관계 DBMS의 실체뷰 기능을 이용한 XML 실체뷰 지원

Supporting XML Materialized Views Using Materialized Views of RDBMS

김승훈 (Seunghun Kim)*, 강현철 (Hyunchul Kang)**

초 록

XML이 웹상에서 데이타 교환의 표준으로 등장한 이래 웹 환경에서 e-Commerce와 같은 웹 기반 비지니스 응용을 효율적으로 지원하기 위해 XML 웨어하우스 기술이 요구되고 있다. 관계 DMBS가 XML 웨어하우스의 저장소로 사용될 경우, XML 웨어하우스의 XML 실체뷰는 관계 DBMS의 관계 실체뷰를 이용하여 제공할 수 있다. XML 문서가 관계 튜플로 저장되기 때문에 XML 실체뷰를 정의하는 XML 질의는 SQL로 변경된다. 만일 변경된 SQL 문으로 관계 실체뷰를 정의하면, XML 실체뷰는 해당 관계 실체뷰를 구성하는 튜플들에 대한 XML 태깅만으로 얻어질 수 있다. 이런 기법의 가장 큰 장점은, 소스 XML 문서가 변경될 때마다 XML 태깅을 제외한 XML 실체뷰의 일관성 유지를 관계 DBMS가 수행해준다는 것이다. 본 논문에서는 이러한 XML 실체뷰 기법을 제시하고 Windows 2000 Professional 환경에서 실체뷰 기능을 갖춘 상용 관계 DBMS를 사용하여 Java로 구현하였다. 성능 실험은 웹상의 e-Commerce 벤치마크인 TPC-W의 XML 문서를 대상으로 수행하였다. 실험 결과 본 논문이 제시한 XML 실체뷰 기법이 매우 효율적인 것으로 나타났다.

ABSTRACT

Since the emergence of XML as the standard for data exchange on the Web, XML warehousing technology is required to efficiently support Web business applications such as *e-Commerce*. When the *RDBMS* is employed as the storage for XML warehouse, *XML materialized views* of the XML warehouse could be provided by leveraging the materialized views of the RDBMS. Because XML documents are mapped into relational tuples, an XML query defining an XML materialized view needs to be transformed into SQL. If relational materialized views were defined with the transformed SQL statements, the XML materialized view could be obtained just by XML-tagging the tuples of the corresponding relational materialized views. The foremost advantage of such a scheme is that the RDBMS does take care of XML materialized view consistency except XML tagging whenever their source XML documents are updated. In this paper, we proposed such a scheme of providing XML materialized views, and implemented it using a commercial RDBMS equipped with materialized view facility in Java on Windows 2000 Professional environment. XML documents in TPC-W, Web e-Commerce Benchmark, were used in performance experiments. The experimental results showed that our proposed scheme for XML materialized views was very effective.

키워드 : e-Commerce, XML, XML 웨어하우스, XML 실체뷰, 관계 실체뷰
e-Commerce, XML, XML warehouse, XML materialized view, relational materialized view

---

이 논문은 2006년도 중앙대학교 학술연구비(일반연구비) 지원에 의한 것임.

* 중앙대학교 대학원 컴퓨터공학과 졸업(석사)
** 중앙대학교 컴퓨터공학부 교수

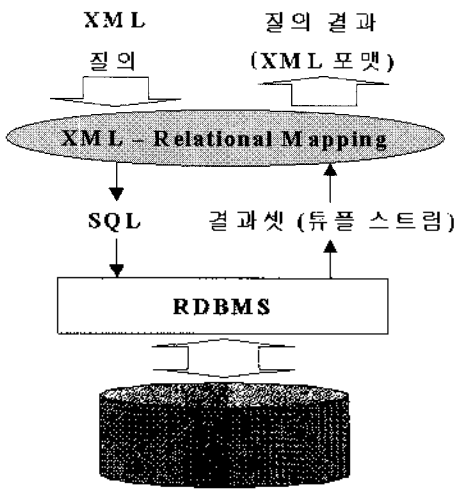
1. 서 론

XML이 웹 상에서 데이터 교환의 표준으로 부각된 이래 웹 상에서 XML 문서가 차지하는 비중이 점점 커져가고 있다. 비즈니스, 공학, 과학 등 여러 컴퓨너티들은 이미 자신의 데이터 표현과 교환을 위한 표준을 제공하고자 DTD, XML Schema, 또는 여타 명세에 의해 XML 데이터의 스키마를 정의해 오고 있다. 예를 들어, 비즈니스 컴퓨너티의 경우, 다양한 산업계 조직과 기관들에 의해 정의된 XML 스키마를 OASIS의 XML.org Registry에 등록함으로써, 각 업종 별로 다양한 응용을 지원하기 위한 컴퓨너티 표준으로 활용하고 있다 [1]. 앞으로 웹에서 교환되는 데이터의 대부분은 일정 스키마를 준수하는 XML 형식이 될 것이며 이들은 스키마 별로 분류되어 XML 웨어하우스를 형성할 것으로 예측된다. 이미 프랑스의 INRIA에서는 Xyleme이라는 XML 웨어하우스를 웹 상에서 서비스하는 기술을 개발하여 상업화하였다 [2]. XML 응용의 대표적인 예로 e-Commerce 등의 웹 응용을 들 수 있는데, 이들 응용은 기존의 비즈니스 응용이 구축하던 데이터 웨어하우스와 유사하게 웹 상에서 XML 데이터의 웨어하우스를 필요로 한다.

XML이 1998년 W3C에 의해 웹에서 데이터 교환의 표준으로 제시된 이래 XML 데이터의 저장 및 검색 기법에 관한 연구가 활발히 수행되고 있다. XML 데이터의 저장 문제는, 그동안 가장 널리 사용되어 왔고 폭넓게 확산되어 있는 관계 DBMS에 분할 저장하는 기법이 그 실용적 중요성으로 인해 가장 대표적인 저장 기법으로 주목받고 있다 [3-6]. 앞으로 XML 데이터는 점점 더 많이 관계 DBMS에 저장될 것으로 예측되고 있으며, 이미 Oracle, DB2, 그리고 SQL Server 등 주요 관계 DBMS 개발사들은 XML을 지원하는 버전을 출시하였고 그 기능을 강화해가고 있다 [7]. XML 데이터를 관계 DBMS에 저장할 경우 기술적인 핵심 문제는 (1) XML 데이터를 관계 튜플로 사상(mapping)하여 저장할 테이블 스키마의 설계, (2) XML 질의의 SQL로의 변환, 그리고 (3) SQL 질의 결과의 XML로의 변환이다. <그림 1>은 이들 중 두 번째와 세 번째의 변환을 나타낸 것이다. XML 질의가 주어지면 이는 먼저 질의의 소스 XML 데이터가 저장된 테이블(들)과 그 스키마에 따라 SQL 질의로 변환된다. 이 SQL 문이 수행되어 결과 셋 (혹은 튜플 스트림)을 생성하게 된다. 이 SQL 결과 셋에 대해서는 XML 질의가 요구하는 결과 형태를 갖추기 위하여 XML 태깅(tagging)이 이루어져 최종 XML 질의 결과인 XML 데이터로 변환된다.

본 논문은 웹 상의 XML 웨어하우스에서 XML 실체뷰(materialized view)를 지원하는 기법에 관한 것으로 XML 웨어하우스의 저장소로 관계 DBMS를 사용하는 경우를 고려하였다. 상용 관계 DBMS들은 종래의 데이터 웨어하우스를 효율적으로 지원하기 위한 관계 실체뷰 기능을 엔진과 관련 툴들을 통하여 다양하게 제공하고 있다. 본 논문에서는 이러한 상용 관계 DBMS의 실체뷰 기능을 이용하여 XML 웨어하우스에서 XML 실체뷰를 효율적으로 지원하는 시스템을 제시한다.

관계 DBMS를 저장소로 사용하여 XML



〈그림 1〉 관계 DBMS 기반 XML 저장 시스템에서 XML 질의 처리

웨어하우스를 구축하였을 때, XML 뷰는 XQuery 또는 XPath와 같은 표준 XML 질의에 의해 정의된다. 이 XML 질의가 그림 1에서와 같이 SQL 질의로 변환되어 처리될 때 변환된 SQL 문으로 관계 실체뷰를 정의하여 이를 관계 DBMS 엔진의 실체뷰 기능과 관련 툴을 이용하여 관리하게 함으로써 XML 실체뷰를 효율적으로 지원할 수 있다. XML 실체뷰는 해당 관계 실체뷰의 튜플들을 모두 검색한 후 XML 태깅 과정을 거쳐 생성할 수 있다. 한편, XML 실체뷰를 제공할 때 요구되는 실체뷰의 일관성 유지 문제 즉, XML 웨어하우스의 소스 XML 문서에 변경이 발생하였을 때 관련 XML 실체뷰에 반영하는 문제는, 관계 DBMS 엔진의 실체뷰 지원 기능 및 관련 툴이 담당하여 해결한다. 즉, XML 실체뷰의 해당 관계 실체뷰는 그것이 정의될 때 명시된 관리 정책에 따라 소스 XML 데이터의 변경에 대해 점진적으로(incrementally) 또는 그것을 정의하는 SQL 문의 재수행(recomputation)에 의해 갱신(refresh)되고, XML 실체뷰는 그것의 갱신 정책에 따라 (예를 들어, 주기적으로 또는 요구 기반(on-demand)으로) RDBMS에 의해 이미 갱신된 해당 관계 실체뷰의 튜플들을 다시 XML 태깅함으로써 갱신된다. 본 논문에서 제시하는 이와 같은 XML 실체뷰 기법의 가장 큰 장점은 XML 실체뷰의 일관성 유지를 태깅 과정만 제외하고 모두 관계 DBMS가 처리해 준다는 것이다.

본 논문에서는 관계 DBMS의 실체뷰 기능을 이용하여 XML 실체뷰를 제공하는 시스템의 구조를 제시하고, 이를 상용 DBMS 상에서 구현하여, 웹 상의 e-Commerce에 관한 TPC-W 벤치마크 [8]의 XML 문서를 대상으로 수행한 성능 평가 실험 결과를 기술한다.

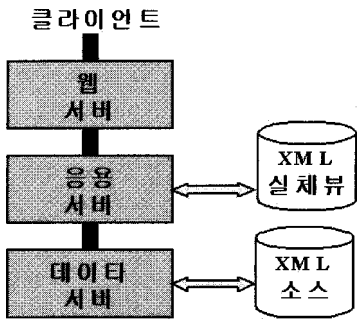
본 논문의 구성은 다음과 같다. 2절에서는 관련 연구를 살펴보고, 3절에서는 관계 실체

뷰 기능을 이용한 XML 실체뷰 시스템을 제시한다. 4절에서는 제시된 시스템을 이용하여 수행한 성능 평가 실험 결과를 기술한다. 마지막으로 5절에서는 결론을 맺고 향후 연구 내용을 기술한다.

2. 관련 연구

데이터웨어하우스의 실체뷰를 지원하기 위한 연구는 관계 데이타 모델을 대상으로 활발히 수행되었다 [9]. 그 결과 상용 관계 DBMS의 엔진 및 관련 패키지 툴로써 관계 실체뷰 기능이 실용 기술로 제공되기에 이르렀다. 그러나 XML 데이터에 대한 실체뷰 연구는 아직 초보적인 단계이며 실용 기술로 정립되지 못하고 있다. XML 실체뷰의 효용성은, e-Commerce와 같은 XML 기반의 웹 응용을 지원하기 위한 〈그림 2〉와 같은 다층(mutli-tier) 구조에서 소스 데이터에 대한 XML 질의 처리 결과를 응용 서버 층에 실체뷰로 유지함으로써 동일한 또는 유사한 후속 질의 처리에 이용할 수 있다는 데 있다. 웹 환경에서 이러한 XML 실체뷰 기술의 실용적 중요성 때문에 크게 두 종류의 연구가 수행되고 있다.

첫째는 XML 질의가 제기되었을 때 이를 응용 서버 층에 저장되어 있는 관련 실체뷰를 활용하는 새로운 질의로 재작성하는 기법의 연구다. XCacher [10], ACE-XQ [11] 등의 시스템이 제안되었으나 이들 연구에서는 실체뷰의 일관성 유지는 고려하지 않았다. 즉, 실체뷰는 정적(static)이라고 가정하였다. 이는 실체뷰의 소스 XML 데이터에 변경이 발



〈그림 2〉 웹 응용을 위한 다층(multi-tier) 구조 및 XML 실체뷰

생할 경우 그에 영향을 받는 실체뷰를 모두 폐기하고 재생성해야 한다는 것을 의미하므로 실용 기술로서 한계가 있다.

둘째는 XML 소스 데이터의 변경에 대해 실체뷰의 일관성을 유지하는 기법의 연구로 소스 XML 데이터에 대한 변경과 실체뷰와의 연관성을 판단하는 알고리즘과 연관성이 있을 경우 변경 내용을 실체뷰에 점진적으로 반영하는 알고리즘이 제시되었다 [12-14]. 그러나 [12][13]에서 제시된 기법의 경우 XML 실체뷰의 저장 구조로 주기억 장치에 상주하는 트리를 고려하여 웹 환경에서 실용 기술이 되기 위한 요건인 확장성(scalability)을 보장하지 못한다. 웹 응용 환경에서는 실체뷰로 유지하는 것이 효율적인 XML 뷰의 갯수와 XML 데이터 소스의 용량이 모두 클 수 있어 디스크 기반의 실체뷰 저장 구조가 제공되어야 한다. 또한 [14]에서 제시된 기법의 경우, XML 질의 결과를 구성하는 XML 노드 식별자 집합만을 저장하기 때문에 XML 뷰는 완전히 실체화되지 못한 상태로 유지되는 문제점을 가지며 그림 2와 같은 웹 응용 다층 구

조에서 효율적이지 않을 수 있다.

본 논문에서는 XML 실체뷰의 정의 및 일관성 유지를 관계 DBMS로 하여금 담당하게 함으로써 디스크 기반의 실체뷰 저장 구조를 제공하여 웹 환경에서 확장성 있는 XML 실체뷰 기법을 제시한다. 또한 본 논문에서 제시하는 기법은 관계 DBMS가 폭넓게 확산되어 있는 점, 그에 따라 XML 저장에 가장 많이 사용되고 있는 점, 그리고 관계 실체뷰 기능을 이미 실용 기술로 제공하고 있는 점을 감안할 때 실용적인 기술이라 하겠다.

3. 관계 실체뷰를 이용한 XML 실체뷰 시스템

본 절에서는 현재 상용 관계 DBMS가 지원하는 실체뷰 기능을 기술한 후 이를 바탕으로 관계 실체뷰를 이용한 XML 실체뷰 시스템을 제시한다.

3.1. 관계 실체뷰 기능

본 절에서는 대표적인 상용 관계 DBMS인 Oracle과 SQL Server가 지원하는 실체뷰 기능을 기술한다. Oracle은 1992년도부터 select-project 연산으로 정의된 실체뷰를 지원하기 시작하였고, Oracle 8.1부터 조인 연산으로 정의된 실체뷰를 비롯하여 더 다양한 SQL문에 의한 실체뷰를 지원하고 있다 [15]. Oracle 9*i* Release 2부터는 Materialized View 라는 이름으로 실체뷰 기능을 제공하고 있다. 예를 들어, 실체뷰를 생성할 때는 확장된 SQL 문인 CREATE MATERIALIZED VIEW 명령어를 사용한다. 〈그림 3〉과 〈그림 4〉는 Oracle에서 실체뷰를 사용하기 위하여 실체뷰 로그 및

```
CREATE MATERIALIZED VIEW LOG on XMLElem
with ROWID (ename, epath, parentid, content, dtdid, ndfseid, rmeid);
```

〈그림 3〉 Oracle 실체뷰 로그 생성 예

```
CREATE MATERIALIZED VIEW MV_Q1
BUILD IMMEDIATE
REFRESH FAST
ON COMMIT WITH ROWID
AS
select * from xmlelem
where epath like '/order/credit_card/credit_card transaction%'
order by did, eid;
```

〈그림 4〉 점진적 갱신 옵션을 갖는 Oracle 실체뷰의 생성 예

실체뷰를 생성하는 구문을 예를 들어 보인 것이다.[1] 실체뷰 로그는 실체뷰의 소스 테이블에 변경이 발생할 경우 변경 로그 레코드를 기록하는 곳이다. 이는 변경 사항만을 반영하는 실체뷰의 점진적 갱신을 위한 목적으로 생성된다. 만약 점진적 갱신이 아니라 질의 결과를 매번 재구성(rebuild)하여 실체뷰를 유지하는 경우에는 실체뷰 로그는 생성하지 않아도 된다. 〈그림 3〉의 구문은 실체뷰 로그를 주키를 제외한 ROWID를 이용하여 생성하는 예이다. with 절에는 ROWID 또는 PRIMARY KEY가 올 수 있다.

〈그림 4〉는 Oracle 실체뷰의 생성 예로서 그것의 갱신은 점진적으로 수행하도록 명시한 것이다. 각 절의 의미는 다음과 같다.

- **BUILD** 절 : 실체뷰의 내용(instance)을 소스 테이블로부터 검색하는 시점을 지정한다. IMMEDIATE과 DEFERRED 옵션이 있다. 전자는 실체뷰의 생성 시점에, 후자는 나중에 하라는 의미이다.

- **REFRESH** 절 : 실체뷰의 갱신 시점과 방법을 지정한다. On-commit과 On-demand 옵션이 있다. 전자는 소스 테이블의 변경이 승인(commit)될 때를 의미하고, 후자는 사용자가 갱신을 위해 관련 패키지인 DBMS_MVIEW 패키지(REFRESH, REFRESH_ALL_MV, REFRESH_DEPENDENT)를 수행할 때를 의미한다. 한편, 이러한 갱신을 수행하는 방법에는 FORCE, COMPLETE, FAST, NEVER의 네 가지가 있다.
  - **COMPLETE** : 실체뷰의 정의에 따라 그 내용 전체가 새로 구성된다.
  - **FAST** : 점진적으로 갱신되는 방안으로 실체뷰 로그를 사용한다.
  - **FORCE** : 먼저 FAST가 가능한지 점검한 후 가능하면 이를 적용하고, 아니면 COMPLETE을 적용한다.
  - **NEVER** : 갱신하지 않는다.

- **AS** 절 : 일반 뷰나 스냅샷의 경우처럼 실체뷰 정의에 필요한 SQL 문을 기술한다.

한편, Microsoft의 SQL Server는 뷰라는 가상 테이블을 정의하는 기능을 지원해 왔는데, SQL Server 2000에서 질의 처리의 성능 향상을 위해 그 기능을 확장했다. 뷰에서 클러스터되지 않은 인덱스뿐만 아니라 고유 클러스터된 인덱스를 생성하여 복잡한 질의의 처리 성능을 향상시킬 수 있다. SQL Server 2000 Enterprise Edition에서 고유 클러스터된 인덱스가 지원되는 뷰를 인덱스된 뷰(indexed view)라고 한다. 〈그림 5〉는 뷰 V_Q1을 생성하는 예이다. 〈그림 6〉은 〈그림 5〉에서 생성된 뷰 V_Q1의 epath 컬럼에 대해 고유 클러스터된 인덱스를 IV_Q1이라는 이름으로 생성하는 예이다. 이처럼 SQL Server에서는 〈그림 5〉의 예처럼 먼저 뷰를 생성하고 〈그림 6〉의

1) 이들 예에서 사용된 데이타는 웹 e-Commerce 벤치마크인 TPC-W 벤치마크의 것으로서 TPC-W 데이타 Generator를 이용하여 생성한 7개의 XML 문서 유형 중 상품 주문 정보를 표현한 order.xml 문서이다. 이들 order.xml 문서가 분할 저장되는 관계 테이블은 이름이 XMLElem이고 did, eid, ename, epath, parentid, content, dtdid, ndfseid, rmeid 컬럼으로 구성되며, 주키가 {did, eid}이다.

```
CREATE VIEW V_Q1 WITH SCHEMABINDING
AS
select did, eid, epath, ename, parentid, dtdid, content, rmeid, ndfseid
from xmlelem
where epath like '/order/credit_card/credit_card_transaction%';
```

〈그림 5〉 뷰 생성 예

```
CREATE UNIQUE CLUSTERED INDEX IV_Q1
ON V_Q1(epath);
```

〈그림 6〉 생성된 뷰에 대한 고유 클러스터된 인덱스 생성 예

예와 같은 과정으로 인덱스된 뷰를 생성한다.

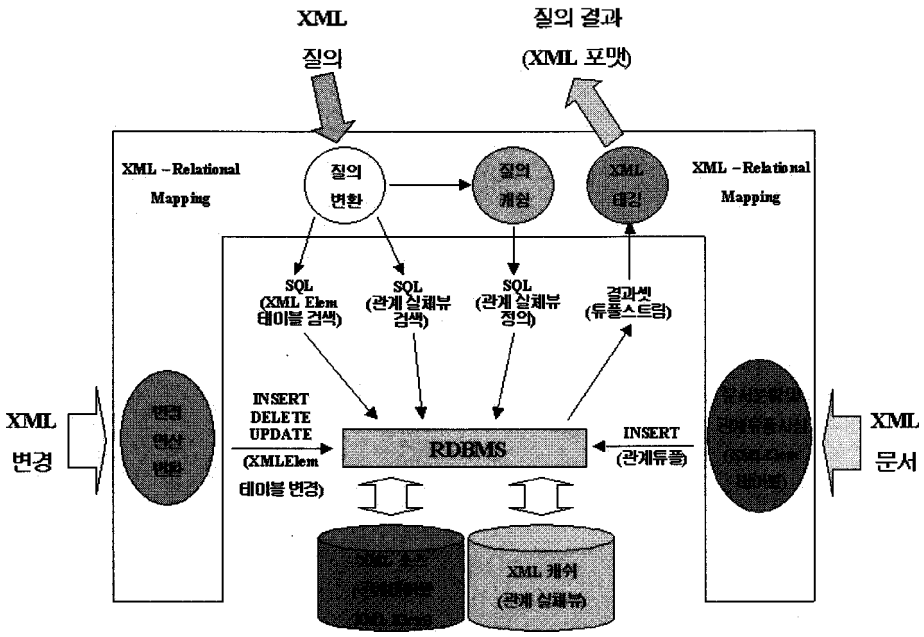
3.2. 관계 실체뷰를 이용한 XML 실체뷰 지원

본 절에서는 관계 실체뷰를 이용한 XML 실체뷰 시스템을 제시한다. 〈그림 7〉은 본 논문에서 제시하여 구현한 XML 실체뷰 시스템의 구조 및 데이타의 흐름을 나타낸 것이다. 이 시스템은 관계 DBMS와 그것의 상부에 위치한 XML-Relational Mapping 모듈로 구성된다. 이 시스템의 기능 및 모듈 컴포넌트에 대해 설명하면 다음과 같다.

XML의 하부 저장소로 관계 DBMS가 이용되고 XML 문서는 엘리먼트 단위로 관계 튜플로 사상되어 SQL의 INSERT 문을 통해 관계 데이터베이스에 저장된다. XML 문서의 파싱과 관계 튜플로의 사상은 XML-Relational Mapping 모듈의 "문서 분할 및 관계 튜플 사상" 컴포넌트에 의해 수행된다. 본 논문에서는 XML 문서의 저장을 위해 [3]에서 제시한 관계 테이블 스키마를 확장한 XMLElem 이라는 이름의 테이블을 사용하였다 (3.1절 참조).

XML 질의가 제기되면 이를 SQL로 변환한다. 이는 XML-Relational Mapping 모듈의 "질의 변환" 컴포넌트가 담당한다. 변환된 SQL 문 중 동일한 것이 자주 제기되는 경우 이를 관계 실체뷰로 정의한다. 이는 XML-Relational Mapping 모듈의 "질의 변환" 및 "질의 캐싱" 컴포넌트가 담당한다. "질의 캐싱" 컴포넌트는 관계 DBMS 엔진에 대하여 관계 실체뷰의 정의를 위한 SQL 문을 제기하며 이때 관계 실체뷰 갱신에 관한 여러 옵션을 설정한다.

변환된 SQL 문이 이미 관계 실체뷰로 정의되어 있느냐의 여부에 따라 XML 질의 처리를 위한 검색 대상이 XMLElem 테이블 혹은 해당 관계 실체뷰가 된다. 이는 XML-Relational Mapping 모듈의 "질의 변환" 컴포넌트가 판단한다. 변환된 SQL 문의 결과가 관계 실체뷰로 정의되어 있지 않으면 변환된



〈그림 7〉 XML 웨어하우스 지원을 위한 관계 실체뷰를 이용한 XML 실체뷰 시스템

SQL 문을 관계 DBMS에 제기한다. 만약 관계 실체뷰로 정의되어 있으면 해당 관계 실체뷰의 튜플 전체를 검색하는 SQL 문을 관계 DBMS에 제기한다. 어느 경우이든 SQL 문의 결과 셋을 구성하는 튜플들(혹은 튜플 스트림)을 XML 태깅하여 XML로 된 최종 질의 결과를 반환한다. XML 태깅은 XML-Relational Mapping 모듈의 "XML 태깅" 컴포넌트가 수행한다.

한편, 이미 관계 데이타베이스에 저장되어 있는 XML 문서에 대한 변경이 요청되면 이를 SQL의 변경문 (INSERT, DELETE, UPDAET 문)으로 변환하여 처리한다. 이는 XML-Relational Mapping 모듈의 "변경 연산 변환" 컴포넌트가 수행한다. 이때, 이들 변경으로 인한 XML 실체뷰의 일관성 유지는 관계 DBMS가 해당 관계 실체뷰를 갱신함으로써 수행된다. 이를 위해 관계 DBMS는 변경에 영향을 받는 관련 실체뷰가 어느 것인지를 판단하게 되고 그들 각각이 정의될 때 설정된 갱신에 관한 옵션에 따른 갱신 과정을 수행하게 된다. 여기서 중요한 점은, XML 실체뷰의 해당 관계 실체뷰 일관성 유지를 관계 DBMS가 담당함으로 인해 동일 기능을 지원하기 위한 컴포넌트를 XML-Relational Mapping 모듈이 제공하지 않아도 된다는 것이다.

4. 구현 및 성능 평가

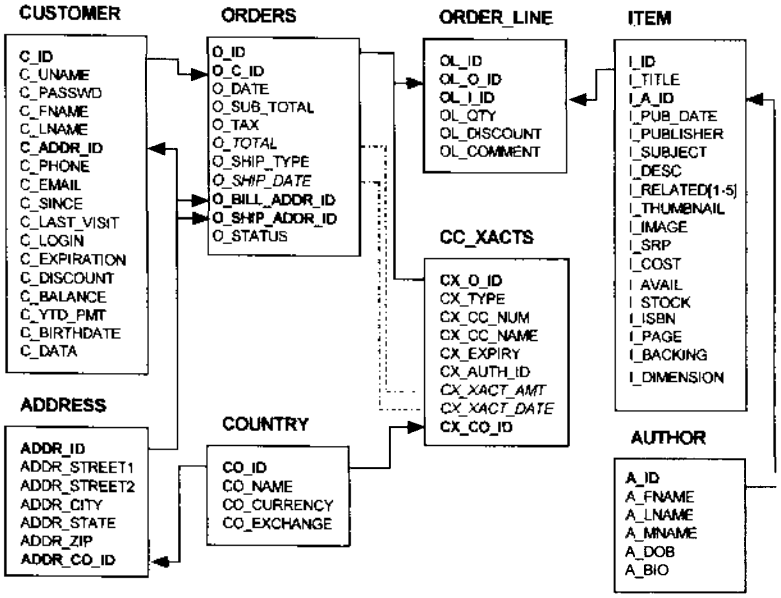
전절에서 제시한 〈그림 7〉의 시스템은 관

계 DBMS로 A사의 제품을 사용하여 Windows 2000 Professional 환경에서 Java로 구현되었다. XML 파서로는 SAX 2.0을 지원하는 Xerces 2 Java Parser 2.4.0 Release [16]를 사용하였다. 본 절에서는 본 논문에서 구현된 시스템을 이용하여 XML 질의가 주어졌을 때 관계 실체뷰를 이용하는 경우와 관계 테이블을 직접 검색하는 경우 (이를 이하, "질의 재수행" 또는 "재수행"이라고 부르자)를 비교하여 본 논문에서 제시한 관계 실체뷰를 이용한 XML 실체뷰 제공이 어느 정도의 성능 향상을 가져오는지 실험을 통하여 평가한 결과를 기술한다.

4.1. 개요

성능 평가를 위한 실험은 Pentium Ⅳ 2.4GHz CPU와 512MB의 메모리가 장착된 Windows 2000 Professional 환경의 시스템에서 수행되었다. 실험 데이타로 사용된 XML 문서는 웹 e-Commerce 벤치마크인 TPC-W 벤치마크 [8]의 상품 주문 정보를 표현한 order.xml 문서들이다. 〈그림 8〉은 TPC-W의 데이터베이스 스키마를 나타낸 것이다. TPC-W의 데이타 Generator는 모두 7 종류의 XML 문서(ADDRESS.XML, AUTHOR.XML, COUNTRY.XML, CUSTOMER.XML, ITEM.XML, ORDER.XML)를 생성해준다. 이중 본 실험에 사용된 order.xml 문서는 복수개의 문서 집합으로 구성되며 각 문서의 평균 크기는 바이트 크기로는 약 2KB, 평균 엘리먼트의 개수로는 약 30개이다. 〈그림 9〉는 본 실험에 사용된 order.xml 문서의 내용 예이다.

실험에 사용된 XML 질의는 상품 주문에



〈그림 8〉 TPC-W 데이타베이스 스키마

```
<?xml version = "1.0" encoding = "UTF-8"?>
<order id = "1"><customer_id>1704</customer_id>
<order_date>2003-06-28</order_date>
<subtotal>6089.80</subtotal>
<tax>502.41</tax>
<total>6756.21</total>
<ship_type>UPS</ship_type>
<ship_date>2003-06-30</ship_date>
<bill_address_id>2369</bill_address_id>
<ship_address_id>1828</ship_address_id>
<order_status>PENDING</order_status>
<customer_info>
<customer_name>D. Robert</customer_name>
<customer_addr>California</customer_addr>
<customer_tel>1-619-882-5745</customer_tel>
</customer_info>
<credit_card>
<credit_card_transaction><credit_card_type>MASTERCARD</credit_card_type>
<credit_card_number>0276899574633526</credit_card_number>
<name_on_credit_card>close escapades </name_on_credit_card>
<expiration_date>2005-05-09</expiration_date>
<authorization_id>)cb0 : vj-Q : N12u #</authorization_id>
<transaction_amount>6756.21</transaction_amount>
<authorization_date>2003-06-30</authorization_date>
<transaction_country_id>45</transaction_country_id>
</credit_card_transaction>
</credit_card>
<order_lines><order_line id = "1"><item_id>605</item_id>
<quantity_of_item>161</quantity_of_item>
<discount_rate>0.01</discount_rate>
<special_instructions> = vO2WWJ3FQ7KtEcM [VeC : [g</special_instructions>
</order_line>
</order_lines>
</order>
```

〈그림 9〉 order.xml 문서의 예

대한 결제를 신용 카드로 하였을 경우 해당 신용 카드 거래에 대한 정보 (즉, order.xml 문서들 내 credit_card_transaction 엘리먼트)를 검색하는 것이다. 이는 우선 order.xml 문서 중 신용 카드로 결제된 주문에 관한 문서만을 검색해낸 후 검색된 각 문서별로 credit_card_transaction 엘리먼트만 다시 추출하는 것이다. 이 질의를 처리하기 위한 SQL 문으로 관계 실체뷰가 정의되어 관리될 때, XML 변경에 따른 관계 실체뷰의 갱신은 점

〈표 1〉 성능 실험 파라미터

파라미터	설 명
T	저장된 전체 XML 문서 개수
U	질의의 조건을 충족하는 문서의 비율
M	실체뷰의 개수

〈표 2〉 성능 실험 파라미터 값 설정

파라미터	설 정 값		
	T 변화 실험	U 변화 실험	M 변화 실험
T (개)	500, 1000, 1500, 2000, 2500	2000	500
U (%)	1	1, 5, 10, 15, 20	100
M (개)	1	1	0, 2, 4, 6, 8, 10

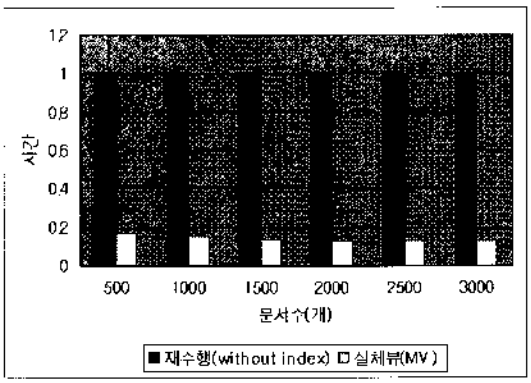
진적으로 수행하는 경우를 고려하였다.

본 성능 평가 실험의 성능 척도는 XML 질의 처리의 성능으로 그 응답시간을 관계 실체뷰를 지원하는 경우와 질의 재수행의 경우로 나누어 측정하였다. 〈표 1〉은 실험을 위한 성능 파라미터를 기술한 것이다. 〈표 2〉는 각 파라미터의 설정값을 실험 유형별로 나타낸 것이다. 질의 응답시간 실험에서는 첫째, 소스 XML 문서수(T)가 증가할 때 관계 실체뷰를 이용하는 경우와 질의를 재수행하는 경우의 질의 응답시간을 측정하였다. 둘째, 소스 XML 문서 중 질의의 조건을 충족하는 XML 문서의 비율(U)이 증가할 때 관계 실체뷰를 이용하는 경우와 질의를 재수행하는 경우의 질의 응답시간을 측정하였다.

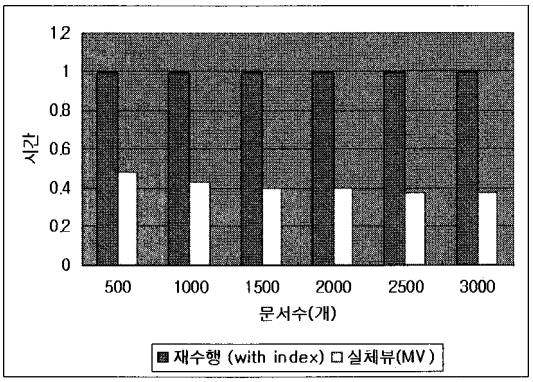
4.2. 실험 결과

4.2.1. 소스 XML 문서 수 변화 실험

〈그림 10〉은 전체 소스 XML 문서 중 1%가 질의의 조건을 충족하는 XML 문서이고 소스 XML 문서 수가 500개부터 3,000개까지 500개씩 증가할 때 질의를 재수행하여 XML 질의 결과를 얻을 때와 해당 관계 실체뷰를 이용해 결과를 얻을 때의 응답시간을 측정한 결과를 나타낸 것이다. 본 결과는 100회의 실험 수행 결과의 평균값이다. (이하 모든 실험 결과도 동일.) 〈그림 10〉의 (a)와 (b)는 각각 질의를 재수행할 때 XML 문서를 저장하고 있는 XMLElem 테이블에 인덱스(DID, EID, EPATH, ENAME, CONTENT 컬럼 각각에 대한)가 있을 경우와 없을 경우에 대한 결과



(a) **XMLElem** 테이블에 **Index**가 없을 경우



(b) **XMLElem** 테이블에 **Index**가 있을 경우

〈그림 10〉 소스 XML 문서 수 변화에 따른 질의 응답시간 (U=1, M=1)

로 질의를 재수행하는 경우의 성능을 1로 정규화한 것이다.

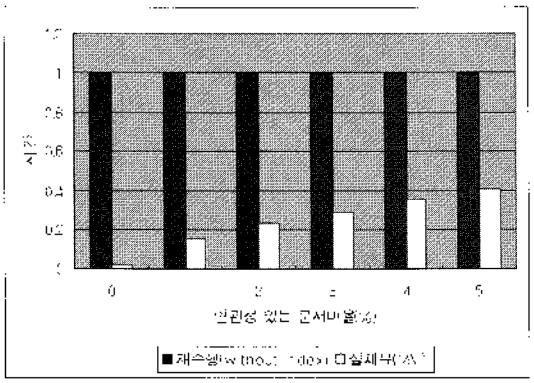
XMLElem 테이블에 인덱스가 없을 경우 관계 실체뷰를 이용했을 때 하부 XML 문서 수 500개의 경우 약 83%의 성능 향상이 있었고, 3,000개의 경우 약 87%의 성능 향상이 있었다. 반면 XMLElem 테이블에 인덱스가 있을 경우에는 하부 XML 문서수가 500개의 경우 약 52%의 성능 향상이 있었고, 3,000개의 경우 약 62%의 성능 향상이 있었다. 이들 성능 향상을 요약하면 〈표 3〉과 같다.

위의 두 실험에서 알 수 있듯이 XMLElem 테이블에 인덱스가 있을 때가 없을 때보다 약 30%의 추가 성능 향상을 보였지만 두 경우 모두 질의를 재수행하는 것보다 관계 실체뷰를 이용함으로써 성능 향상을 가져왔다. 또한 질의의 조건을 충족하는 XML 문서의 비율이 일정할 때 저장된 XML 문서 수가 증가함에 따라 실체뷰 기법의 성능 향상 폭이 증가했다. 이는 질의를 재수행할 경우 검색해야

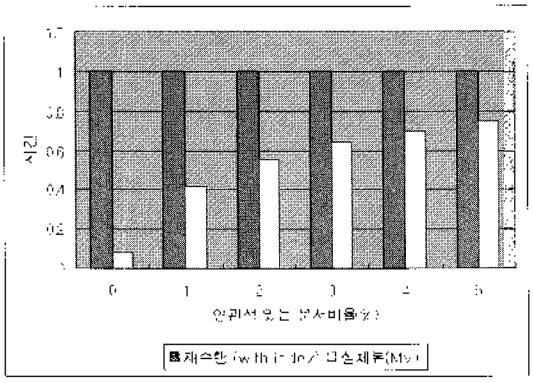
〈표 3〉 소스 XML 문서 수의 변화에 따른 실체뷰를 통한 성능 향상

단위 : %

구분 \ 문서수(개)	500	1000	1500	2000	2500	3000
Index 없을 때	83	85	86	87	87	87
Index 있을 때	52	57	60	60	62	62



(a) XMLElem 테이블에 Index가 없을 경우



(b) XMLElem 테이블에 Index가 있을 경우

〈그림 11〉 질의의 조건을 충족하는 문서 비율 변화에 따른 질의 응답시간 (T=2000, M=1)

될 데이터 양이 늘어남으로 인해 질의 응답시간이 늘어나는 반면, 실체뷰 기법의 경우 전체 XML 문서 수의 1%에 해당하는 XML 문서만이 검색되어 있기 때문에 문서 수 증가에 따른 자연 성능 저하폭이 훨씬 적기 때문이다. 따라서 두 기법 간 질의 응답시간의 차는 전체 문서 수가 커짐에 따라 커지는 것으로 나타났다.

〈표 4〉 질의의 조건을 충족하는 문서 비율의 변화에 따른 실체뷰 기법의 성능 향상

단위 : %

구 분 \ 연관성 문서비율 (%)	0	1	2	3	4	5
Index 없을 때	98	85	77	71	64	59
Index 있을 때	92	58	45	36	30	25

4.2.2. 질의의 조건을 만족하는 XML 문서 비율 변화 실험

〈그림 11〉은 저장된 소스 XML 문서 수가 2,000개이고 질의의 조건을 충족하는 XML 문서의 비율이 0%에서 5%까지 1%씩 증가할 때 응답시간을 측정한 결과를 나타낸 것이다. 〈그림 11〉의 (a)는 XMLElem 테이블에 인덱스가 없을 경우 질의의 조건을 충족하는 XML 문서 비율이 증가할 때 응답시간을 비교한 것이다. 관계 실체뷰를 이용했을 때 질의의 조건을 충족하는 XML 문서의 비율이 0%일 경우 약 99.8%의 성능 향상이 있었고, 5%일 경우 약 59%의 성능 향상이 있었다. 반면 〈그림 11〉의 (b)는 XMLElem 테이블에 인덱스가 있을 경우의 결과인데, 관계 실체뷰를 이용했을 때 질의의 조건을 충족하는 XML 문서의 비율이 0%일 경우 약 98%의 성능 향상이 있었고, 5%일 경우 약 25%의 성능 향상이 있었다. 이들 성능 향상을 요약하면 〈표 4〉와 같다.

위의 두 실험에서 알 수 있듯이 XMLElem 테이블에 인덱스가 있을 때와 없을 때 모두 질의를 재수행하는 것보다 관계 실체뷰를 이용할 때 성능 향상을 가져왔다. 그러나 소스 XML 문서 수가 일정할 때 질의의 조건을 충족하는 XML 문서의 비율이 증가함에 따라 실체뷰 기법의 성능은 저하되었고 따라서 재수행에 대한 실체뷰 기법의 성능 향상 폭은 감소했다. 이는 질의 재수행의 검색 대상은 소스 XML 문서 수가 동일하기 때문에 항상 일정한 반면 질의의 조건을 충족하는 XML 문서 비율이 커짐에 따라 실체뷰가 커지기 때문이다.

5. 결 론

비즈니스 응용을 위한 데이터 웨어하우스 환경에서 실체뷰 기법은 질의 처리의 성능 제고를 위한 중요한 기술이다. 이미 대표적인 상용 관계 DBMS들은 엔진과 툴을 통해 실체뷰 기능을 실용 기술로 제공하고 있다. 본 논문에서는 e-Commerce 등 웹 상의 비즈니스 응용의 지원을 위해 웹 상의 데이터 교환 표준인 XML로 표현된 비즈니스 데이터를 웹 상에서 웨어하우싱했을 때 자주 제기되는 XML 질의에 대한 실체뷰를 관계 DBMS의 실체뷰 기능을 활용하여 지원하는 기법을 제시하였다.

제시한 기법의 효율성은 웹 e-Commerce 벤치마크인 TPC-W 벤치마크의 상품 주문에 관한 XML 문서 (order.xml)를 대상으로 수

행한 성능 평가 실험 결과 확인할 수 있었다. 현재까지 제시된 XML 저장 기술로 가장 대표적인 것이 현재 가장 널리 사용되고 있는 관계 DBMS를 이용하는 기법이라는 점과 이미 상용 DBMS 제품들이 관계 실체뷰 기능을 실용 기술로 다양하게 제공하고 있다는 두 가지 점을 고려할 때 본 논문에서 제시한 기법은 XML 실체뷰 지원을 위한 실용적인 솔루션이다.

향후 연구로는, (1) 현재 상용 관계 DBMS에서 제공되고 있는 실체뷰 기능은 비지니스 데이터가 관계 모델로 표현된 경우를 주로 고려하여 개발된 것인데, 웹 상의 비즈니스 응용에서 XML 질의를 SQL로 변환했을 때 나타나는 XML 질의의 특성을 감안하여 그 기능을 더욱 강화하거나 보완하는 문제와 (2) 관계 실체뷰의 갱신 및 그에 따른 관련 XML 실체뷰의 태깅을 통한 갱신을 XML 질의 및 변환된 SQL 질의의 특성에 따라 양자 간에 연계하여 관리할 수 있는 다양한 정책들을 설정할 수 있는 툴을 제공하는 문제, 그리고 (3) 현재 상용 XML-Enabled DBMS에서 제공하는 XML data Type과 SQL/XML [17] 기능과의 연동 문제 등을 들 수 있다.

참 고 문 헌

[1] XML.org, http://www.xml.org

[2] Xyleme, http://www.xyleme.com

[3] D. Florescu and D. Kossmann, "Storing and Querying XML Data Using an RDBMS," IEEE Data Engg. Bulletin, Sep. 1999. pp. 27-34.

[4] J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, D. DeWitt, and J. Naughton, "Relational Databases for Querying XML Documents : Limitations and Opportunities," Proc. Int'l Conf. on VLDB, 1999.

[5] M. Yoshikawa, T. Amagasa, T. Shimura, S. Uemura, "XRel : A Path-Based Approach to Storage and Retrieval of XML Documents Using Relational Databases,"ACM Trans. on Internet Technology, Vol. 1, No. 1, Aug. 2001, pp. 110-141.

[6] I. Tatarinov et al., "Storing and Querying Ordered XML Using a Relational Database System," Proc. ACM SIGMOD Int'l Conf. on Management of Data, 2002. pp. 204-215.

[7] A. Chaudhri et al. (Eds.), "XML Data Management : Native XML and XML-Enabled Database Systems," Addison-Wesley, 2003.

[8] TPC-W, Transaction Processing Performance Council, http://www.tpc.org/tpcw/.

[9] A. Gupta and I. Mumick, "Materialized Views : Techniques, Implementations, and Applications," MIT Press, 1999.

[10] V. Hristidis and M. Petropoulos, "Semantic Caching of XML Databases,"

Proc. Int'l Workshop on the Web and Databases, 2002.

[11] L. Chen and E. Rundensteiner, "ACE-XQ : A CachE-aware XQuery Answering System," Proc. Int'l Workshop on the Web and Databases, 2002.

[12] L. Quan et al., "Argos : Efficient Refresh in an XQL-Based Web Caching System," Proc. Int'l Workshop on the Web and Databases, 2000, pp. 23-28.

[13] K. Dimitrova et al., "Order-sensitive View Maintenance of Materialized XQuery Views," Tech. Rep. WPI-CS-TR-03-17, Comp. Sci. Dept., Worcester Polytechnic Institute, 2003.

[14] A. Sawires et al., "Incremental Maintenance of Path-Expression Views," Proc. ACM SIGMOD Int'l Conf. on Management of Data, 2005.

[15] R. Bello et al., "Materialized Views in Oracle," Proc. Int'l Conf. on VLDB, 1998, pp. 659-664.

[16] http://xml.apache.org.

[17] ISO/IEC FCD 9075-14, Information Technology - Database languages - SQL - part 14 : XML-Related Specifications (SQL/XML), http://www.ansi.org

저 자 소 개

김승훈 (E-mail : kimsh@dblab.cse.cau.ac.kr)
2002. 경주대학교 컴퓨터공학과 졸업(학사)
2004. 중앙대학교 대학원 컴퓨터공학과 졸업(석사)
관심분야 XML 데이터베이스, 임베디드 시스템

강현철 (E-mail : hckang@cau.ac.kr)
1983. 서울대학교 컴퓨터공학과 졸업(학사)
1985. U. of Maryland at College Park, Computer Science(M.S.)
1987. U. of Maryland at College Park, Computer Science(Ph.D.)
1988 ~ 현재 중앙대학교 컴퓨터공학부 교수
관심분야 XML 데이터베이스, 웹 데이터베이스,
스트림 데이터 관리, 이동 데이타베이스