질의 재구성 기반의 XQuery 질의 정제

최성일[0], 박종현, 강지훈
충남대학교 컴퓨터공학과
{sichoi[0], jonghyunpark, jhkang }@cnu.ac.kr

XQuery query Refinement Based on Query Rewriting

Seong-Il Choi[0], Jong-Hyun Park, Ji-Hoon Kang
Dept. of Computer Engineering, ChungNam National University

요 약

XML은 웹 상에서 데이터의 표현과 교환을 위한 표준이다. XQuery는 XML 질의를 위한 W3C 표준으로서 XML 문서를 효율적으로 검색하기 위해서 W3C에서 제안한 표준질의어이다. XQuery가 등장하면서, 이를 빠르게 처리하기 위한 연구가 많이 진행 중이며, 이 연구 중 한 분야는 XQuery 질의를 정제하는 것이다. 사용자에 따라 다양하게 작성되는 XQuery 질의들은 정제되어 있지 않을 수 있다. 질의의 불필요한 연산이나 표현을 제거하여 간결하게 만드는 것은 질의를 효율적으로 처리하게 하여 성능을 향상시키는데 도움을 준다. 이에 대한 이전의 연구들은 XML 데이터의 저장 구조나 시스템에 의존적인 질의 정제방법을 사용하므로 이들 방법을 일반적인 XQuery 질의 정제로 볼 수는 없다. 그러나 우리의 정제방법은 XQuery 질의를 기반으로 하여 일반적인 상황에서도 질의의 정제가 가능하므로 XQuery를 입력으로 하는 다른 시스템에서 우리의 방법으로 입력 질의를 정제하여 효율적으로 질의를 처리할 수 있다. 본 논문에서는 XQuery 질의를 효율적으로 처리하기 위하여 두가지 정제방법을 제안한다. 첫째는 불필요한 연산이나 표현을 제거하는 방법이고, 둘째는 질의의 순서를 재배치하는 방법이다. 이 방법들을 통하여 질의를 보다 빠르고 효율적으로 처리하도록 한다. 끝으로, 우리는 성능평가를 통하여 우리의 정제방법의 효율성을 입증한다.

1. 서 론

XML은 웹 상에서 데이터를 기술하고 교환하기 위한 표준이다. XML 문서를 효율적으로 검색하기 위해서 W3C에서 XQuery라는 표준 질의어를 제안하였다[1]. XQuery가 등장하면서, XQuery를 빠르고 효율적으로 처리하기 위한 연구가 현재까지 계속 진행되고 있으며, 그 중 한 분야가 질의 재구성 기반으로 XQuery 질의를 정제하는 것이다.

XQuery의 문법은 SQL의 문법과 유사하지만 복잡하고 다양하게 나타낼 수 있다. 동일한 결과를 갖는 XQuery 질의들은 사용자에 따라 다르게 작성된다. 이들 질의 중에는 정제되어 간결한 질의가 있는 반면, 불필요한 연산이나 표현이 사용되어 복잡하고 간결하지 못한 질의도 있다. 위의 두 질의를 비교해 보면 효율성과 수행시간으로 볼 때 전자가 훨씬 유리할 것이다. 그러므로 다양하게 생성된 XQuery 질의를 최적으로 수행할 수 있도록 질의를 정제하는 것은 중요하다. 이는 다른 연구에서도 이미 수행되고 있지만, XML 데이터의 저장 구조나 시스템에 의존적인 질의 정제방법을 사용한다. 그러므로 이들 방법을 일반적인 XQuery 질의 정제라고 보기는 힘들다. 그러나 우리의 정제방법은 XQuery 질의 자체를 기반으로 하여 일반적인 상황에서도 질의의 정제가 가능하다. 그러므로 XQuery가 입력이 되는 모든 시스템에서 우리의 방법으로 입력 질의를 정제할 수 있으며, 효율적으로 질의를 처리할 수 있다.

본 논문에서는 XQuery 질의 자체만을 이용하여 XQuery 질의를 정제하는 2가지 방법을 제안한다. 첫 번째 방법은 XQuery 질의를 수행하는데 불필요하고 중복된 표현을 제거하는 것이고, 두 번째 방법은 XQuery 질의를 보다 효율적으로 처리하기 위해서 질의의 순서를 재배치하는 것이다. 위의 2가지 방법으로 XQuery를 정제하여 질의를 수행하는데 보다 빠르고 효율적으로 처리할 수 있게 한다. 또한 우리는 우리의 방법들의 효율성을 증명하기 위해 성능평가를 수행하였다.

본 논문의 구성은 다음과 같다. 2절에서는 XQuery 정제에 대한 기존 연구들을 기술하고 있으며, 3절에서는 질의 정제 시스템에 대하여 설명한다. 4절에서는 우리의 XQuery 질의의 정제 방법에 관하여 기술하며, 5절에서는 성능평가를 보인다. 마지막으로 6절에서는 결론을 기술한다.

2. 관련연구

XQuery가 XML 데이터들의 통합과 검색을 위해 많이 사용되면서 이를 효율적으로 처리하기 위한 연구가 현재 진행 중에 있다. 이러한 연구들 중 하나가 질의 재구성 기반의 XQuery 질의 정제이다. 이 정제를 위한 방법들 중 하나는 XAT라는 XQuery 질의 모델을 정의하여 XQuery 질의 계획을 생성한 후 이를 정제하는 방법이다[2][3][4]. 그러나 이들 방법은 질의 정제를 위하여 통합 View, XML Schema, DTD와 같은 추가적인 정보를

이용한다. 또한 XAT라는 algebra 표현으로 변환하기 때문에 다른 시스템에서 그 방법을 적용하기는 쉽지 않을 것이다.

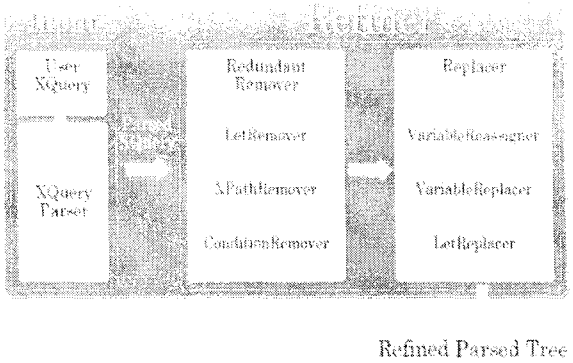
XQuery 질의 정제를 위한 방법들 중 다른 하나는 SQL 질의를 위한 정제 방법을 XQuery 질의의 정제를 위하여 사용하는 방법이다[3][4][5][6]. XQuery의 문법은 SQL과 유사하기 때문에 SQL의 정제방법 중 XQuery에 적용할 수 있는 부분이 많을 것이다. 그러나 이들 방법은 아직까지 초기 연구 단계이므로 SQL 질의의 정제 방법 중 어떤 방법이 XQuery 질의의 정제를 위해서 사용 가능하다고 기술하고 있을 뿐, 구체적인 언급은 없는 상황이다.

본 논문에서는 XQuery의 대상이 되는 문서가 항상 View 또는 XML Schema, DTD와 같은 정보를 기반으로 하고 있지 않다는 점을 고려하여 사용자의 입력 XQuery 질의 자체만을 이용하여 최적화 하는 방법을 제안하고 있다. 본 논문에서 제안하고 있는 방법은 XQuery 질의 이외에 다른 정보는 최적화를 위해서 사용하지 않으므로 어떠한 형태의 XQuery 질의에도 사용이 가능하다. 또한 XQuery가 입력이 되는 다른 시스템에서 우리의 정제방법을 적용한 XQuery 질의를 사용하여 성능을 높일 수 있다.

3. 질의 정제 시스템의 구조

우리의 XQuery 질의 정제 시스템은 XQuery 질의가 입력되면 정제된 XQuery 질의를 출력한다. [그림 1]은 XQuery 질의 정제 시스템의 구성을 나타낸다. XQuery 질의 정제 시스템은 Redundnat Remover와 Replacer의 2가지 모듈로 구성되어 있다. 사용자가 입력한 XQuery 질의는 XQuery Parser를 통해 구문 분석 Tree로 변환된다. 이 구문 분석 Tree가 우리의 시스템의 입력이 되고, 먼저 Redundant Remover로 전달된다. 세부 모듈이 모두 수행되면 Replacer로 구문 분석 트리를 전달하고 세부 모듈을 수행한 후 정제된 XQuery 구문 분석 Tree를 출력한다.

또한 모듈들은 각각 독립적으로 구성되어 있으므로 부분적으로도 수행이 가능하다. RedundantRemover는 LetRemover, XPathRemover, ConditionRemover의 3가지 모듈로 구성되어 있다. LetRemover는 사용하지 않는 LET절을 제거하고, XPathRemover는 중복된 XPath표현을 제거하고, ConditionRemover는 WHERE절의 조건 중에서 중복된 의미의 연산을 제거한다. RedundantRemover는 LET절, XPath표현, WHERE절의 조건에서 불필요한 정보를 제거하기 위한 질의 정제를 수행한다. Replacer는 VariableReassigner와 VariableReplacer, LETReplacer의 3가지 모듈로 구성되어 있다. VariableReassigner는 WHERE절의 변수를 Constant값으로 대체할 수 있는 경우 변수를 그 값으로 치환을 하고, VariableReplacer는 셀렉션에 해당하는 조건을 먼저 수행할 수 있도록 자리를 재배치하고, LETReplacer는 LET절을 최적으로 수행할 수 있는 위치로 이동시킨다. Replacer는 WHERE절과 LET절에서 질의 순서를 재배치하여 질의를 정제하는 모듈이다. 이 모듈들의 구체적인 방법들은 4절에서 자세히 설명한다.



[그림 1] 질의 정제 시스템의 구조

4. XQuery 질의 정제를 위한 방법

4.1 불필요한 정보를 제거하는 방법

XQuery 질의를 정제하기 위한 첫 번째 방법은 XQuery 질의 내에 있는 불필요하고 중복된 연산이나 표현을 제거하는 것이다. 이를 위해 세부적으로 3가지 단계를 거친다.

1. 사용하지 않는 LET절의 제거
2. 중복된 XPath 표현의 제거
3. WHERE절의 조건 중에서 의미적으로 중복된 부분을 제거

```
a. <Results>{
b.   for $dblp in doc("dblp.xml")/dblp
c.   for $book in $dblp//book
d.   for $b_Author in $book/author
e.   for $article in $dblp//article
f.   for $a_Author in $article/author
g.   let $a_Volume := $article/volume
h.   let $a_Title := $article/title
i.   for $inproceedings in $dblp//inproceedings
j.   let $i_Year := $inproceedings/year
k.   let $b_Year := $book/year
l.   where $b_Author = $a_Author
m.         and $a_Volume > 50 and $a_Volume >= 100
n.         and $i_Year = $b_Year and $book/year = 1999
o.   return
          <result>{$b_Author, $article/volume}</result>
p. }</Results>
```

[표 1]XQuery 질의 정제를 위한 원본질의

[표 1]은 우리의 정제방법을 설명하기 위한 예제 질의이다. 첫 번째는 사용하지 않는 LET절을 제거하는 방법이다. LET절에서 선언된 변수가 하위의 어느 곳에서도 사용되지 않으면 그 LET절을 제거한다. 사용하지 않는 LET절이 제거되면 상위의 FOR절의 반복 횟수만큼 LET 변수의 할당이 줄어들기 때문에 그 만큼의 성능이 향상된다. [표 1]의 원본질의의 h 라인을 보면 LET절이 선언되어 있는데 LET 변수인 $a_Title은 하위의 FOR, LET, WHERE, RETURN 어느 곳에도 사용되고 있지 않다. 따라서 이 LET 선언을 제거해도 질의 결과는 변하지 않으며, 수행시간은 줄어들어 성능이 향상된다.

두 번째는 중복된 XPath표현을 제거하는 방법이다. LET절에서 사용된 XPath표현이 WHERE절이나 RETURN절에서 사용되면 그 부분을 해당 변수로 대체하여 중복된 XPath표현을 제거한다. 이는 이미 XPath표현이 계산되어 있으므로 그 결과를 이용하여 중복된 연산을 피한다. 원본질의의 라인 n인 WHERE절을 보면 $book/year가 있고, 라인 o인 RETURN절을 보면 $article/volume이 있다. 이 XPath표현들을 각각 $b_Year, $a_Volume으로 대체하여 질의를 정제한다.

세 번째는 WHERE절의 조건 중에서 동치 또는 포함관계의 연산 중 의미적으로 중복된 부분을 제거하는 방법이다. 이 방법은 값 비교 연산자(Value Comparisons)와 일반 비교 연산자(General Comparisons)를 대상으로 하고, 조건들이 and 연산자로 구성되었을 때 그 포함관계를 비교하여 교집합에 해당하는 조건을 남기고 나머지는 제거한다. 이는 의미적으로 중복된 조건을 제거하므로 불필요한 연산을 줄이는 것이다. 원본질의의 라인 m을 보면 두 조건 중 '$a_Volume > 50'을 제거해도 WHERE절이 의미적으로 변하지 않는다.

4.2 질의의 순서를 재배치하는 방법

XQuery 질의를 정제하기 위한 두 번째 방법은 질의의 순서를 재배치하여 보다 효율적으로 질의를 처리하게 한다. 이를 위해 세부적으로 3단계를 거친다.

1. WHERE절의 변수 값 치환
2. 셀렉션 조건 이동
3. LET절의 위치 이동

첫 번째는 WHERE절에 사용된 변수가 '='와 'eq' 연산자에 의해 값이 명시된 경우에 이를 해당 값으로 대체하는 방법이다. 이 경우는 그 변수에 해당되는 모든 값을 대상으로 하지 않고 명시된 값만을 대상으로 검색해도 질의의 결과는 동일하다. [표 1]의 질의의 라인 n인 WHERE절을 보면 '$book/year'는 1999와 같은 것만 검색하면 되므로 '$i_Year = $b_Year'를 '$i_Year = 1999'로 치환한다. 이와 같은 경우 조인 연산이 셀렉션 연산으로 바뀌어 $i_Year에 해당되는 모든 값을 검색하지 않아도 되므로 그 만큼의 검색시간을 줄일 수 있다.

두 번째는 WHERE절의 조건 중 셀렉션에 해당되는 조건을 먼저 수행하도록 위치를 이동하는 방법이다. 셀렉션 연산은 다른 연산보다 빠르게 수행되며, 중간결과를 줄일 수 있기 때문에 다음에 수행되는 연산의 비용을 감소시킨다. [표 1]의 라인 n을 보면 '$b_Year = 1999'와 위의 방법으로 변환된 '$i_Year = 1999'가 있을 것이다. 이들은 셀렉션 연산이므로 앞쪽으로 재배치하여 먼저 수행하도록 한다.

세 번째는 LET절을 최적으로 처리할 수 있는 위치로 이동하는 방법이다. [표 1]의 라인 k의 LET절은 $book의 child인 year를 할당하는데 $book이 선언된 곳은 라인 c에 위치하므로, 그 사이의 FOR절의 반복 횟수만큼 불필요하게 LET절을 할당하게 된다. 그러므로 이 LET절을 $book이 선언된 FOR절의 아래로 이동하여 불필요한 LET절의 할당을 줄이게 된다.

[표 2]는 [표 1]의 원본질의에 Redundant Remover와 Replacer를 모두 적용하여 질의가 정제된 최종 모습이다.

```
a.<Results>{
b.   for $dblp in doc("dblp.xml")/dblp
c.   for $book in $dblp//book
d.   let $b_Year := $book/year
e.   for $b_Author in $book/author
f.   for $article in $dblp//article
g.   for $a_Author in $article/author
h.   let $a_Volume := $article/volume
i.   for $inproceedings in $dblp//inproceedings
j.   let $i_Year := $inproceedings/year
k.   where $b_Year = 1999 and $i_Year = 1999
l.         and $a_Volume >= 100
m.        and $b_Author = $a_Author
n.   return
          <result>{$b_Author, $a_Volume}</result>
o. }</Results>
```

[표 2] [표 1]의 원본질의를 Redundant Remover와 Replacer를 수행하여 정제된 질의

5. 성능평가

5.1 실험환경

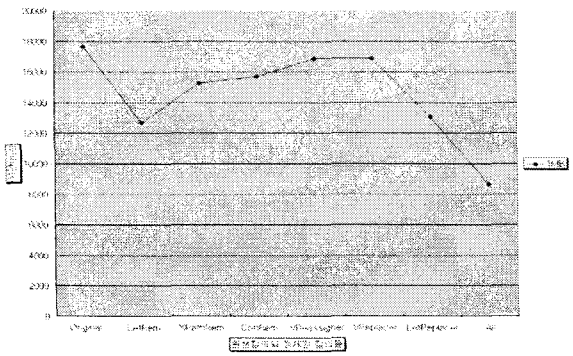
우리는 우리의 정제방법의 성능을 측정하기 위하여 운영체제는 윈도우 2003을 사용하였고 CPU는 Pentium 4 3.0GHz, 2GB의 메인 메모리를 사용하였다. XQuery 엔진은 Oracle XQuery 엔진을 사용한다.

성능평가를 위해 DBLP의 XML문서를 이용하였고, 원본질의와 우리의 정제방법을 각각 적용한 질의, 모두 적용한 질의의 수행 시간을 측정하였다. 또한 데이터의 크기에 따른 수행시간의 변화도 측정하였다.

5.2 실험결과

성능평가에 사용한 질의는 [표 1]의 원본 질의이고,

그 질의를 각각의 방법으로 정제하여 XQuery 엔진이 처리하는 시간을 측정하였다.



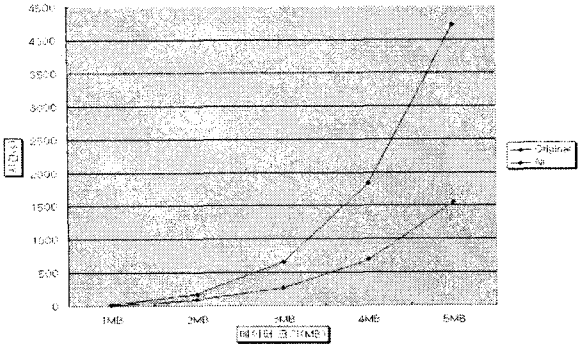
[그림 2] [표 1]의 원본질의와 정제된 질의들의 수행시간

[그림 2]는 1MB의 DBLP XML 문서를 사용하여 원본 질의와 정제된 질의들의 수행시간을 측정한 결과이다. 왼쪽부터 순서대로 Original(원본질의), LetRemover, XPathRemover, ConditionRemover, VariableReassigner, VariableReplacer, LetReplacer, 그리고 모든 정제방법을 적용한 수행시간이다. 각각의 방법들은 원본 질의보다 수행시간이 더 빨라졌으며, 가장 좋은 성능을 보이는 것은 LetRemover였다. 사용하지 않는 LET절을 제거함으로 상위의 FOR절의 반복만큼 불필요한 LET절의 할당을 제거하여 수행시간이 많이 줄어들었다. 다음으로 나은 성능을 보이는 것은 LetReplacer였다. Let절을 최적으로 수행할 수 있는 곳으로 자리를 재배치하는 이 방법 역시 불필요한 LET절의 할당을 줄여 성능을 높일 수 있었다. 위의 결과로 볼 때 LET절과 연관된 정제방법이 성능향상에 많은 도움이 됨을 알 수 있다. XPathRemover와 ConditionRemover가 그 다음으로 성능이 좋았고, 나머지 정제방법들도 원본질의의 수행시간보다 나은 성능을 보이는 것을 확인할 수 있다. 원본질의와 모든 정제방법을 적용한 질의의 수행시간을 비교하면 약 50%의 성능이 향상됨을 볼 수 있다.

[그림 3]은 DBLP XML 데이터의 크기를 1MB에서 5MB로 1MB씩 증가시키면서 [표 1]의 원본질의와 모든 정제방법이 적용된 [표 2]의 질의의 수행시간을 측정한 것이다. 데이터 크기가 증가함에 따라 비례적으로 성능이 점차 향상됨을 볼 수 있다.

6. 결론

본 논문에서는 XQuery 질의를 효율적으로 처리하기 위한 정제방법들을 제안하였다. 우리의 정제방법들은 기존에 연구 되었던 방법들과는 달리 다른 질의 시스템과 연동하여 사용할 수 있으며, 질의 자체를 기반으로 정제를 하므로 일반적인 상황에서도 적용이 가능하다. 우리의 XQuery 질의 정제방법은 다른 XQuery 질의 처리기



[그림 3] 데이터의 크기에 따른 [표 1]의 원본질의와 [표 2]의 정제된 질의의 수행시간

에서도 사용이 가능하므로 특성에 맞게 활용할 수 있으며, 확장하여 사용할 수도 있을 것이다.

7. 참고문헌

[1] W3C, XQuery 1.0: An XML Query Language, W3C Candidate Recommendation 3 November 2005,(http://www.w3.org/TR/2005/CR-xquery-20051103/).

[2] X. Zhang & E. A. Rundensteiner ""XAT: XML Algebra for the Rainbow System."" Technical Report WPI-CS-TR-02-24, Worcester Polytechnic Institute, July 2002.

[3] F. Frasincar, G. J. Houben & C. Pau "XAL: An Algebra For XML Query Optimization Proc. Australasian Database Conference 2002, Melbourne, Australia, February 2002.

[4] X. Zhang, B. Pielech & E. A. Rundensteiner ""XML Algebra Optimization" Technical Report, WPI-CS-TR-02-25, Worcester Polytechnic Institute, October 2002.

[5] M. Grinev, & P. Pleshachkov, "Rewriting-Based Optimization for XQuery Transformational Queries" IDEAS 2005, Montreal, Canada, July 2005.

[6] M. Grinev & S. Kuznetsov. "Towards an Exhaustive Set of Rewriting Rules for XQuery Optimization: BizQuery Experience", Proc. ADBIS Conference, Springer, Bratislava, Slovakia, September 2002.