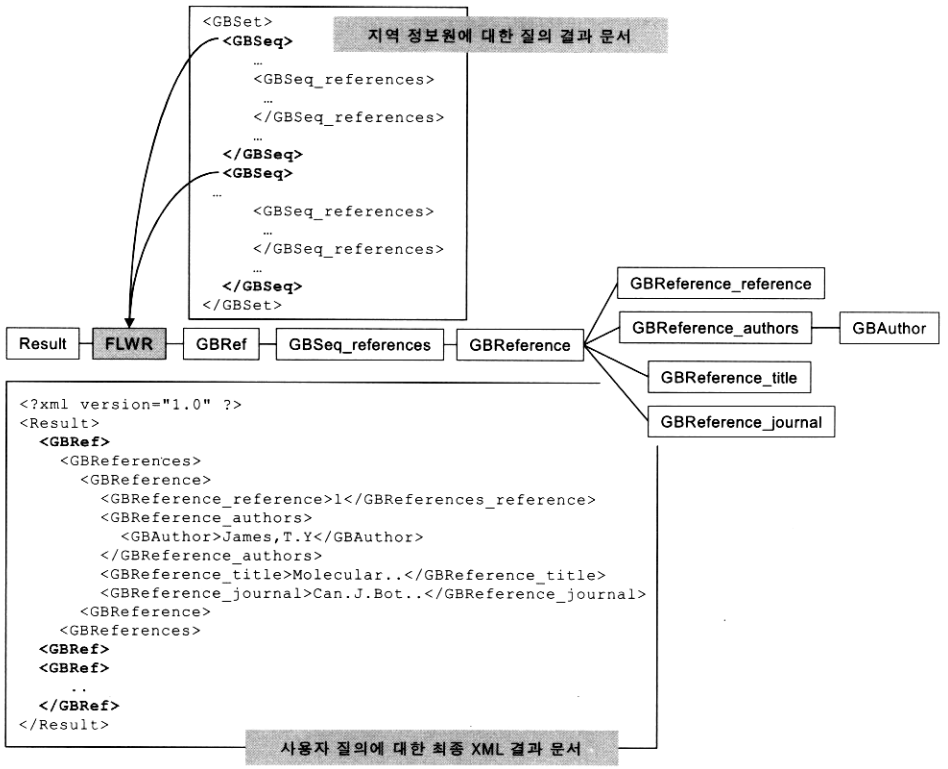
7.2 XML 문서로부터 결과문서 생성

정보원에 대한 질의 결과의 형이 관계형 튜플이 아닌 XML 문서인 경우는, XML 문서로부터 XQuery에서 정의된 사용자가 원하는 형태의 XML 문서를 생성해야 한다. 관계형 데이터베이스의 튜플은 이차원 구조인 반면에 XML 문서는 임의의 계층구조를 가지기 때문에 이 계층구조로부터 질의결과를 생성하기 위해서 본 논문에서는 DOM 파서를 이용한다. 태거는 DOM 파서에 의해서 생성된 DOM 트리의 자료구조를 이용하여 사용자가 원하는 정보를 탐색하고, 해당하는 정보를 가지고 와서 최종적인 XML 문서를 생성한다.

(그림 16)은 질의 결과의 형이 XML 문서인 정보원 Entrez를 이용하여 앞에서 제시한 질의처리 모델을 적용하는 과정을 예시한 것이다. 그림의 가운데에 있는 질의 트리는 (그림 13)에서 기술한 XQuery 질의어를 바탕으로 생성된 것이다. (그림 13)의 7번째 줄에서 나타나는 생략된 경로식 $n/GBSeq_references는 해당 엘리먼트와 함께 그 하위 엘리먼트를 모두 포함시키는 것이므로, 질의 트리에서 GBSeq_references 엘리먼트와 그 하위 엘리먼트의 구조가 반영되어 있다.

(그림 16)에서 최상위 노드가 엘리먼트 노드이기 때문에, 결과 XML 문서에 <Result> 엘리먼트를 만든다. 계속해서 순회하게 될 노드는 FLWR 노드이므로 (그림 14)의 URL을 생성하여 Entrez에 검색질의를 보낸다. 관계형 데이터베이스의 경우 튜플이 변수에 바인딩되지만, XML 문서를 반환하는 웹 자원일 경우 FOR절에서 명시한 엘리먼트가 변수에 바인딩된다. (그림 16)에서, Entrez에서 전달된 XML 문서에서 GBSeq 엘리먼트가 두 번 나오기 때문에 FLWR 노드의 하위 노드들을 차례대로 두 번 순회한다. 태거는 계속 하위 엘리먼트를 방문하여 GBRef, GBReferences, GBReference 엘리먼트가 차례대로 결과 XML 문서에 추가된다. GBReference_reference 엘리먼트의 경우에는 값을 가지는 단말 엘리먼트이기 때문에 변수에 바인딩된 첫 번째 엘리먼트인 GBSeq 엘리먼트 하위에 있는 GBReference_reference를 찾아 해당 값을 문서에 적용시킨다. 그 결과 GBReference_journal 엘리먼트까지 문서를 생성하면, FLWR노드에 바인딩된 두 번째 엘리먼트에 대해서 같은 과정을 반복하여 최종적으로 사용자가 요구하는 형태인 XML 문서가 생성된다.

이와 같이 정보원에서 수행된 결과의 양식이 다양하더라도 태거가 합성된 질의트리를 순회하면서 XML 문서를 생성해 나가는 메커니즘은 동일하다. 즉, 방문하는 노드가 FLWR 노드인 경우 질의를 변환하고, 변환된 질의를 이용하여 정보원에서 수행하고, 질의트리에서 방문하는 노드가 엘리먼트 노드일 경우 정보원에서 수행된 결과를 이용하여 사용자가 원하는 형태의 최종 XML 문서를 구성한다.



(그림 16) URL 수행 결과와 변환된 XML 문서

8. 성능 평가

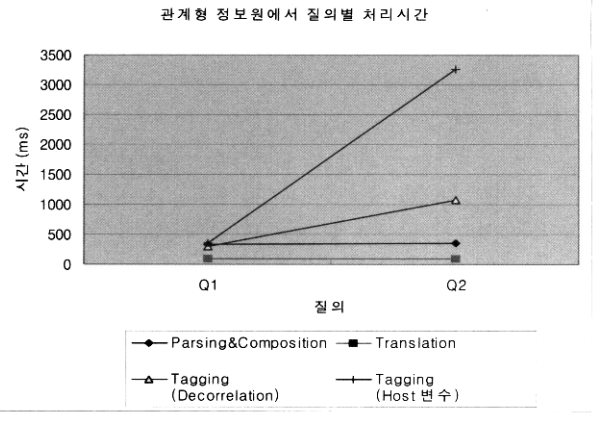
다음은 제시된 통합 시스템에서의 질의처리 성능을 평가한다. 실험에 사용된 정보원으로는 생물학적 실험정보를 저장하고 있는 관계형 모델, 서열 분석 응용 프로그램인 BLAST, 웹 자원인 Entrez, 그리고 PDB이며, 각 랩퍼에서 질의를 처리하는 시간을 기능별로 나누어서 비교한다. 질의처리시간은 크게 3개의 부분으로 나누어지는데, 첫 번째는 XQuery 형태의 사용자 질의를 입력받아 파싱하여 질의 트리를 구성하고, 기본 XML 뷰 트리에 대한 합성을 수행하는데 걸리는 시간이다. 두 번째는 정보원에 대한 질의로 번역하는데 걸리는 시간, 마지막으로 정보원에서 수행된 결과가 랩퍼로 도착하였을 경우, 사용자의 질의에서 요구하는 형태로 문서를 재작성하는 데에 걸리는 태깅(Tagging) 시간이 있다.

8.1 관계형 모델

본 질의처리엔진의 성능을 테스트하기 위해 관계형 모델에서 3개의 테이블을 가지는 생물 정보 데이터베이스에 대해 적은 양의 결과를 반환하는 질의 Q1과 상대적으로 많은 양의 결과를 반환하는 서로 다른 질의 Q2를 임의로 작성하여 실행시켰다. 각 질의에는 조인으로 연관되는 내포된 FLWR(For-Let-Where-Return) 절을 가지고 있다. 내포된 FLWR절이라 함은 (그림 4)의 사용자 정의 뷰의 예에서 나타난 것처럼 FLWR절 안에 또 하나의 FLWR절이 중첩되어 있는 구조를 의미한다. 또한 질의 변환에 있어서 호스트 변수를 이용하는 방법과 디코릴레이션을 이용하는 방법을 모두 테스트 하였다. <표 1>은 두 질의를 여러 번 수행하여 걸린 평균 시간을 제시한 것인데, 질의를 파싱하고 합성하는데 걸리는 시간, SQL로 변환하는데 걸리는 시간, 그리고 변환된 질의를 수행하여 결과 문서를 생성하는데 걸린 시간을 측정한 결과이다. 시간의 단위는 1/1000초(milli-second) 이다.

〈표 1〉 관계형 모델에서의 질의 수행 시간 및 결과 문서 크기

질의	Parsing& Composition	Translation	Tagging		결과 문서 크기
			Decorrelation	Host 변수	
Q1	337	10	290	350	46 KB
Q2	348	10	1065	3265	3650 KB



(그림 17) 관계형 모델에서의 질의별 성능 비교

<표 1>과 그래프로 표현한 (그림 17)의 질의 수행 결과를 보면 사용자 질의를 파싱해서 기본 XML 뷰에 대한 합성을 하는데 걸리는 시간과 SQL로 변화하는 시간은 질의에 관계없이 거의 일정함을 알 수 있다. 그러나 결과 문서를 생성하는 부분에서 결과 문서의 크기가 수행 시간에 영향을 미치는데, 호스트 변수를 사용하는 경우와 디코릴레이션을 사용하는 경우를 비교하면 디코릴레이션을 사용하는 방법이 더 나은 결과를 보여주고 있으며, 두 번째 질의의 경우에 3.5MB 정도의 결과 문서를 생성하는데 1초 정도의 만족할 만한 성능 보여 주고 있다. 그리고 예상대로 호스트 변수를 사용하는 경우는 결과 문서의 크기가 커질수록 수행 시간이 더 크게 증가한다.

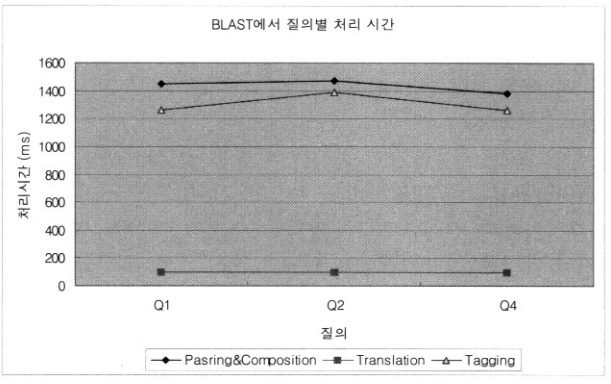
8.2 응용 프로그램(BLAST)

응용프로그램 BLAST의 랩퍼에 대한 성능평가에 사용된 질의는 "주어진 단백질 서열로부터 유사도 점수(score)가 100 이상인 단백질의 ID값과 score 값을 구하라" 와 같이 하나의 XQuery FLWR 절로써 표현이 되는 두 개의 질의 Q1, Q2과, 중첩된 FLWR 절을 필요로 하는 질의 Q3를 임의로 작성하여 성능을 분석하였는데 그 결과는 <표 2>와 (그림 18)과 같다.

<표 2>와 (그림 18)의 결과를 보면, 질의 유형에 상관없이 질의를 처리하는 시간이 비교적 일정하게 나타나는 것을 알 수 있다. 그런데, 결과 문서의 크기가 커질수록 Tagging 시간이 길어질 것으로 기대되는데, 제시된 실험결과에서는 질의처리 시간이 결과문서의 크기에 크게 의존적이지는 않음을 알 수 있다. 이는 정보원에서 랩퍼로 전달되어 오는 결과 문서가 XML 문서인 경우, 앞 절에서 언급한 바와 같이, 전달된 XML 문서를 DOM 트리로 변환하여 이를 순회하면서 질의트리에서 주어진 템플릿에 맞게 XML 문서를 생성하기 때문에, DOM 트리의 구조와 탐색시간이 전체 수행시간에 영향을 미치기 때문인 것으로 판단된다.

〈표 2〉 응용 프로그램(BLAST) 에서의 질의 수행 시간 및 결과 문서 크기

질의	Parsing& Composition	Translation	Tagging	결과 문서 크기
Q1	1451	10	1262	431KB
Q2	1469	10	1389	1275 KB
Q3	1385	10	1261	1604 KB



(그림 18) 응용 프로그램(BLAST) 에서의 질의별 성능 비교

8.3 웹 자원(Entrez, PDB)

다음은 웹 정보원인 Entrez가 지원하는 NCBI 정보원과 PDB에 대하여 수행시킨 질의는 <표 3>과 <표 4>와 같다. 그리고 이 질의에 대한 수행 결과는 <표 5>와 <그림 19>와 같다.

〈표 3〉 Entrez의 PubMed에 대한 질의

질의	질의내용
Q1	PubMed의 ID가 15840428, 15840413, 15839582, 15839348인 문헌 정보의 Abtract를 검색하시오.
Q2	저자의 이름이 "Maddess"이면서 국적이 영국인 문헌정보의 ID 번호와 문헌 제목을 검색하시오.
Q3	저자의 이름이 "Maddess"이면서, 2000년 이후로 발간된 논문의 ID, 문헌 제목, 그리고 Abstract를 검색하시오.
Q4	PubMed의 ID가 15840766, 15840428, 15840143, 15839582, 15839582인 문헌 중에서 2000년 이후로 발간된 논문의 ID, 문헌 제목을 검색하시오. 단, "Maddess"라는 저자가 포함되어 있는 논문에 한해서만 저자의 정보를 출력하시오.

〈표 4〉 PDB에 대한 질의

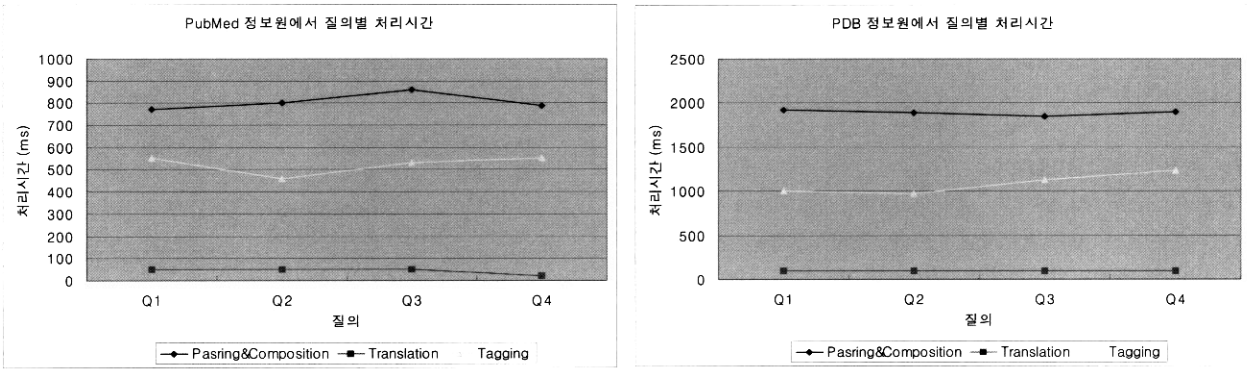
질의	질의내용
Q1	chemicalName이 "N-BUTYL ISOCYANIDE"인 PDB정보 중에서 해당 저자를 검색하시오.
Q2	다음의 조건 중에서 적어도 하나를 만족하는 실험 기술 정보를 검색하시오. - 분류(classification) = "OXYGEN TRANSPORT" - ID(idCode) = "101M" - chemicalName = "N-BUTYL ISOCYANIDE" - 키워드(keyword) = "HEME"
Q3	"HEME"라는 키워드가 포함되어 있는 PDB정보 중에서 해당 저자를 검색하시오.
Q4	분류명이 "OXYGEN TRANSPORT" 이거나 ID가 "101M"인 정보를 모두 검색하여 해당 제목과 관련 저널정보(저자, 제목, 발행명, 국적)을 검색하시오"

위의 두 정보원에 대한 질의에서 Q1, Q2, Q3는 하나의 FLWR절이 포함되어 있는 XQuery로 작성할 수 있는 반면에, Q4의 경우는 두 개의 중첩 FLWR절이 구성되어 있는 XQuery로 표현된다.

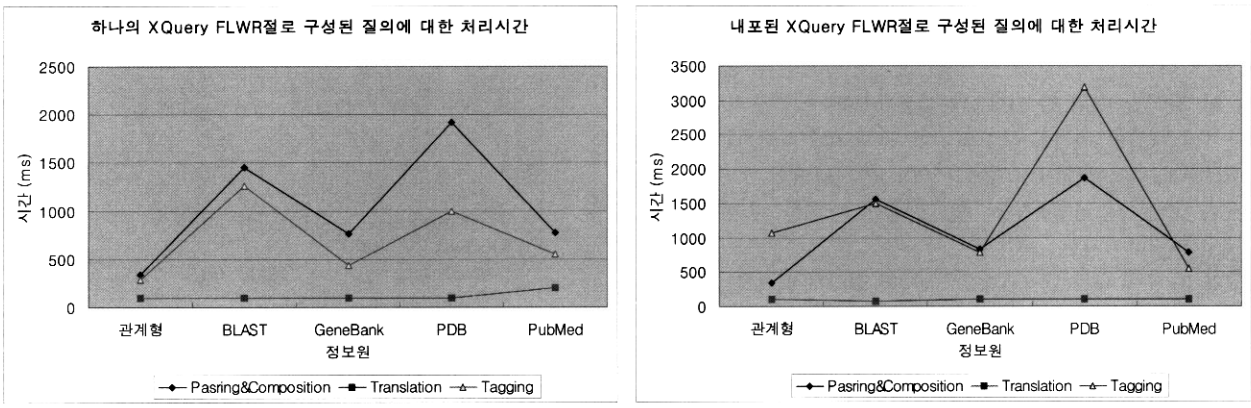
전체적으로, 질의의 복잡성, 결과문서의 크기 등에 크게 영향을 받지 않고 한 정보원에서의 질의처리 시간은 거의 일정하게 걸린다는 것을 알 수 있다. 이는 본 질의처리엔진의 장점 중의 하나이다. 단, PDB의 결과문서 크기가 PubMed에 비하여 평균적으로 더 작음에도 불구하고 경우, Tagging에 걸리는 시간이 더 많이 걸린 이유는, 정보원에서 랩퍼로 전달되는 결과 문서가 PubMed는 XML 문서인 반면에, PDB는 flat file 형태이다. 따라서 PDB 경우는 falt file을 XML 문서로 변환해야 하는데 이에 따르는 시간이 더 걸린 것이다.

〈표 5〉 웹 자원에서의 질의 수행 시간 및 결과 문서 크기

Entrez의 PubMed					PDB				
질의	Parsing& Composition	Translation	Tagging	결과 문서 크기	질의	Parsing& Composition	Translation	Tagging	결과 문서 크기
Q1	772	20	551	5 KB	Q1	1922	10	1000	1 KB
Q2	801	20	461	0.37 KB	Q2	1891	10	969	1 KB
Q3	861	20	530	4 KB	Q3	1843	10	1125	2 KB
Q4	791	10	551	4 KB	Q4	1895	10	1234	9 KB



(그림 19) 웹 자원(Entrez, PDB)에서의 질의별 성능 비교



(그림 20) 정보원별 질의 성능 비교

8.4 정보원별 비교분석

(그림 20)은 각 정보원별로 질의처리에 대한 성능비교를 위하여, 하나의 XQuery FLWR절로 구성된 질의와 두 개의 XQuery FLWR절로 구성된 질의의 평균처리시간을 나타낸 것이다.

우선, 질의 합성/파싱의 처리시간을 살펴보면, 관계형 DB, GeneBank, PubMed의 경우에 비하여 BLAST와 PDB 경우가 처리시간이 더 많이 걸린다. 그 이유는 BLAST와 PDB의 기본 XML 뷰의 크기가 다른 정보원의 기본 XML 뷰보다 훨씬 크고 복잡하기 때문이다. 기본 XML 뷰 트리가 크고 복잡하면 사용자 질의를 사용자 정의 뷰와 합성하기 위한 탐색 또는 순회 시간이 그만큼 커진다.

한편, 각 정보원에서의 파싱/합성 시간을 살펴보면 기본 XML 뷰의 크기가 예외적으로 큰 PDB를 제외하고는 그 처리시간은 1초 내외임을 알 수 있다. 합성과정은 사용자 정의 XML 뷰를 지원하기 때문에 수반되는 것인데, 이는 사용자 정의 XML뷰를 지원함으로써 발생되는 오버헤드는 최대 1초 정도임을 의미한다. 따라서 사용자 정의 뷰의 여러 가지 이점을 고려하면 이정도의 오버헤드는 무시할 수 있는 시간이라고 볼 수 있다.

다음으로, 질의변환에 걸리는 시간은 질의의 복잡도와 정보원에 관계없이 일정하며 그 처리시간이 크지 않기 때문에 전체질의처리 시간에 크게 영향을 미치지 않음을 알 수 있다. 질의변환 오버헤드가 적은 이유는 합성과정에서 이미 질의 변환에 필요한 요소들을 질의트리에 반영해 두었기 때문에 실제 질의변환작업은 비교적 단순하기 때문이다.

마지막으로, 태깅(Tagging)시간은 질의 복잡도와 정보원에서 반환되는 결과 문서의 크기에 영향을 받는다. 특히 질

의에서 조인 연산이 필요할 때, 관계형 모델에서는 디코릴레이션 기법으로써 그 수행시간을 단축시킬 수 있으나, 웹 자원의 경우는 호스트 변수를 사용해야 하기 때문에 이 경우에는 많은 시간이 소요되었다.

한편, 정보원에서 반환되는 결과 문서의 크기가 크면, 관계형의 경우는 튜플수 대로 결과 XML 문서를 만들어야 하므로 수행시간이 튜플 크기에 비례한다. 그러나 웹 자원의 경우는 검색결과가 XML문서인데, 이를 처리하기 위하여 본 논문에서는 DOM 모델을 기반으로 하여 질의 트리와 DOM 트리를 순회하여 결과 XML 문서를 생성해 나간다. 따라서 XML 문서의 크기가 클 경우 트리 순회 오버헤드 및 메모리 오버헤드로 인하여 수행시간이 떨어진다. PDB의 경우는 검색결과가 클 뿐 아니라, 정보원에서 플랫 파일 형식으로 결과를 전달해 주기 때문에 이를 XML 문서로 변환하는 과정이 추가로 더 필요하여 평균수행시간이 가장 많이 소요된다.

9. 결 론

XML 뷰 기반의 랩퍼 시스템은 이질의 분산된 정보원들을 모두 XML 정보원으로 간주하여 사용자로 하여금 XML 정보원처럼 사용할 수 있도록 한다. 이질의 정보원을 XML 정보원처럼 사용하기 위해서는 지역 정보원의 스키마를 XML 뷰로 정의하는 것이 중요하다. 각 지역 정보원의 스키마를 XML 뷰로 변환하여 사용자에게 제공하면, 사용자는 이질의 정보원들을 모두 XML 정보원으로 간주하여 XML 질의어를 이용하여 질의를 하게 됨으로서 이질적인 정보원들의 모델에 독립적인 질의를 가능하게 한다.

본 논문에서는 범용성과 적응력이 뛰어난 생물 정보원 통합 미들웨어을 설계하는데 있어서 사용자 정의 XML 뷰 기반의 질의처리 엔진을 제시하였다. 이를 위해서 각 정보원에 대한 XML 기반의 뷰 정의 모델과 질의 처리 모델을 제시하였다. XML 뷰 기반의 질의 처리 모델은 기본 XML 뷰와 사용자 정의 XML 뷰, 그리고 사용자 질의어에 대한 개념적인 추상화를 통하여 모든 XML 뷰를 동일한 데이터 모델로 표현한 트리 기반의 모델이다. 이를 기반으로 랩퍼에서는 뷰 합성, 지역 정보원에 대한 질의 변환, 그리고 결과 문서 생성 과정을 수행한다. 본 논문에서 제시한 XML 뷰 기반의 질의 처리 모델은 XML 뷰와 사용자 질의가 내부적으로 동일한 트리 모델을 이용하여 표현되기 때문에 구현의 중복을 피할 수 있으며, 뷰 트리의 경로 탐색을 용이하게 함으로서 질의어 합성을 편하게 한다. 그리고, 다단계로 정의되는 사용자 정의 XML 뷰 또한 쉽게 지원할 수 있다.

본 논문에서 제시한 XML 뷰 기반의 질의 처리 엔진은 지역 정보원에 독립적이기 때문에 다른 이질적인 지역 정보원에 대해서도 적용이 가능한 범용적인 모델이다. 또한, 사용자 정의 뷰를 지원함으로서 사용자의 다양한 요구를 수용할 수 있는 융통성 있는 모델이다. 따라서, 본 논문에서 제시한 뷰 정의모델과 트리 모델, 그리고 이를 기반으로 하는 합성 메커니즘은 XML 뷰를 지원하는 랩퍼를 구현하기 위한 프레임워크로 활용될 수 있으며, 미디에이터 기반의 이질의 정보원을 통합하기 위한 미들웨어 시스템을 개발하는데 활용가능하다.

참 고 문 헌

[1] A.S.Kosky, I.A.Chen, V.M.Markowitz, E.Szeto, "Exploring Heterogeneous Biological Databases : Tools and Application", Proc. of the 6th International Conference on Extending Database Technology, pp.3-5, 1998.

[2] B.Eckman, Z.Lacroix, L.Raschid, "Optimized Seamless Integration of Biomolecular Data", Proc. of the IEEEE 2nd International Symposium on Bioinformatics and Bioengineering Conference, pp.1-3, 2001.

[3] BLAST, [Online]. Available : http://www.ncbi.nlm.nih.gov/BLAST, 2004

[4] Entrez-Search and Retrieval System, [Online]. Available : http://www.ncbi.nlm.nih.gov/Entrez, 2004

[5] Eun-Koung Park, Don-Wan Kang, Chai-Young Jung, Jong-Min Bae, "A Wrapper Model for Integrated Access to Biological Information Sources", The KIPS Transactions, Vol.11-D, No.4, pp.768-772, August, 2004.

[6] I.A.Chen, V.M.Markowitz, "An Overview of the Object-Protocol Model(OPM) and OPM Data Management Tools", Inform. Syst., Vol.20, No.5, pp.5-10, April, 1995.

[7] J. Shanmugasundaram, E. Shekita, R. Barr, M. Carey, B. Lindsay, H. Pirahesh, B. Reinwald, "Efficiently Publishing Relational Data as XML Documents", VBDB Journal, Vol. 10, No.2-3, pp.133-154, 2001.

[8] J. Shanmugasundaram, J. Kiernan, E. Shekita, C. Fan, J. Funderburk, "Querying XML Views of Relational Data", VLDB Conference, pp.261-270. 2001.

[9] M. Carey, D. Florescu, Z. Ives, Y. Lu, J. Shanmugasundaram, E. Shekita, S. Subranmanian, "XPERANTO: Publishing Obejct-Relational Data as XML", In Proc. of the Int. Workshop on the Web and Databases (WebDB), pp.105-110, May, 2000.

[10] M. Frenandez, A. Morishima, D. Suciu, "Efficient Evaluation of XML Middle-ware Queries", SIGMOD '01, pp.103-114, May, 2001.

[11] M. Frenandez, W. Tan, D. Suciu, "SilkRoute : Trading between Relations and XML", In WWW9, pp.723-745, May, 2000.

[12] PDB, [Online], Availiable : http://www.rcsb.org/pdb/, 2005

[13] SAX 2.0, "Simple API for XML", [Online]. Available : http://www.saxproject, org/, 2002.

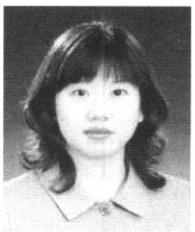
[14] Thomas Hernandez, Subbarao Kambhampati, "Integration of Biological Sources :Current Systems and Challenges Ahead", ASU CSE TR-03-005, pp.3-5, October, 2003.

[15] Val Tannen, "The Information Integration System K2", white paper, "http://db.cis.upenn.edu/K2/papers.html", pp.4-5.

[16] World-Wide Web Consotrium, "XML Schema Part 0:

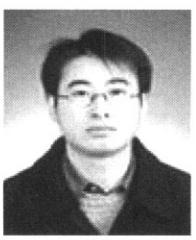
Primer", [Online]. Available :http://www.w3c.org/TR/xmlschema-0/, W3C Recommendation, 2001.
[17] World-Wide Web Consotrium, "XML Schema Part 1: Structures", [Online]. Available : http://www.w3c.org/TR/xmlschema-1/, W3C Recommendation, 2001.
[18] World-Wide Web Consotrium, "XML Schema Part 2: Datatypes", [Online]. Available : http://www.w3c.org/TR/xmlschema-2/, W3C Recommendation, 2001.
[19] World-Wide Web Consotrium, "XQuery 1.0: An XMl Query Language", [Online]. Available : http://www.w3c.org/TR/xquery/, W3C Working Draft, 2003.
[20] Zoe Lacroix, "Biological Data Integration: Wrapping Data and Tools", IEEE Transactions on information technology in biomedicine, Vol.6, No.2, pp.2-4, June, 2002.

박 은 경



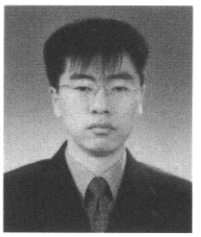
e-mail : pek1028@hanmail.net
1999년 경상대학교 컴퓨터과학과(학사)
2002년 경상대학교 컴퓨터과학과(석사)
2002년~현재 경상대학교 대학원 컴퓨터과학과 박사과정
관심분야 : XML, 데이터베이스 통합, 바이오인포매틱스

강 동 완



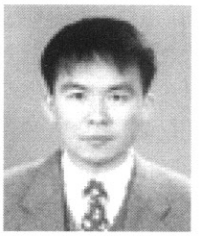
e-mail : wanni76@naver.com
2002년 경상대학교 컴퓨터과학과(학사)
2004년 경상대학교 컴퓨터과학과(석사)
2004년~현재 미디어코리스(주) 근무
관심분야 : XML, 데이터베이스 통합, XML 질의처리

정 채 영



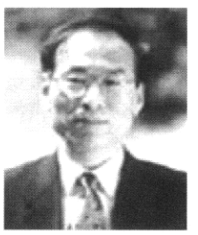
e-mail : wcdi@rtp.gsnu.ac.kr
1997년 경상대학교 전자계산학과(학사)
2001년 경상대학교 대학원 컴퓨터과학과(석사)
2004년 경상대학교 대학원 컴퓨터과학과(박사)
관심분야 : XML, WWW, 데이터베이스, 정보검색

김 현 주



e-mail : khj@jinju.ac.kr
1988년 경상대학교 전산통계학과(학사)
1990년 숭실대학교 전자계산학과(석사)
2000년 경상대학교 컴퓨터과학과(박사)
2002년~현재 진주산업대학교 컴퓨터공학부 조교수
관심분야 : 정보검색, 전자도서관, 웹 프로그래밍, XML

배 종 민



e-mail : jmbae@gsnu.ac.kr
1980년 서울대학교 수학교육과(학사)
1983년 서울대학교 대학원 계산통계학과(석사)
1995년 서울대학교 대학원 계산통계학과(박사)
1982년~1984년 한국전자통신연구소 연구원
1997년~1998년 Virginia Tech. 객원연구원
1984년~현재 경상대학교 컴퓨터과학부 교수
관심분야 : XML, 데이터베이스 통합, 웹 서비스, 생물정보