XML 데이타 관리시스템과 유전체 데이타베이스에의 응용

(An XML Data Management System and Its Application to Genome Databases)

이 경 희 † 김 태 경 †† 김 선 신 ††† 이 충 세 †††† 조 완 섭 †††††

(Kyung-Hee Lee) (Tae-Kyung Kim) (Sun-Shin Kim) (Chung-Sei Rhee) (Wan-Sup Cho)

요 약 최근들어 XML의 급속한 확산으로 인해 DBMS를 이용한 XML 데이타 관리의 필요성이 높아지고 있다. 기존의 DBMS를 이용한 XML 저장 및 검색에 관한 연구들은 편의성 측면이나 성능 측면에서 아직 해결할 문제점을 가지고 있다. 특히, 관계 DBMS를 이용하는 경우 복잡한 XML 문서를 간단한 테이블 형태로 변환하는 데이타 모델 변환과 그에 따른 질의 변환의 복잡성이 문제점으로 지적되고 있다. 본 논문에서는 UniSQL ORDBMS를 이용한 DTD 의존적 데이타 관리 시스템인 Xing을 제안한다. Xing 시스템에서는 ORDBMS의 객체 참조와 다중값 속성을 이용하므로 XML 데이타를 객체 모델로 매핑하는 것이 간단하다. 또한, DTD 의존적인 객체 스키마를 생성하므로 XML 저장 알고리즘이 간단하고, 검색을 위한 질의 변환도 용이하다. 특히, Xing에서는 SAX 파서를 이용하여 메모리 부하가 적은 고유의 Xing 트리를 생성하므로 대량의 XML 데이타를 빠르게 저장할 수 있다. 그리고, 질의결과를 XML 형태로 반환함으로써 완전한 XML 데이타 관리시스템으로 사용할 수 있다. GenBank의 유전체 XML 데이타에 대하여 Xing을 이용한 저장과 관계 데이터베이스로 저장한 경우에 검색성능을 비교한 결과 제안한 시스템이 최고 10배 까지 좋은 성능을 보였다.

키워드 : XML, 객체-관계 데이타베이스, XQuery, 경로 질의

Abstract As the XML data has been widely used in the Internet, it is necessary to store and retrieve the XML data by using DBMSs. However, relational DBMSs suffer from the model difference between graph structure of the XML data and table forms in relational databases. We propose an ORDBMS-based DTD-dependent XML data management system Xing. Xing stores XML data in a DTD-dependent form in an object database. Since the object database schema has a graph structure and supports multi-valued attributes, mapping from an XML data model and queries into an object data model and OQLs is a simple problem. For rapid storing of large quantities of the XML data, we use SAX parser with customized Xing-tree which requires a small memory space compared with the DOM-tree. Xing also returns the query result in an XML document form. We have implemented the Xing system on top of UniSQL object-relational DBMS for the validity checking and performance comparison. For XML genome data from GenBank, and experimental evaluation shows that Xing can provide significant performance improvement (maximum 10 times) compared with the relational approach.

Key words : XML, ORDBMS, XQuery, Path Query

· 본 논문은 산업자원부/한국산업기술재단 지역혁신인력양성사업의 지원을 받았음

† 정 회 원 : 서원대학교 교양학부 교수
khlee@seowon.ac.kr
†† 학생회원 : 충북대학교 정보산업공학과
misoh049@hanmail.net
††† 비 회 원 : 충북대학교 전자계산학과
sskim04@hotmail.com
†††† 종신회원 : 충북대학교 컴퓨터과학과 교수
csrhee@cbnu.ac.kr
††††† 종신회원 : 충북대학교 경영정보학과 교수
wscho@cbnu.ac.kr

논문접수 : 2003년 8월 20일
심사완료 : 2004년 3월 16일

1. 서 론

XML은 시스템간 상호운용성을 위한 언어로 고안되었기 때문에 사용자가 데이타에 관한 정보를 충분히 표현할 수 있도록 허용하고 있다[1]. 이런 특징은 사람뿐 아니라 기계도 프로그램을 이용하여 XML 데이타를 자동으로 처리 가능하게 한다. 특히 유무선 인터넷을 통해 컨텐츠 교환이 이루어지는 전자상거래, 디지털 도서관, 생물정보학 분야 등에서 컨텐츠가 XML 형식으로 만들

어지고 있으며, 그 활용분야는 더욱 확산될 전망이다.

이러한 XML의 확산에 따라 XML 데이타를 기존의 데이타베이스에 저장, 검색, 관리하는 연구가 주목을 받고 있다. 현재, Oracle과 MS-SQL Server 등의 관계 DBMS와 참고문헌[2,3] 등의 객체 DBMS에서 XML 문서를 데이타베이스에 저장하고 검색하는 기법을 제시하고 있다.

그러나, 기존의 DBMS를 이용한 XML의 저장 및 검색에 관한 연구들은 편의성 측면이나 성능 측면에서 문제점을 갖고 있다[4,5,6]. 관계 DBMS를 이용하는 경우, 많은 연구 결과가 제시되었으나[4,5,6,7], 트리 형태의 복잡한 XML 문서를 간단한 테이블 형태로 변환하는 데이타 모델 변환과 그에 따른 질의 변환의 복잡성이 문제점으로 지적되고 있다[2,6,8]. 객체 DBMS를 이용하는 경우는 객체 데이타 모델과 XML의 데이타 모델 유사성으로 인하여 변환 문제에서 장점이 있으나, 아직 이 분야에 관한 연구결과가 많지 않은 실정이다[2].

본 논문에서는 UniSQL 객체-관계 DBMS(Object-Relational DBMS:ORDBMS)[9]를 이용하여 XML 데이타를 저장하고 검색하는 ***Xing 시스템***을 제안한다. Xing 시스템의 특징은 다음과 같이 요약된다.

- DTD로부터 객체-관계 데이타베이스 스키마를 생성할 때 객체 참조와 다중값 속성 등의 객체-관계 데이타베이스 특징을 충분히 이용한다. 이렇게 함으로써 트리 형태의 XML 문서는 자연스럽게 복합 객체 형태로 데이타베이스에 저장된다.
- DTD 의존적 객체-관계 스키마를 생성하므로 XML 데이타의 저장 알고리즘이 간단하고, XML 질의를 객체-관계 DBMS의 질의로 변환하는 것이 용이하다. 이에 비하여 DTD 독립적 스키마는 다양한 구조의 XML 문서를 정해진 몇 개의 테이블에 저장할 수 있다는 장점은 있지만, 상세 검색을 위해 다양한 인덱스 구축이 필수적이고, DTD 의존적 스키마를 이용한 경우보다 많은 수의 인덱스가 필요하다[7]. 뿐만 아니라 XML 질의는 항상 DTD 기반으로 생성되기 때문에 DTD에서 제시한 구조와 다른 DTD 독립적 스키마를 사용하는 시스템의 경우 XML 질의를 데이타베이스 질의로 변환하는 것이 복잡하다[5].
- 객체-관계 데이타베이스 질의 결과를 XML 데이타로 반환함으로써 검색된 문서의 활용도를 높인다. XML 질의어를 객체-관계 데이타베이스 질의어로 변환하여 처리할 수 있고, 질의 결과를 XML 형태로 반환함으로써 즉시 다른 시스템이나 응용 프로그램에서 이용할 수 있도록 지원한다.

Xing은 유전자 데이타베이스 등 대규모의 XML 문서를 저장하고 관리하는데 적합한 시스템 구조이다. 이를 위하여 Xing 시스템에서는 DOM 대신에 SAX를 이용하여 고유의 효율적인 Xing 트리를 생성함으로써, 대량의 XML 데이타를 빠르게 저장할 수 있다. 특히, 저장 속도 개선을 위하여 한꺼번에 다수의 객체를 저장하는 대용량 적재(bulkloading) 방법을 지원한다. DOM의 경우 전체 문서를 트리 구조로 메모리에 적재하여 사용하므로 적은 양의 문서를 트리로 생성하여 임의노드를 접근하는 데는 유리하지만[10], 대량의 문서를 순차 접근할 때 필요 이상의 정보를 메모리에 적재함으로써 비효율적이라고 알려져 있다[11].

Xing 시스템의 실용성과 성능을 검증하기 위하여 대표적 생명정보기관인 GenBank[12]에서 제공하는 GBSeqXML 형식의 XML 유전체 데이타를 Xing 시스템에 저장하고 검색하면서 XML 데이타 처리 성능을 실험한다. 특히, 동일한 DTD와 XML 유전체 데이타에 대하여 UniSQL ORDBMS를 이용하여 관계 데이타베이스와 객체-관계 데이타베이스 형태로 각각 저장하고, 어느 형태의 데이타베이스에서 더 빠른 검색을 하는지 분석한다. 실험 결과, 객체-관계 데이타베이스 형태로 저장한 경우의 검색 성능이 관계 데이타베이스의 경우보다 최고 10배 까지 뛰어남을 확인하였다. 이는 XML 문서를 객체-관계 데이타베이스 형태로 저장·관리하는 것이 데이타베이스 사이즈가 적어지며, 조인(join) 비용을 줄일 수 있다는 점에 기인한다. 이러한 실험으로부터 XML 데이타를 관계 데이타베이스 형태보다 객체-관계 데이타베이스 형태로 저장하고 검색하는 것이 데이타 모델과 질의 변환에서 간단하며, 더욱 뛰어난 질의 처리 성능을 얻을 수 있음을 알 수 있다.

본 논문의 구성은 다음과 같다. 제2장에서는 XML 저장시스템에 관한 기존연구를 정리한다. 제3장에서는 제안하는 Xing 시스템의 구조를 살펴본다. 제4장에서는 Xing 시스템에 대한 성능 평가 및 분석 결과를 제시한다. 제5장에서 결론 및 향후 연구과제를 제시한다.

2. 관련연구

여기서는 XML 데이타에 대한 저장 시스템과 질의 처리 기법에 관하여 기존 연구 결과를 소개한다.

2.1 XML 저장시스템

XML 데이타는 임의의 엘리먼트가 하위 엘리먼트를 가질 수 있는 계층적 구조로 표현된다. 이러한 구조를 갖는 XML 데이타의 저장 및 관리 시스템에 관한 기존 연구는 다음과 같이 세 가지로 분류할 수 있다.

첫 번째 방법은 XML 데이타를 처리하기 위한 새로운 저장 시스템의 개발이다. Tamino[13], eXcelon[14] 등이 그 예이다. 이와 같은 시스템은 XML 데이타를 파일 단위로 저장하고, XML 데이타의 구조 정보를 인덱

스로 구축하여 원하는 데이타를 검색할 때 사용한다. 데이타 모델 변환이나 데이타 저장에서 부담이 적지만, 데이타 검색을 위하여 여러 가지 인덱스를 참조해야 하므로 대규모 XML 데이타 처리에서 성능이 낮다고 알려져 있다[4,5,15].

두 번째 방법은 관계 데이타베이스를 활용하는 경우이다. 안정된 관계 DBMS의 기능을 활용할 수 있다는 장점은 있지만 여기서도 여러 가지 문제점이 발생한다. 먼저, 계층적 그래프 형태의 XML 데이타 구조를 단순한 테이블 구조로 변환해야 한다[2]. 또한, XML 데이타에 자주 사용되는 다중값(collection)속성이 별도의 테이블로 표현되어야 하며, 엘리먼트 간의 순서정보 유지를 위해 부가정보를 추가해야 한다[2,5]. 그리고, XML 질의에 자주 사용되는 경로식은 처리비용이 높은 조인 연산으로 변환되어 처리되므로 비효율적이다[7]. 마지막으로, 질의 결과가 다중값을 포함하는 경우 다중값이 아닌 속성들에서 불필요한 중복이 발생하며, 이로 인해 질의 결과를 XML 데이타로 변환하는 알고리즘이 복잡하다[4]. 이러한 여러 가지 이유로 참고문헌 [7]에서는 관계 데이타베이스의 기능을 확장하지 않고서 XML 데이타 검색의 성능을 향상시키는 것은 어려운 일이라고 주장한다.

세 번째 방법은 객체 데이타베이스를 활용하는 것이다. 객체 데이타베이스는 관계 데이타베이스보다 데이타 모델이 XML과 유사하여 DTD로부터 객체 스키마를 생성하는 문제와 XML 질의를 OQL로 변환하는 과정이 간단해진다. 그리고, 가변 필드의 사용에 제한이 없어서 대용량 데이타의 저장이 쉽고, 다양한 다중값 데이타 형식이 정규 데이타 형식(built-in data type)으로 지원되기 때문에 XML의 순서정보 표현이 간단하다[2]. 또한, XML 질의에 자주 사용되는 경로식은 조인이 아니라 객체 참조 형태로 변환·처리되므로 질의 변환이 용이하고, 성능도 향상된다. 그리고, 질의 결과도 중첩된 형태로 제공되므로 간단히 XML 형태로 변환된다.

XML의 DTD를 객체 데이타베이스 스키마로 매핑하여 관리하는 연구로는 [2,3]이 있다. 관련연구[2]은 DTD 그래프에서 테이블로 매핑되는 엘리먼트 이하의 데이타를 서브 그래프(sub graph) 형태로 관리하는 방법을 제안하고, 그래프의 노드를 검색하는 메소드가 포함된 질의로 변환하는 방법을 제안하였다. 또한, 객체 데이타베이스가 관계 데이타베이스에 비해 저장공간 효율과 질의 처리시 조인연산이 적어서 성능이 우수하다는 실험결과를 제시하였다[2]. 그러나, 본 논문은 클래스로 매핑되지 않는 애트리뷰트는 클래스의 속성으로 모두 인라인 하여 질의 변환이 간단하고, OQL에 사용자 정의 메소드가 포함되지 않으므로 질의 변환과 질의 처리가 보다 간단하다. 관련연구[3]은 클래스 계승을 이용한 객체 데이타베이스 스키마를 생성 기법과 다중경로식 처리 방법을 제안하였으며, 여기서 제안한 기법은 서브 클래스를 이용하므로 클래스 개수는 증가하지만, 특정 서브 클래스에 대한 질의를 수행할 때 객체 탐색범위를 줄이는 장점이 있다. 그렇지만, 기존 DBMS와의 성능비교가 포함되어 있지 않고, 경로식 질의 처리 성능 향상 방법은 다루지 않았다.

본 논문의 Xing 시스템은 DTD로부터 객체 스키마를 생성할 때 관계 데이타베이스를 이용한 [7]의 기본 인라인(basic inlining) 방법과 유사한 것으로, [7]의 연구결과를 객체 데이타베이스에 적용하여 관계 데이타베이스에서 발생하는 질의 처리 성능 저하 문제를 해결하는 것이 주된 아이디어이다.

2.2 XML 질의 처리 기법

XML이 웹을 위한 새로운 마크업 언어의 표준이 되면서 다양한 XML 질의어가 제안되었다[1]. Lorel[16], XML-QL[17], XQL[19], XPath[20], XQuery[1] 등이 그 예이다. 대부분의 XML 질의어는 XML 데이타의 특징을 반영하여 다음과 같은 연산을 제공한다.

- 계층적 데이타 접근 연산
- 엘리먼트 간의 순서를 이용한 연산
- 다중값 엘리먼트를 위한 연산

표 1은 대표적인 XML 질의어가 제공하는 연산을 비교한 것이다. 이 중 표현력이 가장 뛰어난 XQuery가 XML 질의어 표준의 유력한 후보로 논의되고 있다[1].

표 1 대표적 XML 질의어 비교[18]

	Lorel	XML-QL	XQL	Xpath	XQuery
조인	○	○	×	×	○
경로표현	○	○	○	○	○
순서질의	○	×	○	○	○
집합연산	○	×	×	×	○
질의결과 구조정의	×	○	×	○	○

관계 데이타베이스에서는 XML 질의에서 자주 사용되는 경로식을 경로속의 각 엘리먼트에 대응되는 테이블 간의 조인으로 변환하여 처리한다. 예를 들어 Query 1에서 사용된 경로식을 SQL로 변환할 때 SQL:Query 1과 같이 2번의 조인이 필요하다.

(Query 1)

```
FOR $d IN DOCUMENT("GBSeq.xml")//GBSeq
WHERE $d/Gbseq_moltype/value='DNA'
RETURN $d/locus, $d/moltype/value
```

(SQL:Query 1)

```
select GBSeq.locus, moltype.value
from GBSeq, moltype
where GBSeq.gbseqpk=moltype.gbseqfk and
      moltype.value='DNA';
```

반면에 Query1을 OQL로 변환하면 경로식은 OQL: Query 1과 같이 거의 변하지 않으므로 질의변환이 용이하다.

(OQL:Query 1)

```
select GBSeq.locus, GBSeq.moltype.value
from GBSeq
where moltype.value = 'DNA';
```

3. Xing 시스템 구조

여기서는 Xing 시스템의 구조를 설명한 다음, 스키마 생성과 데이타 저장 방법 및 질의 변환기를 차례로 기술한다.

Xing 시스템은 객체 데이타베이스를 이용한 DTD 기반 XML 데이타 저장 및 관리 시스템으로 전체적인 구조는 그림 1과 같다. Xing 시스템에는 스키마 생성기(Schema Generator), XML 저장기(XML Data Insertion Module), 질의 변환기(Query Translator), 질의 결과 XML 변환기(Result to XML Translator) 등의 서브 시스템이 포함되어 있다.

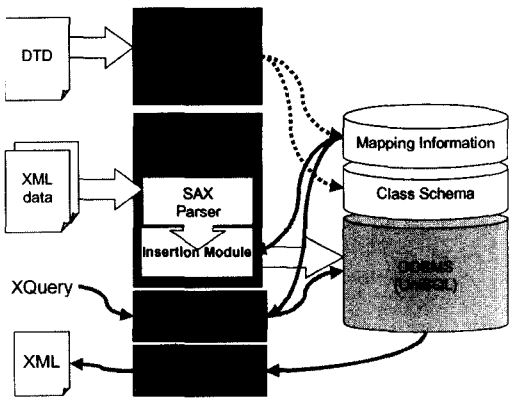

그림 1 Xing 시스템 구조도

- 스키마 생성기 : 주어진 DTD로부터 객체 데이타베이스 스키마를 생성하는 모듈
- XML 데이타 저장기 : XML 데이타를 분할하여 객체 데이타베이스의 클래스에 저장하는 모듈 (삽입 질의나 대용량 적재를 이용함)
- 질의 변환기 : XQuery를 UniSQL의 OQL로 변환하는 모듈
- 질의 결과 XML 변환기 : 질의 결과를 XML 형식으로 변환하는 모듈

다음 절에서 각 세부 모듈에 관하여 상세히 설명한다.

3.1 스키마 생성기

객체 데이타베이스 모델은 XML 데이타 모델과 유사하고, 풍부한 데이타 형을 가지므로 XML DTD를 객체 데이타베이스 스키마로 자연스럽게 변환할 수 있다. Xing 시스템의 객체 스키마 생성 방법은 [7]에서 제안한 방법을 객체 데이타베이스에 적합하도록 변경한 것으로, 다음과 같이 객체 데이타베이스의 특징을 활용한다.

첫째, DTD에서 엘리먼트가 1개 이상의 하위 엘리먼트를 가지거나 애트리뷰트를 가지면 객체 스키마의 클래스로 매핑한다. 그리고, 엘리먼트가 가지는 모든 애트리뷰트와 하위 엘리먼트는 해당 클래스의 속성으로 포함시킨다. 여기에서 하위 엘리먼트가 애트리뷰트 또는 하위 엘리먼트를 가지는 경우 속성-도메인 관계가 된다. DTD를 파싱하여 클래스를 생성할 때 사용하는 규칙으로써 기본적인 방법은 [7]과 같지만, 객체 데이타베이스의 특성인 다중값 속성을 활용하도록 수정하였으며, DTD 그래프의 끝노드만 인라인한다.

둘째, DTD에서 '*'나 '+'연산자를 가지는 하위 엘리먼트를 가지는 엘리먼트는 그 하위 엘리먼트를 해당 클래스의 다중값 속성으로 매핑한다. 그리고, XML 데이타에서 엘리먼트 간의 순서정보는 객체 데이타베이스의 다중값 데이타 형식을 이용하여 표현한다. 관계 데이타베이스의 경우, 다중값을 지원하지 않기 때문에 별도의 테이블을 생성하였으며, 순서정보의 표현을 위하여 별도의 속성을 포함시켜 데이타베이스 사이즈가 커지게 된다.

셋째, DTD에서 나타나는 엘리먼트 간의 상하 관계(parent-child relationship)를 표현하기 위하여 객체 데이타베이스는 속성-도메인(domain) 관계로 설정하면 된다. 관계 데이타베이스의 경우에는 테이블 간의 관계를 설정하기 위하여 주키-외래키를 이용한다.

알고리즘 1은 DTD 그래프로부터 객체 데이타베이스 스키마를 정의하는 것으로써 앞서 언급한 규칙세 가지에 따라 엘리먼트를 클래스로 정의하고, 클래스로 매핑되지 않는 엘리먼트는 상위 엘리먼트의 속성으로 정의한다. 즉, DTD에서 EMPTY, #PCDATA, #CDATA와 같이 선언된 엘리먼트나 애트리뷰트는 DTD 그래프의 끝노드에 해당되는데, 이런 경우 상위 엘리먼트의 속성으로 정의한다. Define_Class(e)는 e를 클래스로 정의하는 서브프로그램, Define_MultiValued_Attribute(s_e, e)는 클래스(e)의 다중값 속성으로 s_e를 정의하는 서브프로그램 그리고 Define_Attribute_of_Class(s_e, e)는 s_e를 클래스(e)의 속성으로 정의하는 서브프로그램이다.

알고리즘 1 객체 데이타베이스 스키마 생성 알고리즘

```
INPUT : DTD 그래프
        e : 엘리먼트, E : 엘리먼트의 집합
        s_e ∈ S_e  S_e: 엘리먼트 e의 하위 엘리먼트 집합
        p_e ∈ P_e  P_e: 엘리먼트 e의 상위 엘리먼트 집합
OUTPUT : 객체 데이타베이스 스키마
METHOD :
  for e in E {
    if n(S_e)≥1 AND n(A_e)≥1 then {
                Define_Class(e);
                for s_e in S_e {
                if s_e has '*' OR '+' then
                Define_MultiValued_Attribute(s_e, e);
                else Define_Attribute_of_Class(s_e e);
                };
    }else {
                Define_Attribute_of_Class(e, p_e);
    };
  };
```

Xing 시스템은 DTD 독립적인 방식이나 관계 데이타베이스를 사용하는 경우와는 달리 주어진 DTD 그래프를 객체 데이타베이스의 복합 객체를 나타내는 스키마 그래프 형태로 변환하므로 DTD 그래프의 형태가 거의 변형없이 객체 스키마에 보존된다. 따라서, 스키마 생성이 간단할 뿐 아니라 DTD에 대하여 기술하는 XML 질의도 간단한 알고리즘으로 객체 데이타베이스 스키마에 대한 OQL로 변환할 수 있게 된다.

DTD로부터 객체 스키마를 생성하는 동안 DTD와 객체 스키마 간의 매핑은 매핑정보(mapping information) 파일에 저장된다. 이 정보는 이후 XML 데이타를 데이타베이스에 저장할 때와 XML 질의를 OQL로 변환할 때 이용된다. 표 2는 DTD의 각 엘리먼트와 애트리뷰트가 클래스 스키마의 어느 요소로 매핑되는지 나타내는 매핑정보의 구조이다.

표 2 매핑정보 DTD

```
<!ELEMENT Maps (ClassMap)*>
<!ELEMENT ClassMap (Element)*>
<!ATTLIST ClassMap name                #REQUIRED,
                   root                #REQUIRED
                   empty               #REQUIRED>
<!ELEMENT Element  (#EMPTY)>
<!ATTLIST Element  name        #REQUIRED,
                   referencing #REQUIRED,
                   set         #REQUIRED
                   attr        #REQUIRED>
```

매핑정보는 3개의 엘리먼트를 가지는데, Maps는 루트 엘리먼트이고, ClassMap은 클래스로 매핑되는 엘리먼트에 관한 정보를 표현하며, Element는 클래스로 매핑한 애트리뷰트 또는 하위 엘리먼트를 매핑한 정보를 표현한다. ClassMap이 갖는 name 애트리뷰트는 클래스 이름, root 애트리뷰트는 엘리먼트가 루트인지의 여부, empty 애트리뷰트는 해당 엘리먼트가 내용을 가지는지의 여부를 나타낸다. 그리고 Element의 name 애트리뷰트는 속성이름, referencing 애트리뷰트는 다른 클래스를 참조하는지 여부, set 애트리뷰트는 해당 속성이 다중값을 갖는지 여부, 그리고 attr 애트리뷰트는 이 속성에 대응되는 엘리먼트가 DTD 상에서 애트리뷰트인지 여부를 나타낸다.

그림 2는 Xing 시스템을 이용하여 GenBank[12]의 GBSeqXML DTD(부록 1)를 객체 데이타베이스 스키마로 변환한 것을 나타낸다. GenBank에는 생명체에 대한 유전체 정보가 수록되어 있으며, 질의 검색 결과를 부록 1의 DTD에 따라 XML 형태로 리턴해 준다. Xing 시스템은 부록 1의 DTD를 입력으로 받아서 그림 2와 같이 9개의 클래스로 구성되는 객체 데이타베이스 스키마를 생성한다.

3.2 XML 데이타 저장기

XML 데이타 저장기에서는 XML 데이타를 파싱하고, DTD로부터 생성한 매핑 정보를 이용하여 삽입질의 또는 대용량 적재 기법으로 XML 데이타를 해당 클래스에 나누어 저장한다.

저장기에서는 XML 데이타를 파싱할 때 매핑정보를 참조하여 각 엘리먼트마다 고유번호, 이름, 클래스 여부, 값, 엘리먼트 깊이 정보를 추출하고, 이를 Xing 시스템의 고유 트리인 Xing-Tree로 생성한다. Xing-Tree는 SAX 파서를 사용하기 때문에 DOM으로 생성한 트리보다 메모리 효율이 좋고, 순회속도가 빠르다. 왜냐하면, DOM은 문서 전체를 메모리에 적재한 후 트리를 구성하기 때문에 1GB이상의 문서를 DOM 파서로 파싱하면 페이지 폴트(Page fault)가 발생한다. 반면에 SAX 파서는 문서 전체를 메모리에 적재하지 않고, 문서 처음부터 끝까지 읽으면서 발생하는 시작태그의 발견이나 끝태그 발견과 같은 이벤트를 인식하여 사용자 응용프로그램에게 알려주어 처리할 수 있도록 하기 때문에 더 효과적이다[10,11].

그러므로, Xing 시스템은 메모리 부하를 줄이기 위하여 저장에 필수적인 정보만 추출하여 메모리 요구량이 작은 Xing-Tree를 생성함으로써 대량의 XML 데이타를 빠르게 처리할 수 있게 된다. 저장기에서는 생성된 Xing-Tree를 깊이우선(Depth-First) 방식으로 순회하면서 매핑정보를 참조하여 삽입 질의를 생성하거나 대규모 XML 문서의 신속한 입력을 원하는 경우 대용량 적재를 위한 입력 파일을 생성한다. 이렇게 질의를 만들거나 로

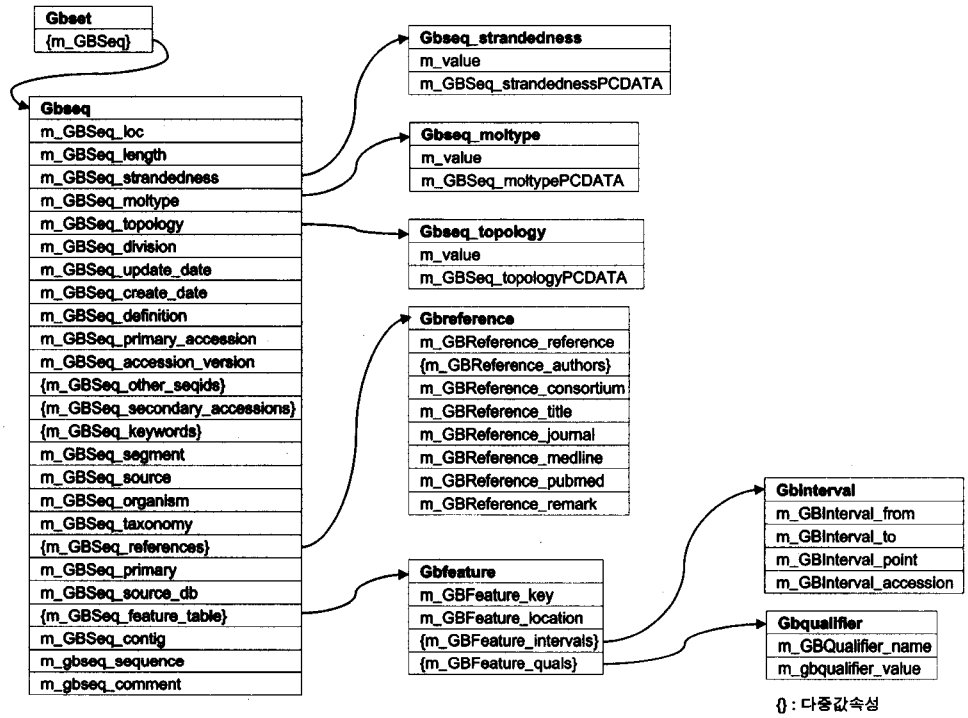

그림 2 객체 데이타베이스 스키마

드(load) 파일을 생성하여 데이타베이스에 저장한다.

3.3 XML 질의 변환기

여기서는 DTD를 기반으로 작성한 XQuery를 OQL로 변환하는 방법을 소개한다. XML 질의어의 대표적인 특징은 경로식과 다중값 관련 연산을 포함하는 것이며, XQuery도 예외는 아니다. 이러한 연산의 처리를 위하여 먼저 몇가지 정의를 한다.

정의 1. XML 질의에서 경로식

XML 질의어에서 사용하는 경로식 P_x : $//E_0/E_1/\ldots/E_n$는 다음과 같이 정의한다.

1. $E_0, E_1, \ldots, E_n$은 엘리먼트 이름이다.
2. E_0는 루트-엘리먼트, $i<j$이면 E_j는 E_i의 하위-엘리먼트이다.
3. '*', '?'는 다중경로를 표현하는데 사용하는 연산자이다.

다중경로는 여러 가지 단순경로를 포괄하는 것으로써, '*'은 여러 개의 엘리먼트로 구성된 경로식을 표현하며, '?'는 하나의 엘리먼트로 구성된 경로식을 표현한다. 예를 들어, 다중 경로식 A/?/C은 A 엘리먼트가 임의의 한 엘리먼트 다음에 C 엘리먼트로 이어지는 경로를 나타낸다.

정의 2. 객체 데이타베이스 질의에서 경로식

객체 데이타베이스에 대한 스키마 그래프 G에 포함된 경로식 P_o : $C_0.A_1.\cdots.A_n$는 다음과 같이 정의한다[21].

1. $C_0, \cdots, C_n$ 은 G에 속하는 클래스 이름
2. C_i는 클래스 C_{i-1}에서 정의된 속성 A_i의 도메인, $(1 \le i \le n)$

객체 DBMS 이용하는 Xing 시스템의 XML 질의처리에는 XML 경로식 P_x를 객체 데이타베이스 경로식 P_o로 변환하는 것이 필수적인 문제이다. 스키마 생성기에서 만든 매핑정보를 이용하여 엘리먼트와 클래스간의 매핑관계를 참조하여 경로식 P_x에 있는 엘리먼트가 객체 데이타베이스 스키마의 어떤 데이타 형식으로 변환되었는지 알 수 있다. 객체 데이타베이스의 형식은 다음의 세 가지 유형으로 분류할 수 있는데 정규데이타형식(built-in data type)인 AtomicType, 사용자 정의 데이타 형식(user defined data type)인 ComplexType 그리고 위의 두 가지 유형을 다중값으로 갖는 형식인 MutipleType이다.

XML 경로식이 알고리즘 2에서와 같이 객체 데이타베이스의 데이타 형식에 따라 서로 다르게 OQL 경로식으로 변환된다. 엘리먼트가 AtomicType으로 매핑되었으면 속성 이름을 반환하고, ComplexType이면 클래스 이름을 반환하며, MultipleType이면 다중값 속성이기

알고리즘 2 XML 질의 변환 알고리즘

```
INPUT : XQuery의 경로식
OUTPUT : OQL의 경로식
METHOD :
  for $e in $p{
      case AtomicType    return $a;
      case ComplexType   return $c;
      case MultipleType  return Table($pe.$e) as $r($a);
  };
```

때문에 OQL에서 Unnest된 가상클래스로 변환하는 문법에 따른 표현식을 반환한다.

본 논문에서 제안한 스키마 생성 방법을 이용하면 P_x에서 P_o로의 변환은 매핑정보를 이용하여 (알고리즘) 와 같이 간단히 이루어진다. 반면에, P_x를 관계 데이타베이스의 SQL로 변환하는 문제는 조건절에서 경로식에 포함된 각 엘리먼트가 대응되는 테이블 간의 조인으로 변환되고, 경로식에 다중값이 있을 경우 조인이 추가되어야 하므로 질의 변환이 복잡해진다. 다음의 예를 통하여 두 경우의 차이를 살펴보자.

예제 1. 객체 데이타베이스와 관계 데이타베이스에서 경로식 P_x 변환비교.

(XQuery)의 경로식 : GbSeq/GbSeq_moltype/value ='DNA'

(OQL)의 경로식 : GbSeq.GbSeq_moltype.value ='DNA'

(SQL)의 조건절 : GbSeq.gbseqPK = GbSeq_moltype.gbseqFK and GbSeq_moltype.value='DNA'

예제 1에서 보는 바와 같이 XQuery 경로식은 슬래쉬(/)를 점(.)으로 대치하여 간단히 그대로 OQL 경로식으로 변환할 수 있다. 그러나, SQL에서는 경로식에 사용된 엘리먼트에 대하여 해당하는 테이블을 찾고, 이들을 조인하여 변환해야 한다. 더구나 XQuery 경로식에 다중값 엘리먼트가 있으면 이는 별도의 테이블에 저장되어 있으므로 조인이 추가되어 질의 변환 과정이 더욱 복잡해진다.

3.4 질의 결과 XML 변환기

질의 결과를 XML 데이타로 변환할 때도 객체 데이타베이스 형태로 관련된 XML 데이타의 경우가 관계 데이타베이스의 경우보다 변환 절차가 단순하다. 예를들어 제2.2절의 (Query 1)을 OQL과 SQL로 각각 변환하여 실행하였을 때 질의 결과는 그림 3과 그림 4와 같이 반환된다.

객체 데이타베이스는 다중값을 허용하기 때문에 그림 3과 같이 중첩된 형태의 결과를 반환하지만, 관계 데이타베이스는 그림 4와 같이 그림 3에 Unnest 연산[22]을 적용한 형태로 결과를 반환한다. 그러나, 그림 3과 같이 중첩된 형태의 질의 결과는 불필요한 중복이 제거되어 있기 때문에 XML 문서로 간단히 변환될 수 있다. 그림 4의 결과를 XML 형태로 변환하려면 Nest 연산[22]을 적용하여 그림 3의 형태로 변환한 후, XML 문서를 생성해야 하므로 그림 3의 경우보다 복잡하다.

Xing 시스템은 질의 결과에 대하여 두 가지 형태의 XML 변환방법을 제공한다. 하나는 기본변환방식으로 XQuery의 RETURN 절에 별도의 질의 결과 구조를 지정하지 않은 경우로써 객체 데이타베이스의 형태로 결과가 리턴된다. 두 번째는 지정변환방식으로 질의결과 구조를 새롭게 지정한 경우로써 RETURN절에서 명시한 태그이름 및 구조를 이용하여 질의 결과를 XML 형태로 리턴하는 방법이다.

4. 성능 평가 및 분석

여기서는 Xing 시스템의 성능을 평가하는 실험을 수행한다. 성능평가의 주된 목적은 XML 데이타를 객체 데이타베이스 형태로 저장하는 방식과 관계 데이타베이스 형태로 저장하는 방식의 성능 차이를 분석하기 위함이다.

관계 데이타베이스 스키마는 [7]에서 제안한 방식으로 생성하고, 객체 데이타베이스 스키마는 Xing 시스템의

```
== <SELECT 명령어의 결과 (라인 1 안의)> ==

  dt_qualifier.n_gbqualifier_name  feature.n_gbfeature_key  dt_qualifier.n_gbqualifier_value  gbreference.n_gbreference_reference  gbreferen
ce.n_gbreference_authors.n_gbauthor
================================================================================
  'bound_moiety'        'misc_binding'        'DNA'                '5 (bases 1 to 2986)'  ('Beato,M.', 'Hagen,G.', 'Muller,S.', 'Suske,G.')
  'bound_moiety'        'misc_binding'        'DNA'                '4 (bases 1 to 2986)'  ('Grzeschik,K.H.', 'Kalff_Suske,M.', 'Kunz,J.', '
Suske,G.')
  'bound_moiety'        'misc_binding'        'DNA'                '3 (bases 1 to 2986)'  ('Beato,M.', 'Dennig,J.', 'Hagen,G.', 'Preiss,A.'
, 'Suske,G.')
  'bound_moiety'        'misc_binding'        'DNA'                '2 (bases 1 to 2986)'  ('Farber,D.B.', 'Gribanova,V.E.', 'Knox,B.E.', 'L
erner,L.E.', 'Whitaker,L.')
  'bound_moiety'        'misc_binding'        'DNA'                '1 (bases 1 to 2986)'  ('Dehal,A.', 'Bernstein,S.L.', 'Bouffard,G.', 'Pe
terson,K.', 'Ray,S.', 'Smith,D.', 'Touchman,J.W.', 'Wistow,G.', 'Wyatt,M.K.')

5 rows selected.
```

그림 3 OQL 질의결과

```
=== <SELECT 명령어의 결과 (라인 1 안의)> ===

gbqualifier_name      gbfeature_key        gbqualifier_value    gbreference_reference   gbauthor
=====================================================================================================
'bound_moiety'        'misc_binding'       'DNA'                '5 (bases 1 to 2986)'   'Hagen,G.'
'bound_moiety'        'misc_binding'       'DNA'                '5 (bases 1 to 2986)'   'Muller,S.'
'bound_moiety'        'misc_binding'       'DNA'                '5 (bases 1 to 2986)'   'Beato,M.'
'bound_moiety'        'misc_binding'       'DNA'                '5 (bases 1 to 2986)'   'Suske,G.'
'bound_moiety'        'misc_binding'       'DNA'                '4 (bases 1 to 2986)'   'Kalff_Suske,M.'
'bound_moiety'        'misc_binding'       'DNA'                '4 (bases 1 to 2986)'   'Kunz,J.'
'bound_moiety'        'misc_binding'       'DNA'                '4 (bases 1 to 2986)'   'Grzeschik,K.H.'
'bound_moiety'        'misc_binding'       'DNA'                '4 (bases 1 to 2986)'   'Suske,G.'
'bound_moiety'        'misc_binding'       'DNA'                '3 (bases 1 to 2986)'   'Hagen,G.'
'bound_moiety'        'misc_binding'       'DNA'                '3 (bases 1 to 2986)'   'Dennig,J.'
'bound_moiety'        'misc_binding'       'DNA'                '3 (bases 1 to 2986)'   'Preiss,A.'
'bound_moiety'        'misc_binding'       'DNA'                '3 (bases 1 to 2986)'   'Beato,M.'
'bound_moiety'        'misc_binding'       'DNA'                '3 (bases 1 to 2986)'   'Suske,G.'
'bound_moiety'        'misc_binding'       'DNA'                '2 (bases 1 to 2986)'   'Lerner,L.E.'
'bound_moiety'        'misc_binding'       'DNA'                '2 (bases 1 to 2986)'   'Gribanova,V.E.'
'bound_moiety'        'misc_binding'       'DNA'                '2 (bases 1 to 2986)'   'Whitaker,L.'
'bound_moiety'        'misc_binding'       'DNA'                '2 (bases 1 to 2986)'   'Knox,B.E.'
'bound_moiety'        'misc_binding'       'DNA'                '2 (bases 1 to 2986)'   'Farber,D.B.'
'bound_moiety'        'misc_binding'       'DNA'                '1 (bases 1 to 2986)'   'Wistow,G.'
'bound_moiety'        'misc_binding'       'DNA'                '1 (bases 1 to 2986)'   'Bernstein,S.L.'
'bound_moiety'        'misc_binding'       'DNA'                '1 (bases 1 to 2986)'   'Wyatt,M.K.'
'bound_moiety'        'misc_binding'       'DNA'                '1 (bases 1 to 2986)'   'Ray,S.'
'bound_moiety'        'misc_binding'       'DNA'                '1 (bases 1 to 2986)'   'Behal,A.'
'bound_moiety'        'misc_binding'       'DNA'                '1 (bases 1 to 2986)'   'Touchman,J.W.'
'bound_moiety'        'misc_binding'       'DNA'                '1 (bases 1 to 2986)'   'Bouffard,G.'
'bound_moiety'        'misc_binding'       'DNA'                '1 (bases 1 to 2986)'   'Smith,D.'
'bound_moiety'        'misc_binding'       'DNA'                '1 (bases 1 to 2986)'   'Peterson,K.'

27 rows selected.

1 명령어가 수행 되었습니다.
한글sqlx>
```

그림 4 SQL 질의결과

표 3 시스템 사양

구분	사양 및 기종	
	실험환경-1	실험환경-2
OS	SunOS 5.7	Windows 2000 server
RAM	1GB	540MB
CPU	400Mhz * 2	PentiumⅣ 2.4GHz
DBMS	UniSQL ORDBMS UniSQL RDBMS	UniSQL ORDBMS MS SQL-Server

표 4 객체 데이타베이스와 관계 데이터베이스의 공간복잡도 비교

DB종류 / 데이타용량	객체 데이타베이스		관계 데이타베이스		객체 감소율(%) = ⓐ/ⓑ
	class	ⓐobject	table	ⓑrecord	
80MB	9	208,074	21	415,010	50%
160MB	9	478,665	21	1,045,632	46%
240MB	9	717,595	21	1,625,513	44%

스키마 생성 알고리즘을 사용한다. 그리고, 동일한 XML 데이타를 관계 데이타베이스 형태와 객체 데이타베이스 형태로 각각 저장하여 표 3에서 제시한 실험환경-1과 실험환경-2에서 검색 성능과 공간 요구량을 분석한다. 실험환경을 두 가지로 한 것은 특정 RDBMS의 성능으로 인해 XML 데이타 저장관리 기반 시스템으로써 ORDBMS의 성능이 낮게 혹은 높게 평가가 되는 오류를 방지하기 위한 것이다.

제안한 방식과 RDBMS를 사용하는 두 가지 저장방식의 성능을 비교하는 실험은 대표적인 생물학 데이타베이스인 GenBank로부터 검색한 GBSeqXML형식 유전체 데이타를 Xing 시스템을 통해 하부 DBMS인 UniSQL과 MS SQL-Server에 각각 저장한 후 저장 공간의 효율과 질의 처리 성능을 비교 분석한다. 실험데이타는 실험환경-1에서는 80MB, 160MB, 240MB와 같이 다양한 크기의 데이타를 사용하고, 실험환경-2는 1GB의 실험데이타를 사용한다.

4.1 공간복잡도

표 4는 동일한 XML 데이타에 대하여 객체 데이타베이스와 관계 데이타베이스의 공간복잡도를 비교한 것이다. GBSeqXML DTD(부록 1)를 객체 데이타베이스 스키마로 매핑한 경우 클래스가 9개, 관계 데이타베이스 스키마로 매핑한 경우 테이블 수가 21개이다.

관계 데이타베이스는 다중값 지원하지 않으므로 이를 다루기 위해 별도의 테이블이 생성된다. 이로 인해 표 4에서와 같이 관계 데이타베이스의 테이블 수가 객체 데이타베이스의 클래수 개수보다 많아지게 된다. 그리고, 객체 데이타베이스는 다중값을 하나의 객체로 표현할 수 있지만, 관계 데이타베이스는 다중값의 원소 하나 하나를 다른 레코드로 구분하여 생성하기 때문에 레코드의 개수도 더욱 많아지게 된다.

표 4에서 데이타의 용량이 80MB인 경우 객체 데이타베이스가 관계 데이타베이스에 비하여 투플수가 50%감소, 240MB인 경우 44% 감소함을 보여주고 있다. 즉, 데이타의 용량이 클수록 관계 데이타베이스에 비하여 객체 데이타베이스의 투플 수가 더욱 감소됨을 알 수 있다.

4.2 시간복잡도

실험에 사용한 질의는 XML 질의에서 널리 사용되는

두 가지 유형이며, 각각에 대하여 선택률을 다양하게 변경하면서 질의 성능을 측정한다.

질의유형-1. 경로식의 시작노드에 서술식(predicate)을 부과하고, 이를 만족하는 경로식 끝노드 객체를 선택하는 질의

(organisim이 사람인 서열의 qualifier를 추출)

```
FOR $d IN DOCUMENT("GBSeq.xml")//GBSeq
WHERE $d/GBSeq_organism='Human'
RETURN $d/GBSeq_feature_table/GBfeature/
        GBfeature_quals/GBqualifier_name
```

질의유형-2. 서로 다른 두 개의 경로식에 대하여, 한 경로식의 끝노드에 서술식을 부과하고, 이를 만족하는 시작노드의 객체로부터 또 다른 경로식의 끝노드 객체를 선택하는 질의

(qualifier값이 mRNA인 서열을 참조하는 feature 키 값 및 참고문헌의 저자를 추출)

```
FOR $d IN DOCUMENT("GBSeq.xml")//GBSeq
WHERE $d/feature_table/feature/feature_quals
        /Qualifier_value='mRNA'
RETURN
  $d/feature_table/feature/feature_quals/gualifier_name/,
  $d/feature_table/feature/feature_key,
  $d/feature_table/feature/feature_quals/qualifier_value,
  $d/references/reference/reference_authors
```

그림 5는 질의유형-1의 수행 시간을 비교한 그래프이다. 그래프의 X축은 질의결과의 선택률을 나타내며, Y축은 질의처리시간을 나타낸다. 데이타의 양이 80MB인 경우는 객체 데이타베이스가 관계 데이타베이스의 경우보다 최대 10배 정도까지 좋은 성능을 보인다. 질의 결과 선택률이 낮수록 객체 데이타베이스가 더욱 좋은 성능을 보이고, 선택률이 90%이상인 경우도 차이가 적어지기는 하지만 여전히 성능이 더 좋다. 그림 5의 (b), (c)는 대규모 데이타의 경우에 비교한 것으로 데이타의 양이 클수록 객체 데이타베이스의 성능이 더욱 뛰어남을 알 수 있다. 이러한 결과는 객체 데이타베이스가 표 4에서 나타난 것처럼 관계 데이타베이스 보다 데이타베이스 투플 수가 적고, 질의에 포함된 경로식을 처리할 때 조인이 아니라 객체 참조 관계를 이용하여 해당 객체를 검색하기 때문이다. 즉, 객체 데이타베이스는 경로식의 시작노드(질의유형-1의 GBSeq)의 서술식을 만족하는 객체만 선택하여 경로식 끝노드의 객체를 객체 참조 방식으로 검색하기 때문에 시작노드에서 서술식을 만족하는 객체 수가 적은 경우 (즉, 질의 결과 선택률이 낮은 경우) 관계 데이타베이스보다 10배 까지 빠르게 질의를 처리한다. 그런데, 선택률이 90%이상인 경우, 경로식의 시작노드에서 서술식을 만족하는 객체가 많기 때문에 관계 데이타베이스와 마찬가지로 처리비용이 커지게 되며, 그 결과 질의 성능에 차이가 적어지게 된다. 유의할 점은 대부분의 경우 질의의 결과는 전체 데이타

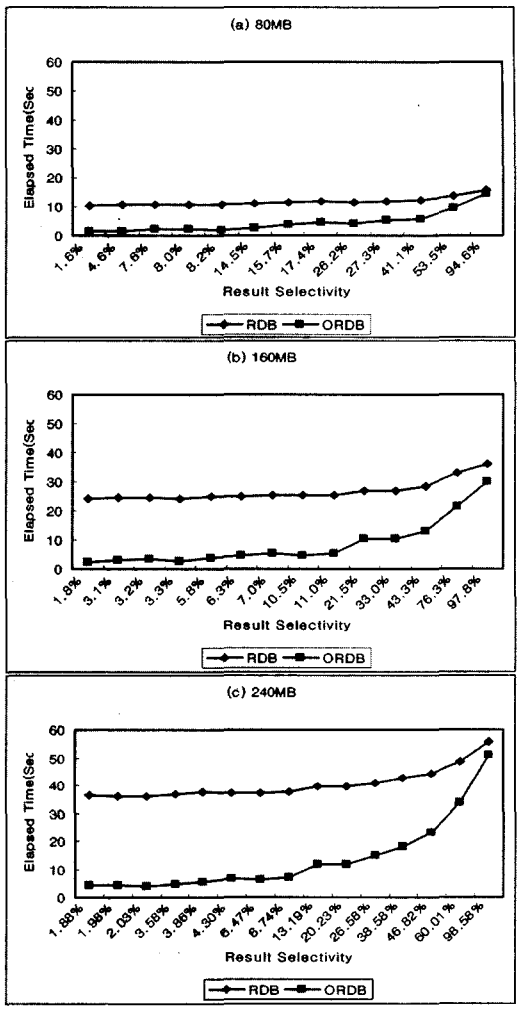

그림 5 실험환경-1에서 질의유형-1 수행 시간 비교

베이스보다 매우 작으므로 제안된 기법은 유용하다고 판단된다.

그림 6은 경로식을 2개 포함한 질의유형-2의 수행시간을 비교한 그래프이다. 질의유형-2는 두 개의 경로식이 질의에 포함되므로 전체적인 질의 수행 시간이 질의유형-1보다 오래 걸린다. 특히, 관계 데이타베이스를 사용하면 투플 수가 객체 데이타베이스의 경우보다 많은 상황에서 두 경로식에 대한 조인까지 처리해야 하므로 질의 수행 시간이 더욱 증가하게 된다. 그림 6에서 보는 바와 같이 질의 결과 선택률이 높아질수록 질의 성능이 더욱 떨어진다. 질의 선택률이 높아지면 한 경로식에서 조건을 만족하는 시작노드의 객체 수가 많기 때문에 이들 객체로부터 또다른 경로식의 끝노드 객체를 가져오는 비용이 증가하게 되어 비용이 높아지게 된다.

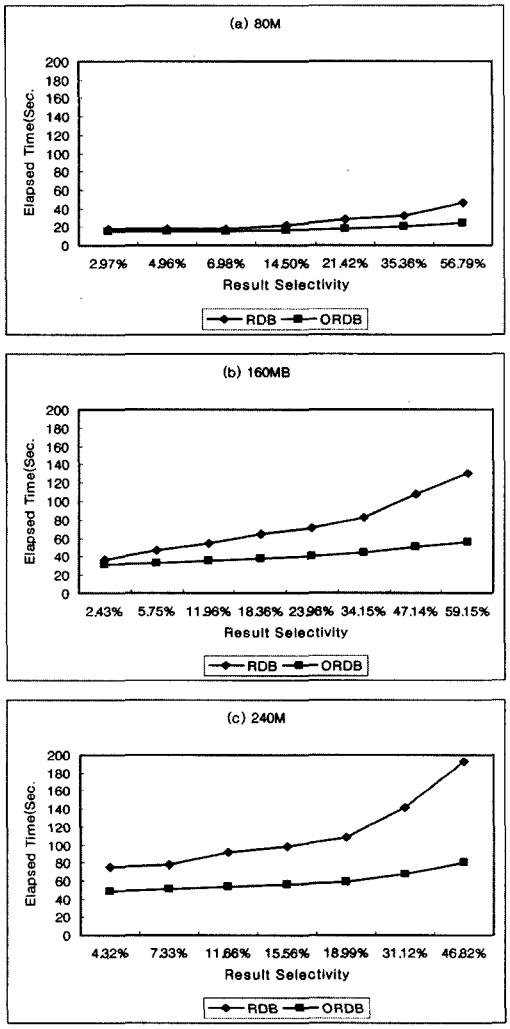

그림 6 실험환경-1에서 질의유형-2 질의 수행 시간 비교

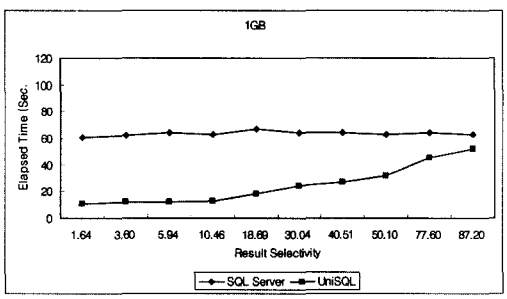

그림 7 실험환경-2에서 질의유형-1 질의수행시간 비교

그림 7은 1GB의 XML 데이타를 저장한 실험환경-2에서 질의유형-1을 처리하는 시간과 결과 선택률의 상관관계를 나타낸 그래프이다. 제시된 그래프에서 보는 바와 같이 SQL Server는 결과 선택률에 거의 영향 받지 않고 고른 성능을 나타냈지만, 실험환경-1의 데이타 양이 240MB인 경우 질의처리 시간보다 평균 10초 이상 더 소요되는 것을 알 수 있다. 그러나, 객체 데이타베이스인 UniSQL의 경우, 240MB의 데이타에서 SQL Server보다 질의 수행 시간의 증가량이 더 적고, 최고 6배 이상 성능이 좋음을 알 수 있다. 그리고, 실험환경-2에서 질의유형-2의 처리 성능도 실험환경-1과 같은 경향을 보일 것으로 판단된다.

실험결과를 고찰해 본 결과, 경로식 처리가 필수적인 XML 질의를 수행할 때, 경로식에 사용된 엘리먼트의 탐색을 조인으로 변환하는 관계 데이타베이스에 비해 객체 참조를 사용하는 객체 데이타베이스가 질의 처리 성능이 좋음을 알 수 있었다. 그리고, 제안한 방법은 데이타 양에 민감하게 영향받지 않는 XML 질의 처리 수행 성능을 갖기 때문에 XML 데이타를 관리하는데 효과적인 방법임을 알 수 있다.

5. 결론 및 향후 연구

본 논문에서는 UniSQL ORDBMS를 이용한 DTD 기반 XML 데이타 관리 시스템 Xing을 제안하였다. Xing 시스템에서는 객체 데이타베이스의 객체 참조와 다중값 속성을 이용하여 XML DTD로부터 객체 스키마를 생성하는 작업이 간단하게 이루어진다. 그리고, DTD 의존적 객체 스키마를 생성하므로 XML 데이타 저장 알고리즘이 간단하고, XML 질의어를 OQL로 변환하는 것이 용이하다.

Xing 시스템을 유전자 데이타베이스 등 대규모의 XML 데이타를 저장하고 관리하는데 적합한 시스템이다. 이를 위하여 SAX 파서를 이용한 메모리 부하가 적은 고유의 Xing 트리를 생성하고, 이를 이용하여 대량의 XML 데이타를 빠르게 자동으로 저장하도록 하였다. 또한, 질의결과를 XML로 반환함으로써 완전한 XML 데이타 관리시스템으로 사용할 수 있다.

Xing 시스템의 유용성을 입증하기 위하여 대표적인 생물 정보 사이트인 GenBank로부터 대규모 XML 유전체 데이타를 다운로드 받아서 Xing 시스템으로 저장·관리하면서 시스템의 유용성과 질의 성능을 분석하였다. 특히, 관계 데이타베이스 형태로 저장·관리하는 경우보다 객체 데이타베이스를 사용하는 경우에 XML 질의 처리 성능이 더욱 뛰어남을 확인하였다. 본 시스템은 경로 질의를 수행할 때 객체 참조관계를 이용하기 때문에 조인을 이용한 관계 데이타베이스보다 뛰어난 성능을 나타내었다. 더구나, Xing 시스템은 관계 데이타베이스를 이용하는 경우와 달리 데이타베이스의 규모와 질의 선택률의 변화에 거의 무관하게 뛰어난 성능을 제공함

을 알 수 있었다.

향후 연구로는 경로 질의 처리에서 성능 향상을 위하여 새로운 인덱스 구조를 고안할 예정이다. 그리고, XML 질의 처리 기능 확장을 위해 다중값 연산을 효율적으로 처리할 수 있는 매소드도 연구할 예정이다.

참 고 문 헌

[1] W3C XML Query,http://www.w3c.org/XML/Query/

[2] K. Runapongsa and J. M. Patel, "Storing and Querying XML Data in Object-Relational DBMSs," *EDBT Workshop*, pp.266-285, 2002.

[3] 정태선, 박상원, 한상영, 김형주, "XML 데이타를 위한 객체지향 데이타베이스 스키마 및 질의처리", *한국정보과학회 논문지 : 데이타베이스*, 29(2), 2002.

[4] S. Abiteboul, P. Buneman. and D. Suciu, *Data on the web: from relations to semistructured data and XML*, Morgan Kaufmann Publisher, Los Altos, CA114022, USA, 1999.

[5] I. Tatarinov and S. D. Viglas, "Storing and Querying Ordered XML Using a Relational Database System," In *Proc. Intl. Conf. on Management of Data, ACM SIGMOD*, 2002.

[6] XML-DBMS, http://www.rpbourret.com/xmldbms/

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

[8] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. "Lore: A Database Management System for Semistructured Data," *ACM SIGMOD Record*, 26(3):54-66, September 1997.

[9] UniSQL, http://www.unisql.com/

[10] N. Idris, "Should I use SAX or DOM?," http://www.xml.com/pub/SAX_vs_DOM, 2000.

[11] 송정길, *XML 정복을 위한 지름길 XML 프로그래밍*, 생능출판사, pp.273-379, 2003.

[12] NCBI GenBank, http://www.ncbi.nlm.nih.gov/

[13] Tamino XML database, http://www.softwareag.com/tamino/

[14] eXcelon Inc., http://www.exceloncorp.com/

[15] J. McHugh and J. Widom., "Query Optimization for XML," In *Proc. Intl. Conf. on the 25th VLDB*, 1999.

[16] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. "The Lorel Query Language for Semistructured Data," *International Journal on Digital Libraries*, 1(1):68-88, April 1997.

[17] XML-QL, http://www.w3.org/TR/NOTE-xml-ql/

[18] A. Bonifati and S. Ceri, "Comparative Analysis of Five XML Query Languages," *ACM SIGMOD Record*, 29(3), pp.76-87, 2000.

[19] XQL, http://www.w3.org/TandS/QL/QL98/pp/xql.html

[20] XPath, http://www.w3.org/TR/xpath.html

[21] E. Bertino and W. Kim, "Indexing Techniques for Queries on Nested Objects," *IEEE Trans. on Knowledge and Data Engineering*, 1(2), pages 196-214, June 1989.

[22] R. Emasri and S. Navathe, *Fundamentals of Database Systems*, Addison-Wesley, 3rd edition, 2000.

(부록 1) GBSeqXML.dtd

```
<!-- ================================== -->
<!-- This section mapped from ASN.1 module NCBI-GBSeq -->
<!-- ================================== -->
<!-- Definition of GBSeq -->
<!--
**************************************************
ASN.1 and XML for the components of a GenBank format
sequence
J.Ostell 2002
**************************************************
-->
<!ELEMENT GBSeq (
GBSeq_locus ,
GBSeq_length ,
GBSeq_strandedness? ,
GBSeq_moltype? ,
GBSeq_topology? ,
GBSeq_division ,
GBSeq_update-date ,
GBSeq_create-date ,
GBSeq_definition ,
GBSeq_primary-accession? ,
GBSeq_accession-version? ,
GBSeq_other-seqids? ,
GBSeq_secondary-accessions? ,
GBSeq_keywords? ,
GBSeq_segment? ,
GBSeq_source ,
GBSeq_organism ,
GBSeq_taxonomy ,
GBSeq_references ,
GBSeq_comment? ,
GBSeq_primary? ,
GBSeq_source-db? ,
GBSeq_feature-table? ,
GBSeq_sequence? ,
GBSeq_contig? )>
<!ELEMENT GBSeq_locus ( #PCDATA )>
<!ELEMENT GBSeq_length ( %INTEGER; )>
<!ELEMENT GBSeq_strandedness ( %INTEGER; )>
<!ATTLIST GBSeq_strandedness value ( not-set
|single-stranded |double-stranded |mixed-stranded )
#IMPLIED >
<!ELEMENT GBSeq_moltype ( %INTEGER; )>
<!ATTLIST GBSeq_moltype value ( nucleic-acid |dna
|rna |trna |rrna |mrna |urna |snrna |snorna |peptide
) #IMPLIED >
<!ELEMENT GBSeq_topology ( %INTEGER; )>
<!ATTLIST GBSeq_topology value ( linear |circular )
#IMPLIED >
<!ELEMENT GBSeq_division ( #PCDATA )>
<!ELEMENT GBSeq_update-date ( #PCDATA )>
<!ELEMENT GBSeq_create-date ( #PCDATA )>
<!ELEMENT GBSeq_definition ( #PCDATA )>
<!ELEMENT GBSeq_primary-accession ( #PCDATA )>
<!ELEMENT GBSeq_accession-version ( #PCDATA )>
```

```
<!ELEMENT GBSeq_other-seqids ( GBSeqid* )>
<!ELEMENT GBSeq_secondary-accessions (
GBSecondary-accn* )>
<!ELEMENT GBSeq_keywords ( GBKeyword* )>
<!ELEMENT GBSeq_segment ( #PCDATA )>
<!ELEMENT GBSeq_source ( #PCDATA )>
<!ELEMENT GBSeq_organism ( #PCDATA )>
<!ELEMENT GBSeq_taxonomy ( #PCDATA )>
<!ELEMENT GBSeq_references ( GBReference+ )>
<!ELEMENT GBSeq_comment ( #PCDATA )>
<!ELEMENT GBSeq_primary ( #PCDATA )>
<!ELEMENT GBSeq_source-db ( #PCDATA )>
<!ELEMENT GBSeq_feature-table ( GBFeature* )>
<!--
Optional for other dump forms
-->
<!ELEMENT GBSeq_sequence ( #PCDATA )>
<!ELEMENT GBSeq_contig ( #PCDATA )>
<!-- Definition of GBSecondary-accn -->
<!ELEMENT GBSecondary-accn ( #PCDATA )>
<!-- Definition of GBSeqid -->
<!ELEMENT GBSeqid ( #PCDATA )>
<!-- Definition of GBKeyword -->
<!ELEMENT GBKeyword ( #PCDATA )>
<!-- Definition of GBReference -->
<!ELEMENT GBReference (
GBReference_reference ,
GBReference_authors? ,
GBReference_consortium? ,
GBReference_title? ,
GBReference_journal ,
GBReference_medline? ,
GBReference_pubmed? ,
GBReference_remark? )>
<!ELEMENT GBReference_reference ( #PCDATA )>
<!ELEMENT GBReference_authors ( GBAuthor* )>
<!ELEMENT GBReference_consortium ( #PCDATA )>
<!ELEMENT GBReference_title ( #PCDATA )>
<!ELEMENT GBReference_journal ( #PCDATA )>
<!ELEMENT GBReference_medline ( %INTEGER; )>
<!ELEMENT GBReference_pubmed ( %INTEGER; )>
<!ELEMENT GBReference_remark ( #PCDATA )>
<!-- Definition of GBAuthor -->
<!ELEMENT GBAuthor ( #PCDATA )>
<!-- Definition of GBFeature -->
<!ELEMENT GBFeature (
GBFeature_key ,
GBFeature_location ,
GBFeature_intervals? ,
GBFeature_quals? )>
<!ELEMENT GBFeature_key ( #PCDATA )>
<!ELEMENT GBFeature_location ( #PCDATA )>
<!ELEMENT GBFeature_intervals ( GBInterval* )>
<!ELEMENT GBFeature_quals ( GBQualifier* )>
<!-- Definition of GBInterval -->
<!ELEMENT GBInterval (
GBInterval_from? ,
GBInterval_to? ,
GBInterval_point? ,
GBInterval_accession )>
<!ELEMENT GBInterval_from ( %INTEGER; )>
<!ELEMENT GBInterval_to ( %INTEGER; )>
<!ELEMENT GBInterval_point ( %INTEGER; )>
<!ELEMENT GBInterval_accession ( #PCDATA )>
<!-- Definition of GBQualifier -->
<!ELEMENT GBQualifier (
GBQualifier_name ,
GBQualifier_value? )>
<!ELEMENT GBQualifier_name ( #PCDATA )>
<!ELEMENT GBQualifier_value ( #PCDATA )>
<!-- Definition of GBSet -->
<!ELEMENT GBSet ( GBSeq+ )>
```


이 경 희

1995년 2월 충북대학교 컴퓨터과학과(학사). 1999년 2월 충북대학교 대학원 전자계산학과(석사). 2004년 2월 충북대학교 대학원 전자계산학과(박사). 2002년 9월~현재 서원대학교 교양학부 강의전담교수. 관심분야는 XML, 객체데이타베이스, 데이타마이닝, 바이오인포매틱스

김 태 경

2002년 충북대학교 경영정보학과(학사) 2002년~현재 충북대학교 정보산업공학과 석사과정 재학중. 관심분야는 XML, OODB, 바이오인포매틱스, OLAP

김 선 신

1989년 2월 충북대학교 물리학과 졸업 1995년 2월 충북대학교 물리학과 석사 2002년 5월 Syracuse University computer science 석사. 2003년 3월~현재 충북대학교 전자계산학과 박사과정. 관심분야는 바이오인포매틱스, 알고리즘

이 충 세

1990년 University of South Carolina 전산학 박사. 1983년~1987년 University of North Dakota 전산학 조교수. 1989년~1991년 동아대학교 경영정보학과 부교수. 1991년 9월~현재 충북대학교 컴퓨터과학과 교수. 관심분야는 결함허용, 알고리즘, 전문가시스템

조 완 섭

1985년 경북대학교 통계학과(학사). 1987년 한국과학기술원 전산학과(석사). 1996년 한국과학기술원 전산학과(박사). 2002년~2003 미국 Univ. of Florida Post Doc. 연구원. 1987년~1990년 한국전자통신연구원 컴퓨터연구소 연구원. 1997년~현재 충북대학교 경영정보학과 교수. 관심분야는 데이타베이스, 데이타마이닝, XML, 바이오인포매틱스