적응력 있는 XML 스트림 데이터 질의처리 기법*

김영현° 강현철
중앙대학교 컴퓨터공학부
yhkim°@dblab.cse.cau.ac.kr hckang@cau.ac.kr

Adaptive Processing of Queries over XML Streaming Data

Young Hyun Kim° Hyunchul Kang
School of Computer Science and Engineering, Chung-Ang University

요 약

스트림 데이터를 생성하는 응용의 증가로 스트림 데이터 처리에 대한 연구가 활발히 진행되고 있다. 이러한 응용의 예로 센서네트워크, 모니터링, Selective dissemination of information(SDI)등이 있다. 특히 SDI와 같은 웹 환경의 응용은 XML을 기반으로 스트림 데이터 처리에 대한 연구를 진행하고 있다. XML은 웹상의 데이터 교환의 표준으로 웹 응용의 증가로 인해 스트림 데이터 처리 분야에서도 XML을 사용하는 스트림 데이터 처리 시스템에 대한 연구가 많이 진행되고 있는 실정이다. 하지만 XML을 기반으로 하는 기존의 시스템들은 정적인 질의계획을 사용하여 스트림 데이터를 처리하기 때문에 동적으로 변하는 스트림 데이터에 적응력있게 대처할 수 없다. 이와 달리 관계 데이터 모델을 사용하는 스트림 데이터 처리 시스템은 동적인 질의계획과 질의 처리 연산자의 라우팅(스케쥴링) 기법을 사용하여 적응력있는 질의처리를 지원한다. 본 논문에서는 관계 데이터 모델을 사용하는 시스템의 적응력있는 질의처리 모델을 XML을 기반으로 하는 시스템에 적용하는 기법들에 대하여 설명한다. 그리고 XML을 기반으로 하는 기존의 대표적인 시스템인 YFilter[7]와 본 논문의 제안하는 시스템과의 질의처리 성능을 비교 평가한다.

1. 서 론

스트림 데이터를 생성하는 응용의 증가로 스트림 데이터 처리에 대한 연구가 활발히 진행 되고 있다. 이러한 응용의 예로는 센서네트워크[1],모니터링[2], Selective Dissemination of Information(SDI)[3]등이 있다. 특히 SDI와 같은 웹 환경의 응용은 XML을 기반으로 하는 스트림 데이터 처리에 대한 연구를 진행하고 있다. 그리고 웹 상의 데이터 교환의 표준으로 XML이 부각된 이래 XML 스트림 데이터에 대한 처리 시스템의 연구가 활발히 진행되고 있다.

스트림 데이터는 연속적이며 빠르게 생성되는 데이터이다. 기존의 시스템들은 이런 스트림 데이터에 대해 실시간 처리를 보장하기 위해 다양한 질의 처리 기법들을 제안하였다. 제안된 기법들은 스트림 데이터의 종류에 따라 두 가지 부류의 시스템으로 나뉠 수 있다. 첫 번째로는 관계 데이터 모델을 스트림 데이터로 사용하는 시스템[4-6]이다. 이런 시스템들은 동적인 질의 계획과 연산자 라우팅(스케쥴링) 기법을 사용하여 스트림 데이터를 처리한다. 동적인 질의 처리 기법을 통해 가변적으로 변하는 스트림 데이터를 적응력 있게 처리한다. 두번째로는 XML을 기반으로 하는 XML 스트림 데이터 처리 시스템[7-9]이다. 이런 시스템들은 우선적으로 사용자로부터 입력받은 질의를 오토마타로 변형한다. 그리고 입력으로 들어오는 XML 스트림 데이터를 파싱하면서 오토마타의 상태와 비교하여 질의를 처리한다. XML의 구조를 파악하기 위해 파싱이라는 작업을 하기 때문에 XML을 스트림 데이터를 기반으로 하는 시스템들은 정적인 질의 계획을 생성한다. 하지만 이런 정적인 질의 계획은 가변적으로 변하는 스트림 데이터를 적응력 있게 처리할 수 없다는 문제가 있다. 따라서 본 논문에서는 XML을 스트림 데이터로 사용하는 시스템에서의 동적인 질의 처리 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2절에서는 스트림 데이터 처리 시스템에 관한 관련 연구를 기술 하며, 3절에서는 본 논문에서 제안하는 XSDMS 시스템에 대해 설명한다. 4절에서는 본 논문이 제안하는 시스템과 관련된 성능 평가를 수행하고, 끝으로 5절에서는 결론을 맺고 향후 연구 내용을 기술한다.

2. 관련 연구

본 논문은 기존의 관계 데이터 모델을 기반으로 하는 스트림 데이터 처리 시스템들의 동적인 질의 처리 모델을 XML 스트림 데이터 처리 시스템에 적용시키는 데 있어 해결해야 할 문제와 제안하는 기법을 설명한다.

기존의 관계 데이터 모델을 기반으로 하는 시스템들은 동적인 질의 처리 모델을 가진다.[4-6] 그 중 본 연구의 기반이 되는 시스템은 TelegraphCQ[4]이다. TelegraphCQ는 SteMs를 사용하여 질의와 데이터를 모두 스트림 데이터의 형태로 처리한다. 그리고 Eddy System을 사용하여 연속적으로 들어오는 스트림 데이터를 라우팅하여 SteMs에 저장한다. Eddy System은 티켓 기반의 라우팅 정책을 사용하여 스트림 데이터를 라우팅하는데, 이 기법은 Eddy System 내부에 존재하는 각각의 질의 처리 연산자에 티켓을 할당하는 방법이다. 티켓은 스트림 데이터에 해당하며 질의 처리 연산자가 스트림 데이터를 하나 처리하게 되면 티켓을 하나 할당받게 된다. 그리고 할당된 티켓은 연산자에 계속적으로 누적된다. 각각의 연산자가 서로 다른 티켓의 값을 가지게 되고 많은 티켓을 가지는 연산자 순으로 스트림 데이터를 처리한다. 많은 티켓을 가지는 연산자는 다른 연산자에 비해 많은 데이터를 걸러내는 기회를 가지게 되며 이런 동적인 연산자 배치를 통해 다음의 연산자가 처리하는 데이터의 수가 적어지는 이점을 얻을 수 있는 것이다. 하지만 XML 스트림 데이터 모델을 기반으로 하는 시스템들[7-9]은 XML 문서의 구조적 정보를 알아내기 위해 XML 문서를 파싱한다. 이러한 파싱작업은 XML 문서에 대해 순차적인 접근을 강요한다. 그래서 XML 스트림 데이터 처리시스템들은 정적인 질의 계획을 사용하여 XML 문서를 파싱하면서 질의 처리를 수행한다. 이런 정적인 질의 처리기법은 위에서 언급한 동적인 질의 처리 기법이 가지는 장점을 가질 수 없으며, 다양한 워크로드를 가지는 스트림 데이터에 적응력있게 대처할 수 없다.

본 논문에서 제안하는 XSDMS 시스템과 기존의 시스템이 가지는 큰 차이점은 첫째, 기존의 시스템들이 주로 관계 데이터 모델을 사용하는데 반해 본 시스템은 웹 데이터 교환의 표준으로 자리 잡은 XML을 데이터 모델로 사용하여 다양한 응용에서 발생 할 수 있는 XML 스트림 데이터에 대한 처리를 가능하게 한다는 것이다. 둘째, XML을 기반으로 하는 기존의 시스템이 정적인 질의 처리 모델을 사용하는 것과는 달리 본 논문이 제안하는 시스템은 동적인 질의 처리 모델을 사용하여 적응력 있는 질의 처리 기법을 제시한다는 것이다.

3. XSDMS (XML Streaming Data Management System) 모델

3.1 개요

XSDMS는 XML 형태의 스트림 데이터를 실시간으로 처리하는 시스템이다. XML 형태의 스트림 데이터를 처리하기 위해서 XML 문서의 파싱작업이 필요하지만, 본 시스템에서는 XML 스트림 뷰[10]를 사용하여 XML 스트림 데이터를 파싱 작업없이 질의를 처리하는 모델을 제시한다. 전체적인 시스템의 모델은 <그림 1>과 같다. XML 문서는 Stream Mediator를 경유하여 XML 스트림 뷰가 더해진 다음에 StreamSlimer모듈에 전달된다. 전달된 XML 스트림 데이터는 StreamSlimer의 시스템 타임 스탬프를 부여받고 Buffer Manager에 저장된다. 만약 StreamSlimer에 등록된 질의가 존재한다면 질의·처리기는 질의 계획 생성기에 의해 만들어진 질의 계획을 사용하여 Buffer Manager에 저장되어 있는 XML 스트림 데이터를 처리하여 질의결과에

*본 논문은 한국과학재단 특정기초연구사업(R01-2003-000-10395-0) 지원으로 수행되었음.

해당하는 데이터를 Result Manager에 전달해 주는 것으로 질의 처리 과정을 마치며, Buffer Manager에 존재하는 다음 XML 스트림 데이터를 처리를 시작한다.

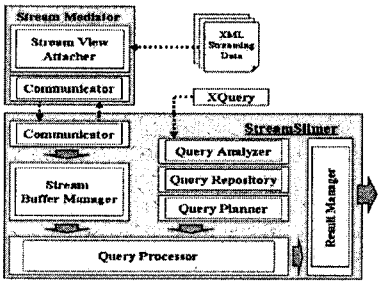

그림 1. XSDMS 모델

3.2 XML 스트림 뷰

스트림 뷰는 스트림에 대한 뷰로서 즉, 스트리밍 되는 데이터에 대한 질의의 결과 스트림을 의미한다. XML 스트림 뷰[10]는 XML 문서에 대한 XML 질의의 결과 엘리먼트 정보를 문서의 헤더 형태로 함께 저장하는 것을 말한다. XML 스트림 뷰에 추출하고자 하는 엘리먼트의 정보가 존재할 경우 XML 문서의 파싱 작업 없이 빠르게 접근할 수 있다는 장점이 있다. 본 논문에서는 XML 스트림 데이터에 XML 스트림 뷰를 더해 질의 처리 시의 파싱 오버헤드를 줄였으며 스트림 뷰의 사용으로 동적인 질의 처리가 가능하도록 하였다. XML 스트림 뷰를 XML 문서에 추가하는 작업 과정은 <그림 2>를 참조하면 된다.

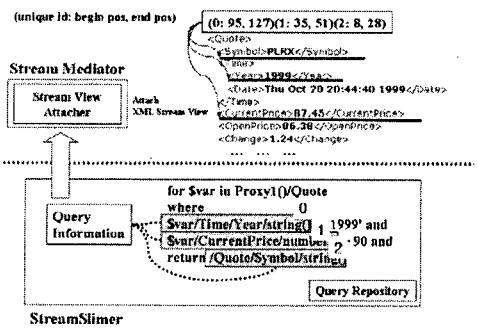

그림2. XML 스트림 뷰를 XML 문서에 추가하는 과정

3.3 Query Processing

XML 문서를 스트림 데이터로 사용하는 시스템들[7-9]은 XML 문서를 파싱하여 질의 처리를 수행한다. XML 문서를 파싱하면서 질의 처리를 하게 되면 질의 처리 계획이 정적일 수밖에 없기 때문에 질의 처리가 순차적으로 진행된다. 예를 들어 다음과 같은 XPath 질의가 있다고 가정해 보자.

/Quote[Symbol[text()="RSAS"]][Time/Year[text()="1999"]]

- **/Quote/Symbol/text() = "RSAS" has 100% selectivity.**
- **/Quote/Time/Year/text() = "1999" has 20% selectivity.**

이와 같은 XPath 질의를 처리함에 있어 XML 문서를 파싱하여 질의를 처리하는 시스템들의 경우 만약 XML 문서 상에 Symbol이 Time보다 먼저 나타난다면 20%의 Selectivity를 가지는 /Quote/Time/Year 조건을 먼저 처리하지 못하고 항상 100%의 Selectivity를 가지는 /Quote/Symbol 조건을 먼저 처리하기 때문에 질의 처리 과정이 효율적이지 않다. 하지만 본 논문에서 제시하는 시스템은 XML 문서를 스트림 데이터로 사용하지만, 관계 데이터 모델을 기반으로 하는 스트림 데이터 처리 시스템[4-6]과 같이 동적인 질의 계획을 세움으로써 위와 같은 상황에서 낮은 Selectivity를 가지는 조건이 먼저 처리될 수 있는 질의 처리 기법을 제시한다. 본 시스템에 1개 이상의 질의가 등록되면 <그림 3>과 같은 질의 계획이 만들어 지게 된다.

질의 Q0가 등록이되면 StreamSlimer 내부의 질의 분석기를 통해 질의가 분석된 후 질의 저장소에 저장된다. 질의 저장소에 저장된 질의 Q0를 사용하여 질의 계획 생성기는 단계 1과 같은 질의 계획을 만든

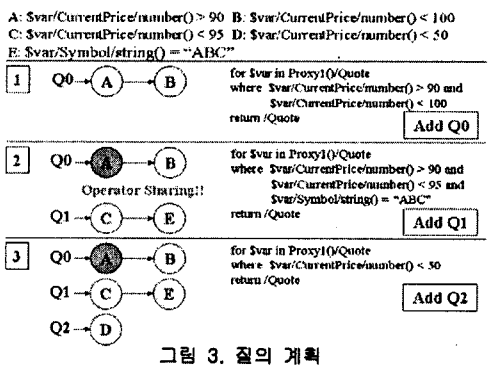

그림 3. 질의 계획

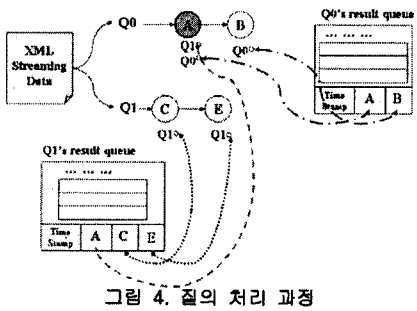

그림 4. 질의 처리 과정

다. 다음으로 Q1이 등록이되면 전과 같은 과정이 반복되면서 질의 Q1에 공유할 수 있는 조건(연산자)이 존재하는 지를 검사하며 공유할 수 있는 조건 A를 Q1이 가지고 있기 때문에 단계 2와 같이 Q0에 Q1의 연산자 A를 합친 질의 계획을 만들어 낸다. 이와 같은 연산자를 공유하는 질의 계획은 [4-6]과 같은 관계 데이터 모델 기반의 시스템들의 질의 처리 방식과 유사하다. 그 다음으로 질의 Q2가 등록이되면 단계3과 같은 질의 계획이 세워진다. <그림 3>과 같은 질의 계획을 사용하여 질의 처리기는 질의 처리를 하게 된다. (질의 처리기법에 대해서는 다음의 <그림 4>를 보면서 설명)

본 시스템에서는 [4]의 Eddy와 유사하게 질의 조건을 만족했는 지를 비트 마스크를 사용하여 처리한다. 하지만, Eddy는 데이터에 비트 마스크를 추가함으로써 데이터의 크기를 증가시키지만 본 시스템에서는 비트 마스크를 <그림 4>와 같이 결과 큐에 둠으로써 데이터의 증가를 피하였다. A~E의 연산자들은 자신과 관련된 질의의 결과 큐에 존재하는 비트 마스크를 <그림 4>와 같이 참조하게 된다. <그림 4>와 같은 상태에서 XML 스트림 데이터가 들어오게 되면 Q0와 Q1은 싱글 쓰레드 내에서 질의 처리를 하게 된다. (처리될 연산자를 고르는 순서는 다음 절의 연산자 라우팅에서 설명) 각각의 연산자들은 XML 스트림 데이터를 처리하고 난 후 처리 결과를 참조하고 있는 결과 큐의 비트 마스크에 현재 처리하고 있는 데이터의 타임 스탬프와 함께 적게 된다. 만약 결과 큐에 이미 타임 스탬프 값이 설정되어 있다면 지금 할당하려고 하는 타임 스탬프 값과 비교하여 다른 타임 스탬프를 할당하려고 하면 결과 큐의 비트 마스크에 쓰여진 값은 예전의 데이터를 처리할 때 쓰인 값이므로 비트 마스크를 갱신 한 후에 결과를 적는다. 질의를 처리하는 도중 결과 큐에 존재하는 비트 마스크가 전부 참으로 설정되면 결과를 큐에 삽입하고 그렇지 않으면 삽입하지 않는다. 그리고, 만약 <그림 4>에 존재하는 Q0의 A 연산자의 처리 결과가 거짓이라면 다음 B 연산자와 질의 Q1은 수행하지 않아도 된다. 그럼 다음 절에서는 질의 각각의 연산자에 적용하는 라우팅 기법을 살펴보겠다.

3.4 연산자 라우팅

입력된 질의에 포함되는 조건(연산자)들은 시스템 상에 존재하는 데이터에 따라 데이터를 필터링하는 비율이 틀려지게 된다. 그래서 각각의 조건들을 처리함에 있어 낮은 Selectivity를 가지는 조건들을 먼저 처리하는 질의 계획을 세우게 되면 더 효과적인 질의 처리를 할 수 있게 되는 것이다.

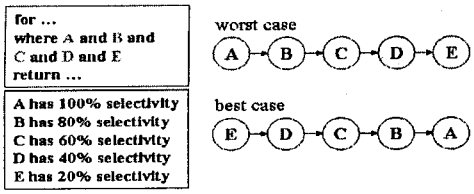

그림 5. 연산자 라우팅

<그림 5>를 보면 A~E 연산자들은 서로 다른 Selectivity를 가진다. 각각의 Selectivity에 따라 가장 좋은 질의 계획과 가장 나쁜 질의 계획을 볼 수 있다. 가장 좋은 질의 계획의 경우 E 연산자에 의해 20%의 확률로 D연산자의 질의 처리 과정이 수행이 된다. 하지만, 가장 나쁜 경우의 질의 계획은 A 연산자에 의해 100%의 확률로 B 연산자의 질의 처리 과정을 수행하게 된다. 예에서 볼 수 있듯이 Selectivity가 낮은 연산자의 질의 처리 수행을 먼저 함으로 인해서 좀 더 효율적인 질의 처리를 할 수 있다. 본 논문이 제안하는 시스템에서는 이런 연산자 라우팅을 지원하기 위해 질의 처리를 할 때 다음과 같은 식을 사용하여 각각의 연산자의 Selectivity를 측정한다.

$$\text{오퍼레이터 X의 지금 까지 의 Selectivity} = \frac{\text{지금까지의 오퍼레이터 X의 처리 결과가 참인 횟수}}{\text{오퍼레이터 X에 의해 지금 까지 처리된 데이터의 양}}$$

<그림 3>의 단계 3에 있는 질의 계획을 사용하여 질의 처리를 할 때 Q0의 A 연산자의 Selectivity가 낮아 가장 먼저 처리되었다고 가정하자. 그리고 만약 A 연산자의 처리 결과가 거짓이었다고 하면 Q0의 질의 처리 과정을 정지하는 것과 더불어 Q1의 질의 처리를 시작하지 않아도 되는 효과를 볼 수 있다.

4. 성능 평가

3절의 내용을 만족하는 XML 스트림 데이터 처리 시스템을 Java를 사용하여 Windows 2000 Server에서 구현하였다. 성능 평가 실험에 사용한 문서는 Niagara[8] 시스템의 성능 평가에서 사용한 "Quotes.xml" 문서[12]를 사용하였다. 이 문서는 2 MB 가량의 크기를 가진다.

실험은 두 가지 성능에 대한 측정을 한다. 첫 번째는 YFilter[7]와 본 시스템의 질의 처리 성능에 대한 평가이다. YFilter는 오토마타 기법을 사용하여 입력되는 XML 문서에 대한 정적인 질의처리 기법을 사용하는 대표적인 시스템이다. 그래서 본 논문이 제안하는 시스템이 가지는 동적인 질의처리 기법의 비교 대상으로 선정하였다. 두 번째는 본 시스템에서 사용하는 연산자 라우팅에 대한 효율성 실험이다. 수행 결과는 다음과 같다. (그림 6, 7 참조)

<그림 6>은 본 시스템과 YFilter의 질의 처리 시간에 대한 실험결과로 실험에 사용한 질의유형은 다음과 같다.

- YFilter: /Quotes/Quote[Symbol[text()="PLRX"]][Time/Year[text()="1999"]]
- XSDMS:
```
for $var in proxy1()/Quotes/Quote
    where $var/Symbol/string() = "PLRX" and
    $var/Time/Year/string() = "1999"
    return $var
```

실험 결과는 <그림 6>을 보면 알 수 있듯이 XML 스트림 뷰를 사용하는 본 시스템의 성능이 YFilter보다 뛰어났다. 하지만, 확장성 측면에서 본다면 YFilter가 뛰어남을 알 수 있었다. (그림 6의 XSDMS w/o Operator Marker 참조) 본 논문이 제안하는 시스템에 확장성을 더하기 위해 연산자 마커라는 기법을 사용하였다. 연산자 마커를 사용한 후의 실험결과는 <그림 6>과 같다. (그림 6의 XSDMS w/ Operator Marker 참조) 연산자 마커는 이미 처리한 연산자의 XPath질의를 기억하여 다음의 불필요한 연산을 막는 역할을 한다. (자세한 설명은 지면 관계상 생략) 그림 다음으로 본 시스템의 비율 기반의 연산자 라우팅 기법에 대한 평가인 <그림 7>을 보자. 이 실험은 <그림 5>와 같은 조건을 가진 질의를 사용하여 각각의 XML 스트림 데이터가 평균적으로 어느 정도의 연산자를 거치게 되는 지를 측정한 것이다. 값이 작을수록 좋은 성능을 가진다. 실험 결과를 보면 본 시스템의 비율 기반의 라우팅 기법이 최적의 기법과 별로 차이가 나지 않는 것을 볼 수 있다.

5. 결론 및 향후 연구

본 논문에서는 XML 스트림 데이터 기반의 동적인 스트림 데이터 처리 시스템을 제안하고, 구현 및 다른 시스템과의 성능 평가를 수행하였다. 성능 평과 결과, 본 시스템의 질의 처리 성능이 YFilter에 비해 우수함을 알 수 있었다.

향후 연구 과제는 스트림 간의 조인과 [11]에서 연구했던 것을 기반으로 과거 데이터와 스트림 간의 조인 기법을 연구하는 것이다.

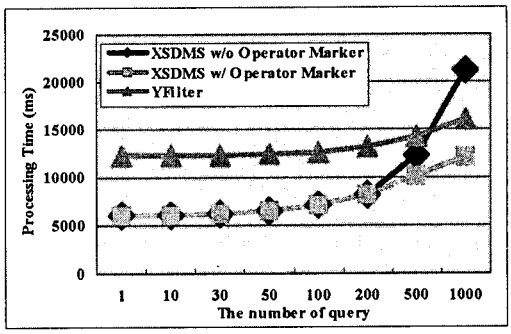

그림 6. 질의 수행 시간 측정그림

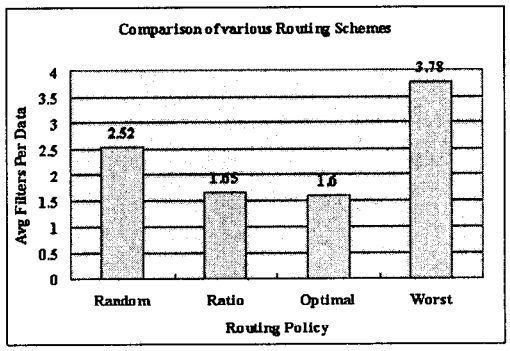

그림 7. 서로 다른 연산자 라우팅 기법의 비교

참고 문헌

[1] Samuel R. Madden et al., "The Sensor Spectrum: Technology, Trends, and Requirements", ACM SIGMOD Record, v.32 n.4, Dec. 2003.

[2] D. Carney et al., "Monitoring Streams: A New Class of Data Management Applications", Proc. of VLDB, 2002.

[3] Y Diao, Shariq Rizvi, and Michael J. Franklin, "Towards an Internet-Scale XML Dissemination Service", Proc. of VLDB Conference, 2004.

[4] Samuel Madden et al., "Continuously adaptive continuous queries over streams", Proc. of ACM SIGMOD, 2002.

[5] The STREAM Group. "STREAM: The Stanford Stream Data Manager", IEEE Data Engineering Bulletin, Vol. 26 No. 1, 2003

[6] D. Abadi et al.,"Aurora: A New Model and Architecture for Data Stream Management", In VLDB Journal, 2003.

[7] Y. Diao, M. Franklin, "High-Performance XML Filtering: An Overview of YFilter", In Proc. of ICDE, 2003.

[8] Jianjun Chen et al., "NiagaraCQ: A Scalable Continuous Query System for Internet Databases", Proc. of ACM SIGMOD, 2000.

[9] Makoto Onizuka, "Ligt-weight XPath processing of XML Stream with Deterministic Automata", Proc. of Int'l Conf. on Information and knowledge management, 2003.

[10] A. Gupta, A. Halevy, and D. Suciu, "View selection for XML stream processing", In WebDB, 2002.

[11] 한승철, 강현철, "XML 스트림 데이터에 대한 연속 질의 처리 시스템," 한국정보처리학회 논문지 D, 제 11-D권 제7호, 2004.

[12] http://www.cs.wisc.edu/niagara/data, 2000