실체뷰 지원 XML 저장 시스템의 구현 및 성능 평가*

성호상° 강현철
중앙대학교 컴퓨터공학과
hssung @dblab.cse.cau.ac.k hckang@cau.ac.kr

Implementation and Performance Evaluation of the XML Storage System Supporting Materialized Views

HoSang Sung° Hyunchul Kang
Dept. of Computer Science and Engineering, Chung-Ang University

요 약

실체뷰는 질의 처리 성능 향상을 위한 수단으로 널리 연구되어 왔다. 이러한 실체뷰는 하부 데이타가 변경되었을 경우에 일관성을 유지해야 하는데, 그 기법으로는 뷰를 하부 데이타로부터 재생성하는 방법과 변경 내용 중 뷰와 관련 있는 것만 반영하는 점진적 갱신이 있다. 최근 XML 이라는 웹 문서 표준이 대두되면서 여러 응용 분야에서 이를 활용 하려는 노력이 진행 중이다. 본 논문에서는 XML 문서를 대상으로 하는 XML 실체뷰를 지원하는 XML 저장 시스템의 구현과, 구현된 시스템을 대상으로 실체뷰를 재생성하는 방법과 점진적으로 갱신하는 기법의 성능을 비교 평가한 결과를 기술한다.

1. 서론

XML(Extensible Markup Language)[1] 이란 용어는 인터넷 상에서 가장 흔히 볼 수 있는 단어가 되었다. 비록 XML이 등장한지 몇 년 되지 않았지만 현재 사용되는 여러 응용들을 강력하게 지원하고 있어서 다른 기술들에 비해 급속히 성장하고 있다. 특히 데이타를 관리하고 조직화해야 할 뿐 아니라, 자원의 게시를 위한 여러 웹 사이트나, 이러한 기능을 제공해야 하는 응용들에서 많이 사용되고 있다. 이러한 이유로 여러 DBMS 벤더들도 XML의 강점과 유연성을 인식하고 있으며, 그들의 제품이 XML을 지원할 수 있도록 만들고 있다[2][3][4][5]. 실제 eXcelon 같은 전문 XML 데이타베이스가 이미 존재한다[3]. 이러한 저장 시스템들에서 중요한 이슈 중에 하나는 많은 양의 XML 문서들에 대한 질의 처리 성능이다. 뷰는 데이타 통합과 데이타 여과 기능을 통하여 사용자가 요구하는 데이타를 제공한다. 따라서 XML 문서에 대해서도 유용한 개념이다[6]. 뷰는 질의 처리의 성능 향상을 위해 실체뷰(materialized view)로 유지할 수 있는데, 일관성 유지를 위해서는 뷰를 하부 데이타로부터 재생성하는 방법과 변경 내용 중 뷰와 관련 있는 것만 반영하는 점진적 갱신 기법이 있다.

본 논문에서는 [7]의 연구에 기반을 둔 실체뷰를 지원하는 XML 저장 시스템을 구현하고, 실체뷰를 재생성하는 방법과 점진적으로 갱신하는 기법의 질의 처리 성능을 비교 평가한다.

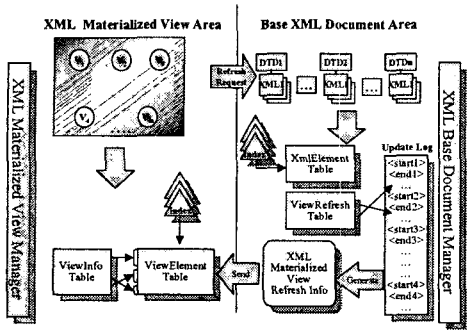
본 논문의 구성은 다음과 같다. 2 절에서는 [7]을 기반으로 점진적 갱신에 기반을 둔 실체뷰 관리 기법을 기술한다. 3 절에서는 실체뷰 지원 XML 저장 시스템의 구현에 대해 기술하고, 4 절에서는 구현된 시스템을 이용하여 실체뷰를 재생성하는 방법과 점진적으로 갱신하는 기법의 성능을 비교 평가한 결과를 기술한다. 5 절에서는 결론을 맺고 향후 과제를 제시한다.

2. 실체뷰 관리 기법

실체뷰를 지원하는 XML 저장 시스템은 (그림 1)과 같이 하부 XML 문서 영역과 XML 실체뷰 영역으로 구성되고 이들 영역은 XML 하부 문서 관리 모듈과 XML 실체뷰 관리 모듈에 의해 각각 관리된다. 하부 XML 문서 영역에는 다수의 DTD 와 XML 문서, 검색을 위한 인덱스, XML 문서의 변경을 기록하기 위한 로그 화일과 각 실체뷰의 점진적 갱신을 위해 필요한 정보가 저장된다. 그리고 실체뷰 영역에는 뷰에 대한 정보와 실체뷰, 그리고 검색을 위한 인덱스가 저장되어 있다. XML 실체뷰의 점진적 갱신은 지연 수행된다. 즉, 하부 XML 문서의 변경이 발생하면 실체뷰에 즉각 반영하지 않고, 변경 내용을 로그 화일에 기록한다. XML 실체뷰 관리 모듈은 사용자로부터 뷰가 요청되었을 경우, XML 하부 문서 관리 모듈에게 해당 뷰에 대한 갱신정보를 요청하게 된다. 하부 문서 관리 모듈은 변경 로그로부터 해당 실체뷰를 갱신하기 위한 갱신정보를 생성하고 이를 반환한다. 그리고 갱신정보를 전송 받은 XML 실체뷰 관리 모듈은 해당 XML 실체뷰에 대한 점진적 갱신을 수행한 후 사용자에게 뷰를 제공한다.

2.1 XML 문서 저장 구조

XML 문서의 저장 구조는 ORDBMS 를 기반으로 하였다. 그리고 문서 정보를 저장하는 Doc_Info 테이블과 DTD 를 저장하고 있는 Dtd_Info 테이블, 그리고 엘리먼트 단위의 XML 문서를 저장하고 있는 XmlElement 라는 세 개의 테이블로 구성되어 있다. (그림 2)는 이들 테이블의 스키마를 나타낸 것이다. XML 문서들은 해당 DTD 를 준수하는 트리구조를 취한다. 그리고 XmlElement 테이블에서 경로 엘리먼트 ID 는 자신이 부모 엘리먼트의 몇 번째 자식 엘리먼트인가를 숫자로 나타내고 이를 부모 엘리먼트 ID 끝에 공백문자를 구분으로 추가하여 구조적 정보를 나타낸다.



(그림 1) XML 실체뷰 관리 기법

Doc_Info(XML 문서 정보 테이블)

DID	Doc_Name	DTDID	createdate
문서 ID	문서명	DTD 번호	저장날짜

Dtd_Info(DTD 정보 테이블)

DTDID	Doc_type	content	contsize	createddate
DTD 번호	문서형식	DTD 내용	DTD 크기	저장날짜

XmlElement(XML 문서 저장 테이블)

DID	DTDID	EID	Ename	Content
문서 ID	DTD 번호	엘리먼트번호	엘리먼트이름	내용

(그림 2) XML 하부 문서 저장 스키마

2.2 실체뷰 저장 구조

본 논문에서는 XML 질의어의 구문으로 표현 가능한 뷰 중에서 단일 DTD 를 대상으로 생성되는 XML 뷰 만을 고려하였다. 또한 뷰의 생성 조건은 DTD 를 구성하는 엘리먼트 중 한번만 나오는 엘리먼트 하나에 대해 명시된다고 가정하였다.

실체뷰는 뷰에 대한 정보와 데이타를 ViewInfo 테이블과 ViewElement 테이블로 구성된다. ViewInfo 테이블에는 각 XML 실체뷰마다 한 개의 레코드가 존재하며 각 XML 실체뷰에 부여되는 뷰의 ID(ViewID), 뷰의 생성 조건을 정의한 뷰 정의(ViewDef), 뷰가 참조하는 DTD의 ID 등으로 구성된다. ViewInfo 테이블은 (그림 3)과 같이 뷰 정의 인 ViewDef의 데이타 타입을 중첩 테이블 형태로 구성하고 있다. ViewElement 테이블은 ViewID, DID, BaseEID, Ename, ViewEID, Content 컬럼으로 구성된다. ViewID는 ViewInfo 테이블의 ViewID를 참조하는 외래키이다. DID는 뷰를 생성한 하부 XML 문서의 ID를 나타낸다. ViewEID는 뷰를 구성하는 엘리먼트의 EID를 나타낸다. BaseEID는 DID가 가리키는 하부 XML 문서의 엘리먼트 ID를 나타낸다. Content에는 실제 XML 뷰를 구성하는 데이타가 저장된다.

ViewInfo(실체뷰 정보테이블)

ViewID	ViewDef	DTDID
뷰 ID	뷰 정의	뷰 참조DTDID

뷰 정의

CE	CC	OP1	OP2	TE	CN
조건엘리먼트	엘리먼트 내용	내용포함, 일치여부	관계연산타입	타겟엘리먼트	문자 개수

ViewElement(실체뷰 엘리먼트테이블)

ViewID	DID	BaseEID	Ename	ViewEID	Content
뷰 ID	하부문서 ID	엘리먼트 ID	엘리먼트 이름	뷰 엘리먼트 ID	내용

(그림 3) XML 실체뷰 스키마

2.3 XML 문서 변경 모델

XML 문서는 문서 단위, 엘리먼트 단위, 속성 단위로 변경될 수 있다. 본 논문에서는 문서 및 엘리먼트 단위의 변경 만을 고려하였다. 즉, 수정은 XML 문서 내에서 값을 갖는 엘리먼트에 대해 엘리먼트 단위로 수행되고, 삽입과 삭제는 문서 단위로 수행된다고 가정하였다.

XML 하부 문서 관리 모듈은 XML 문서가 변경될 때마다 XML 실체뷰의 점진적 갱신을 지원하기 위해서 그 내용을 로그 화일에 기록한다. 변경로그 레코드의 자료구조는 (그림 4)와 같다.

```
<StartUpdateLog, DTDID, DID, ObjType, OpType>
  [<BaseEID, Ename, Content>[,< BaseEID, Ename, Content>, ··· ]]
<EndUpdateLog>
```

(그림 4) 변경로그 레코드 자료구조

변경로그 레코드는 <StartUpdateLog> 필드로 시작해서 <EndUpdateLog> 필드로 끝나는 블록구조를 갖는다. DTDID와 DID는 변경이 발생한 XML 문서의 DTD와 문서 ID를 각각 나타낸다. ObjType은 변경의 단위가 엘리먼트('ELEMENT')인 지 문서('DOCUMENT')인 지를 나타낸다. OpType은 XML 문서에 대한 변경의 종류를 나타낸다. <BaseEID, Ename, Content>는 변경된 엘리먼트에 대한 정보로서 ObjType과 OpType에 따라 생략되거나 한번 또는 그 이상 기록될 수 있다. BaseEID는 변경되는 엘리먼트의 ID를 나타내고, Ename은 변경된 엘리먼트 명을 나타낸다. Content는 엘리먼트의 값을 나타내며, Optype='DELETE'인 경우 Content는 NULL로 설정된다.

2.4 실체뷰의 점진적 갱신

실체뷰의 점진적 갱신은 첫째, 변경로그의 검색, 둘째, 그를 통한 갱신정보의 생성, 그리고 셋째, 갱신정보의 실체뷰로의 반영으로 구성된다. 변경로그의 검색은 이전의 뷰 갱신 이후에 기록된 변경로그 레코드들을 대상으로 수행된다. 새로이 검색해야 할 첫 번째 변경로그 레코드의 위치는 ViewRefresh 테이블에 기록되어 있다. (그림 5)는 XML 실체뷰의 점진적 갱신에서 필요한 정보를 저장하고 있는 ViewRefresh 테이블의 구조를 나타내고 있다. 실체뷰 갱신정보의 생성은 변경로그의 검색을 통해 각 로그 레코드별로 해당 뷰에 영향을 미치는지 여부를 판단하고 <표 1>과 같은 연관성이 있을 경우 해당 실체뷰에 반영해야 할 내용을 갱신정보 레코드로 구성하여 갱신정보에 기록한다. 각 갱신정보 레코드에는 RefType, DID, BaseEID, Ename, ViewEID, Content가 저장된다. RefType은 갱신 유형을 나타내는 것으로 'MODIFY', 'INSERT', 그리고 'DELETE'의 세 가지 유형이 있다. DID는 변경된 문서의 ID, BaseEID는 변경된 하부 문서 엘리먼트의 ID, Ename은 엘리먼트 이름, ViewEID는 삽입될 뷰 엘리먼트의 ID, 그리고 Content는 실체뷰에 반영 될 엘리먼트의 값을 나타낸다. 이들 중 해당 사항이 없는 것은 NULL로 설정된다. 마지막으로 실체뷰에 대한 반영은 생성된 갱신정보를 해당 뷰에 반영하는 것이다.

ViewRefresh(갱신정보 생성을 위한 정보테이블)

ViewID	DTDID	LastULOffset	DIDList
뷰 ID	뷰 관련 DTDID	시작 옵셋 값	관련 문서번호

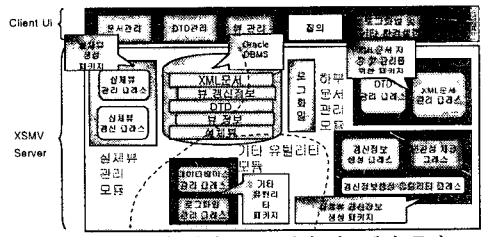
(그림 5) 실체뷰의 점진적 갱신정보를 위한 스키마

<표 1> 연관성 유형

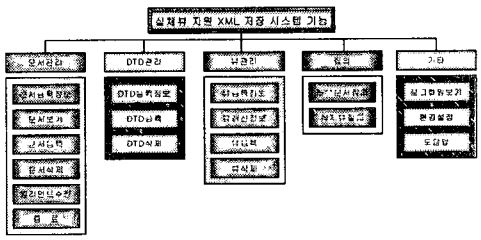
연관성 유형	내용
INSERT	뷰의 조건을 만족하는 문서 삽입
DELETE	뷰의 조건을 만족하던 문서 삭제
MODIFY-M	뷰의 조건을 만족하는 문서의 추출 엘리먼트 수정
MODIFY-I	뷰의 조건을 만족하지 않던 문서의 조건 엘리먼트 값이 수정되어 뷰의 조건을 만족하게 됨
MODIFY-D	뷰의 조건을 만족하던 문서의 조건 엘리먼트 값이 수정되어 뷰의 조건을 만족하지 않게 됨

3. 실체뷰 지원 XML 저장 시스템의 구현

본 논문의 XML 저장 시스템은 DBMS로 Oracle8*i*를 사용하였고, Java로 구현하였다. 실체뷰 지원 XML 저장 시스템은 (그림 6)과 같이 크게 클라이언트 UI(User Interface) 부분과 서버측 기능을 수행하는 XSMV(XML Storage System Supporting Materialized Views)Server로 구성되어 있다. XSMVServer는 하부문서 관리 모듈과 실체뷰 관리 모듈 그리고 기타 유틸리티 모듈로 구성되어 있으며, 각 해당 관리 모듈들은 기능별로 패키지와 클래스로 구성되어 있다. 하부 문서 관리 모듈은 XML 문서를 관리하는 패키지를 가지고 있으며, 이는 XML 문서를 저장하고, 엘리먼트 단위의 수정을 하며, 문서를 삭제하는 등의 기능을 수행한다. 또한, DTD를 관리하는 기능과 변경로그 기록을 담당한다. 실체뷰 갱신정보 생성 패키지는 갱신정보를 생성하기 위한 연관성을 체크하며 생성한 갱신정보를 반환하는 기능을 수행한다. 실체뷰 관리 모듈은 실체뷰 생성 패키지를 가지고 있으며, 뷰의 질의 요청에 대한 응답을 담당하고 해당 뷰에 대한 갱신을 요청하는 기능을 수행한다. 기타 유틸리티 모듈은 데이타베이스 연결에 대한 관리와 로그 화일에 대한 관리를 수행하는 클래스 등으로 이루어져 있다. 클라이언트 UI는 (그림 6)처럼 다섯 가지 메뉴를 가지고 있으며 XSMVServer의 해당 부분의 메소드를 호출함으로써 각각의 기능을 수행한다. (그림 7)은 이들 시스템의 각 기능들을 메뉴별로 나타낸 것이다.



(그림 6) 실체뷰 지원 XML 저장 시스템의 구성



(그림 7) 실체뷰 지원 XML 저장 시스템의 기능

4. 성능 평가

본 논문의 실험에서는 실체뷰를 재생성하는 방법과 점진적으로 갱신하는 기법의 성능을 여러 가지 성능 파라미터들의 변화에 따라 평가한 결과를 기술한다.

4.1 실험 환경 및 데이타

본 논문의 성능 평가를 위한 실험 환경과 실험 데이타는 <표 2>와 같다.

4.2 평가 척도 및 성능 파라미터

본 논문에서는 실체뷰의 일관성 유지를 위하여 실체뷰를 하부 데이타로부터 재생성하는 방법과 변경 내용 중 뷰와 관련있는 것만 반영하여 점진적으로 갱신하는 기법의 성능을 측정한다. 성능 척도는 XML 질의를 제기한 시점부터 질의 결과를 생성 완료하여 디스플레이하기 전까지 시간이다. <표 3>은 실험을 위한 성능 파라미터와 내용을 나타내었다. <표 4>에서는 각 파라미터의 설정치를 실험 유형별로 기록하였다.

<표 2> 실험 환경 및 데이타

항목		환경 내용
실험 데이타		◆ "영화" XML 문서 XML 문서의 평균 엘리먼트 개수: 20 개
뷰 정의		◆ "감독" 엘리먼트 내용이 "장이모" 일때 영화 문서의 "제목", "상영시간", "수입배급사" 로 정의된 실체뷰 생성
OS		Windows 2000 Server
DBMS		Oracle 8i
시스템	CPU	Pentium III 1GHz
	RAM	512 MB

<표 3> 성능 파라미터

파라미터	내용
U	저장 시스템에 저장된 하부 문서 개수
L	하부 문서에 대한 변경 비율
S	뷰 Selectivity(하부 문서 중 뷰 와 연관성 있는 문서 비율)
R	연관성 비율(로그 레코드 중 뷰와 연관성 있는 로그 레코드 비율)
I	변경로그 레코드 중 문서 INSERT 연산 비율
D	변경로그 레코드 중 문서 DELETE 연산 비율
M	변경로그 레코드 중 엘리먼트 MODIFY 연산 비율
MI	연관성 있는 MODIFY 연산 중 Modify-Insert 연관성 비율
MD	연관성 있는 MODIFY 연산 중 Modify-Delete 연관성 비율
MM	연관성 있는 MODIFY 연산 중 Modify-Modify 연관성 비율

<표 4> 파라미터 설정치

파라미터	설정치	
	L 변화 실험	U 변화 실험
U	20000	5000,10000,15000,20000,25000,30000
L	0, 0.1, 0.2, 0.3, 0.4, 0.5	0.2
S	0.2, 0.3	0.2, 0.3
R	0.2, 0.3	0.2, 0.3
I	0.2	0.2
D	0.2	0.2
M	0.6	0.6
MI	0.2	0.2
MD	0.2	0.2
MM	0.6	0.6
관련 실험	(그림 8)	(그림 9), (그림 10)

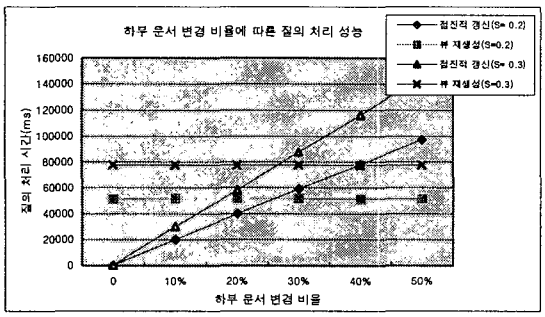
4.3 실험 결과

본 논문에서는 실체뷰를 재생성하는 방법과 점진적으로 갱신하는 기법의 성능이 교차하는 지점의 조건을 찾고자 하부 문서 변경 비율에 따른 실험과 하부 문서 저장 개수에 따른 실험을 하였다. 또한, 전자에 비해 후자가 얼마나 더 효율적인지를 알아보고자 하였다.

4.3.1 하부 문서 변경 비율에 따른 질의 처리 성능 비교

첫째 실험에서는 하부 문서 개수가 일정할 때 하부 문서 변경 비율에 따라 실체뷰를 재생성하는 방법에 비해 점진적으로 갱신하는 기법의 성능이 더 효율적일 때의 조건을 찾고자 하였다. 하부 문서 개수는 INSERT 연산과 DELETE 연산이 같은 비율로 발생한다고 가정하여 항상 일정한 개수를 유지한다. 따라서 (그림 8)과 같이 실체뷰를 재생성하는 방법에서는 실체뷰를 재생성하기 위해 검색해야 할 전체 하부 문서 개수가 일정하게 유지됨으로써 질의 처리 시간이 일정하게 유지되었다. 반면에 점진적 갱신 기법에서는 INSERT 연산과 DELETE 연산이 같은

비율로 발생하고, MODIFY 연산에서 MODIFY-INSERT 와 MODIFY-DELETE 연관성 유형이 같은 비율로 발생한다고 가정했기 때문에 뷰를 제공하기 위해 검색해야 할 실체뷰 크기가 일정하지만, 하부 문서 변경 비율이 높아질수록 변경로그 레코드 개수가 증가하므로 질의 처리 시간이 증가하였다. 또한 (그림 8)과 같이 S(View Selectivity)가 0.2 에서 0.3 으로 증가하면 실체뷰의 연관성 비율이 증가하므로 두 가지 방법 모두 질의 처리 시간이 증가하였다. S=0.2, 0.3 의 두 경우 모두에서 하부문서 변경 비율이 약 27% 이하일 때는 뷰 재생성 기법에 비하여 점진적 갱신 기법이 더 효율적인 것으로 나타났다.



(그림 8) 하부문서 변경 비율에 따른 실험 결과

4.3.2 하부 문서 저장 개수에 따른 질의 처리 성능 비교

둘째 실험에서는 하부 문서 변경 비율이 일정할 때 하부 문서 저장 개수에 따른 질의 처리 성능을 측정하였다. (그림 9)는 하부 문서 변경 비율이 20%로 일정하고 하부 문서 저장 개수가 <표 4> U 의 변화와 같이 증가할 때, S(View Selectivity)가 0.2 와 0.3 의 비율일 때의 실체뷰를 재생성하는 방법과 점진적으로 갱신하는 기법의 성능을 측정한 결과를 보인 것이다. (그림 9)에서는 전자와 후자의 질의 처리 시간이 모두 증가함을 보이고 있다. 이는 실체뷰를 재생성하는 방법의 질의 처리 시간이 하부 문서 저장 개수가 증가함에 따라 실체뷰를 재생성하기 위해 검색해야 할 전체 하부 문서 개수가 증가했기 때문이다. 그리고 점진적 갱신 기법에서의 질의 처리 시간은 전체 하부 문서 개수가 증가하면 하부 문서 개수에 따른 하부 문서 변경 비율에 의해 변경로그 레코드 개수가 증가하므로 질의 처리 시간은 증가한다. 또한, (그림 9)의 결과는 <표 4>의 파라미터 값 설정하에서 점진적 갱신 기법이 항상 더 효율적임을 의미한다. (그림 10)에서는 (그림 9)의 실험 결과를 토대로, 점진적 갱신 기법이 실체뷰를 재생성하는 방법보다 성능 향상이 얼마나 더 이루어졌는지를 보여 주고 있다. 즉, 실체뷰를 재생성하는 방법에 대한 점진적 갱신 기법의 성능 향상 정도를 계산하여 나타낸 그래프이다. 아래 식에서 VR 은 실체뷰를 재생성하는 방법의 질의 처리 시간이고, MV 는 점진적 갱신 기법의 질의 처리 시간이다. 성능 향상 △는 다음과 같이 정의된다.

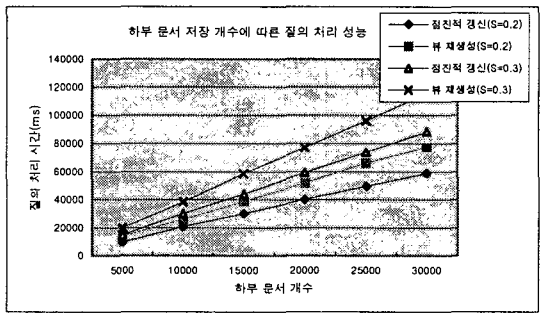
$$\Delta = \frac{(VR - MV)}{VR} \times 100\%$$

(그림 10)에서 성능 향상은 S(View Selectivity)에 따라 20%에서 30%까지에 이름을 알 수 있다.

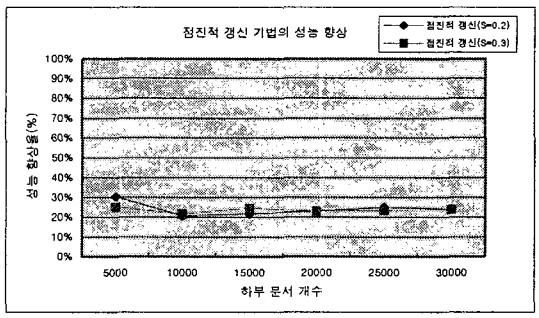
5. 결론 및 향후 연구 과제

본 논문에서는 실체뷰 지원 XML 저장 시스템을 구현하고, 구현한 시스템을 대상으로 XML 실체뷰의 일관성을 유지하기 위해 실체뷰를 재생성하는 방법과 점진적으로 갱신하는 기법의 성능을 실험을 통해 측정하고 이를 비교하였다. 점진적 갱신 기법은 실체뷰의 일관성을 유지하기 위하여 오버헤드가 있음에도 불구하고 전자에 비해 더 나은 성능을 갖는 조건을 가지고 있었다. 이런 결과는 자주 제기되는 XML 질의에 대해 그 결과를 재생성하는 방법은 방대한 양의 XML 문서를 검색해야 하지만 점진적 갱신 기법은 자주 제기되는 XML 질의에 대해 실체뷰를 유지함으로써 적은 양의 XML 문서를 검색하는 데서 오는 상대적인 검색 시간의 이득을 가지고 있기 때문이다. 실험 결과를 요약하면, (그림 8)의 실험에서는 하부 문서 변경 비율이 약 27% 이하일 경우 점진적 갱신 기법이 더 효율적임을 보였고, (그림 9)와 (그림 10)의 실험에서는 점진적 갱신 기법이 재생성 방법보다 항상 성능이 우수했으며, 성능 향상은 약 20%에서 30%에 이르렀다.

본 논문에서는 제한된 가정 하에 실험을 수행하였다. 뷰 정의를 내용 기반 조건에 국한하고 있으며, 그 조건은 DTD 를 구성하는 엘리먼트 중 한번만 나오는 엘리먼트 하나에 대해서 명시된다고 가정하였다. 따라서 구조적 질의를 통한 뷰 정의 지원, 조건 엘리먼트에 대한 제약 조건이 없는 경우를 지원하는 시스템의 구현과 그에 대한 성능 평가가 필요하다. 또한 평균 엘리먼트 개수가 7000 개 이상 일 수 있는 세익스피어 희곡과 같은 큰 XML 문서에 대한 실험 등도 필요하다.



(그림 9) 하부 문서 저장 개수에 따른 실험 결과



(그림 10) 점진적 갱신 기법의 성능 향상

참고문헌

[1] T. Bray et al., "Extensible Markup Language(XML) 1.0," http://www.w3.org/TR/1988/REC-xml-19980210, 1998.

[2] D. Hunter et al, "Beginning XML", Wrox, 2000.

[3] DATEC, http://www.datec.co.kr.

[4] Oracle, http://www.oracle.com.

[5] SQL Server 2000 http://www.microsoft.com/korea/sql/.

[6] S. Abiteboul, "On Views and XML," Proc. ACM Symp. on Principles of Database System, 1999, pp.1-9.

[7] 임재국 외, "점진적 갱신에 기반을 둔 XML 형성뷰 관리 프레임워크," 정보처리학회논문지, 제 8-D 권, 4 호, 2001. 8, pp.327-338.