Journal of Digital Contents Society Vol. 8 No. 4 Dec. 2007(pp. 541-546)

스트림 데이터를 위한 데이터 구동형 질의처리 기법

민미경*

요 약

많은 양의 연속적인 스트림 데이터를 대상으로 하는 연속적인 질의처리의 경우는 전통적 방식의 요구구동형 질의처리 방식이 적합하지 않다. 본 논문에서는 자료구동형 방식을 도입하여 질의를 처리함으로써 스트림 데이터에 알맞은 질의처리 기법을 제안하고 질의계획의 구조와 질의실행 방식을 설명하였다. 제안된 질의처리 기법은 다중질의 처리가 가능하며, 질의 간에 공유가 가능하게 한다. 또한 부분질의의 실행결과가 저장됨으로써 실행시간을 단축할 수 있다. 본 질의처리 모델에 XML 데이터와 XQuery 질의를 적용하였다.

A Data-Driven Query Processing Method for Stream Data

Mee-Kyung Min*

Abstract

Traditional query processing method is not efficient for continuous queries with large continuous stream data. This paper proposes a data-driven query processing method for stream data. The structure of query plan and query execution method are presented. With the proposed method, multiple query processing and sharing among queries can be achieved. Also query execution time can be reduced by storing partial results of query execution. This paper showed an example of query processing with XML data and XQuery query.

Keywords : stream, query plan, query execution, XML

1. 서론

전통적인 데이터베이스 관리 시스템은 한정된 저장 데이터 집합에 대하여 한 번에 하나의 질의를 수행하는데 적합하게 설계되어 있다. 안정된 저장소를 통해 데이터를 관리하고 데이터가 존재하는 동안 이에 대한 질의를 수행한다. 그러나 네트워크 모니터링, 회계분석, 제조, 센서 네트워크 등과 같은 현대 응용에서는 많은 연속적인 스트림 데이터에 대한 연속적 질의를 처리해야 한다. 정적인 데이터를 여러 번 반복하여 처리할 수 있는 기존의 분야와는 달리 연속적으로 끊임없이 입력이 주어진다. 사용자들은 질의를 필요에 따라 입력하여 결과를 얻는 것이 아니라 원하는 질문을 먼저 등록한다. 질의처리기는 이 질문을 스트림 데이터에 따라 처리한다.

스트림 데이터에 관한 연구는 활발히 이루어지고 있다. 스탠포드 대학에서는 연속질의 언어인 CQL[3]을 개발하고 이를 처리하는 데이터 스트림 관리 시스템 개발을 위한 STREAM 프로젝트를 수행하였다[4]. Telegraph-CQ는 UC Berkely 의 프로젝트로서 연속적 질의간 공유가 가능하고 실행시간이 긴 장기 질의를 처리가능하도록 하였다[10].

최근 스트리밍 XML 데이터 처리에 관한 연구도 활발히 진행되어 [2][5][7][9][12][16]과 같은 연구가 발표되었다.

또한 전통적인 생성시스템의 빠른 패턴매칭을 위한 Rete 알고리즘[6]이 있다. 이 알고리즘은 [11] 등에서 성능향상 이슈가 제기되었고 최근까지도 연구가 계속되고 있다[1][8][9][15].

본 논문에서는 대량의 스트림 데이터에 대해 연속적 질의처리를 하는 응용에 적합한 질의처리기법을 제안한다. 이 기법은 Rete 알고리즘을

※ 제일저자(First Author) : 민미경
접수일자:2007년08월20일, 심사완료:2007년09월10일
* 서경대학교 컴퓨터학과 교수
mkmin@skuniv.ac.kr

기반으로 하며 많은 객체와 많은 패턴간의 빠른 매칭이 기본목적이다. 본 질의처리 기법은 연속적인 스트림과 전통적인 저장 데이터 집합을 모두 처리한다. 특징은 다음과 같다.

· 대량의 데이터가 연속적으로 처리되며 여러 개의 질의가 한꺼번에 처리된다. 사용자가 미리 등록한 질의들은 물리적인 질의계획으로 변환된다.
· 질의는 물리적인 질의계획으로 바뀌는데, 이 계획은 노드, 저장소, 대기열로 구성된다.
· 대량의 데이터에 대해서 데이터구동형으로 질의가 처리된다.
· 여러 질의간에 공유가 이루어져 질의계획이 생성된다. 많은 연속 질의를 처리하기 위해서는 빠르고 효율적인 계획이 필요하다. 본 모델에서는 여러 질의간에 공유를 가능하게 함으로써 효율성을 높인다.

본 질의 처리 기법은 XML 문서들을 여과하는데 사용될 수 있다. 본 논문에서는 XML 문서를 스트림 데이터로 보고 질의를 처리하는 예를 보인다.

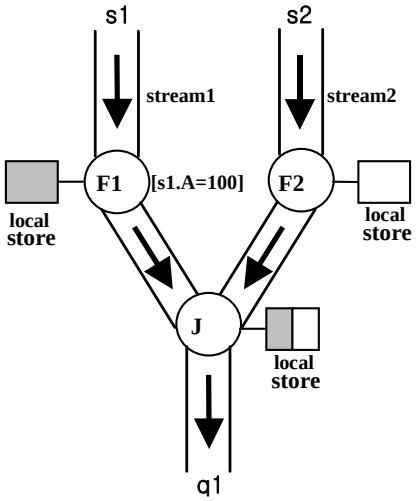
2. 질의처리 모델

2.1 질의계획

질의처리를 위해서는 질의계획이 수립되는데 질의계획은 노드, 저장소, 대기열로 구성된다. 각 노드는 대기열에서 입력되는 스트림이 질의조건을 만족하는지 검사하는 역할을 한다. 각 노드에는 지역저장소가 있다. 조건검사를 만족하는 데이터는 일단 이 저장소에 저장되어 추후 공유와 재사용을 위해 활용된다. 노드에서 걸러진 데이터들은 계속적인 처리를 위해 다음 노드로 전달된다. 각 노드들에는 처리를 기다리는 스트림으로 구성된 대기열이 연결되어 있다.

다음 (그림 1)은 본 논문에서 제시하는 질의처리 모델의 질의계획을 보여준다. 이 질의계획은 다음과 같은 질의를 위한 것이다.

q1 : Select * **From** S1, S2 **Where** S1.A = S2.A **And** S1.A = 100



(그림 1) 질의계획

2개의 필터노드 F1, F2와 1개의 조인노드가 생성된다. F1은 S1의 데이터집합 s1에 대한 필터링을 수행한다. J는 두 스트림을 조인한다. 지역저장소는 각 노드마다 존재한다. 노드의 출력 스트림은 다음노드의 입력이 된다. 스트림 데이터는 차례로 처리되므로 대기열에 존재한다.

2.2 질의실행

질의계획이 수립되면 데이터에 대한 질의가 실행된다. (그림 1)의 질의가 실행되는 과정은 다음과 같다.

· 각 대기열 s1과 s2에는 S1의 스트림 데이터와 S2의 스트림 데이터가 노드 F1과 F2의 입력으로 들어온다.
· 노드 F1은 s1에 대한 필터이고 노드 F2는 s2에 대한 필터이다. 각각은 단일조건의 만족여부를 검사한다. 이 예에서는 s1에 대한 조건만 존재하므로 F2는 dummy 노드이고 F1에서는 S1.A = 100인가를 검사한다.
· 필터노드를 통과한 스트림은 일단 지역저장소에 저장된다. 각 필터노드마다 지역저장소가 있다. 지역저장소에 저장된 데이터들은 복사형태가 아닌 참조형태로 필터노드를 통과하여 다음 노드로 전달된다.
· 필터노드 F1과 F2는 각각 데이터 스트림을

조인노드인 J로 전달한다. 조인노드에서는 대기열로부터 들어오는 데이터들을 받아서 S1.A = S2.A 인지 검사한다.

· 조인조건을 만족하는 데이터 쌍들은 노드 J의 지역저장소에 저장된 후 다음 플로우로 전달된다.

· 더 이상의 노드가 없으므로 질의 q1의 실행이 완료되고 결과가 출력된다.

2.3 성능향상을 위한 노드 공유

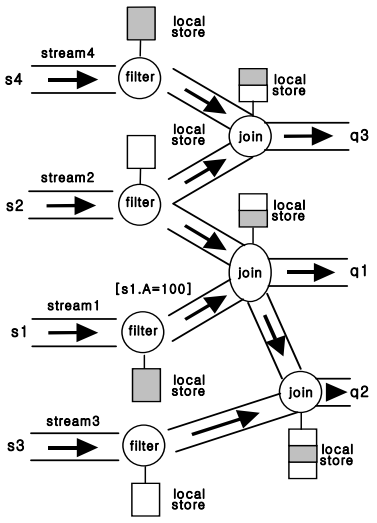
본 기법은 여러 개의 질의를 한꺼번에 처리할 수 있다. 사용자가 등록한 여러 개의 질의는 질의계획으로 변환된다. 여러 개의 질의계획이 수립되면 이들 간에 서로 중복되는 내용이 있게 된다. 이러한 중복을 피하기 위해서 질의계획에서 하나 또는 그 이상의 노드들이 공유된다.

다음 (그림 2)는 3개의 질의 q1, q2, q3 에 대한 질의계획이다. 조인노드와 필터노드가 공유되고 있다.

q1: **Select** * **From** S1, S2 **Where** S1.A = S2.A **And** S1.A = 100

q2: **Select** * **From** S1, S2, S3 **Where** S1.A = S2.A **And** S1.A = 100 **And** S2.B = S3.B

q3: **Select** * **From** S2, S4 **Where** S2.C = S4.C



(그림 2) 질의계획에서의 노드공유

2.4 데이터구동형 부분질의실행

전통적인 데이터베이스관리시스템은 한 번에 하나의 질의를 처리하므로 사용자가 질의를 입력할 경우 이에 대한 계획을 세우고 실행하는 요구구동형 질의실행방식을 택하고 있다. 그러나 스트림 데이터의 경우는 많은 양의 데이터와 질의가 발생된다. 따라서 질의가 입력될 때마다 이를 실행하는 것은 시간이 많이 필요하다.

본 논문에서 제시하는 질의처리 모델은 요구구동형이 아니라 데이터구동형으로 작동한다. 데이터가 입력될 때마다 미리 만들어진 질의계획에 따라 질의가 실행된다. 실행된 결과는 전체질의의 결과가 될 수도 있고 부분 질의의 결과가 될 수도 있다. 부분 질의의 처리 결과는 각 노드마다 연결되어 있는 지역저장소에 저장된다. 이와 같은 모델의 장점은 다음과 같다.

첫째 실행된 부분 결과들은 항상 다른 질의와 공유될 수 있다. 노드의 지역 저장소에 저장되어 있는 데이터의 결과는 현재까지 조건이 만족된 데이터에 대한 부분질의 결과이다. 그러므로 이 노드를 공유하고 있는 다른 질의와 자연스럽게 부분질의의 실행결과가 공유된다.

둘째, 실행된 부분 결과들이 각 노드의 지역 저장소에 저장되어 있기 때문에 추후에는 이 데이터들을 다시 검사할 필요가 없다. 다만 새로 입력되는 데이터나 변화되는 데이터에 대한 조건검사만 필요하다. 따라서 질의실행 시간이 절약된다.

이러한 데이터구동형 질의처리 모델이 특히 유용하게 사용되는 경우는 다음과 같다.

· 한 번 사용된 질의가 여러 번 재사용되거나 비슷한 질의가 계속 사용된다.

· 사용될 질의가 미리 등록되어 있어서 질의계획을 미리 세워놓을 수 있다.

· 질의계획과 부분 실행 결과가 저장되어 있다.

3. 질의계획의 구성요소

질의계획은 (그림 1)에서와 같이 데이터 스트림, 대기열, 노드, 지역 저장소로 구성된다.

3.1 데이터스트림과 대기열

대량의 연속적인 데이터스트림은 데이터베이스나 문서, 기타 형태의 데이터이며 대기열을 통해서 들어온다. 이 대기열은 각 노드의 입출력이 되며 노드들을 연결해준다.

데이터스트림에 들어오는 데이터들은 다음과 같은 형태이다.

[정의] 필터노드 대기열 = {<tag, time, data>} 이다. 태그는 + 또는 - 이며, + 는 삽입을 - 는 삭제를 나타낸다. time 은 각 데이터가 도착한 시간을 말하며 대기열의 데이터는 time 이 작은 것부터 소비된다. data 는 입력데이터이다.

[정의] two-input 조인노드 대기열 = {<tag, time, data1, data2>} 이다. data1과 data2는 조인된 데이터의 쌍을 나타낸다.

마찬가지로 n-input 조인노드를 정의할 수 있다.

3.2 노드

노드는 연산자에 해당하며, 후에 연산자의 평가를 위해 데이터베이스의 상태를 저장할 수 있다. 노드는 가능한 다른 질의와 공유한다. 노드의 역할중 하나는 부질의에서 걸러지는 부분 결과를 실체화하는 것이다. 또한 노드에는 질의 실행의 결과가 유지된다.

본 논문에서 제시하는 기본적인 노드는 필터노드와 조인노드이다. 이들의 정의는 다음과 같다.

[정의] 필터노드는 조건식이 논리연산자로 결합된 논리식이다.

이 때 조건식 = (f, a, b) 이다.

f 는 검사함수이며, a, b는 피연산자이다.

[정의] 조인노드 = (조인테스트, 대기열1, 대기열2) 이다.

이때 조인테스트= (f, a, b) 의 조건식이 논리연산자로 결합된 논리식이다.

3.3 지역저장소

지역저장소의 형태는 입력스트림 수에 따라 달라진다. one-input node인 필터노드의 저장소는 1개 데이터의 집합이다. two-input 노드인 조인노드의 저장소는 조인된 2개 데이터 쌍의 집합이다. 3-input 조인노드의 저장소는 3쌍 데이터의 집합이다.

[정의] 필터노드의 저장소={<$data_1$>}, two-input join node의 저장소={<$data_1$, $data_2$>}, n-input join node의 저장소={<$data_1$, $data_2$, . . . <$data_n$>} 이다.

4. XML 데이터 처리

논문에서 제안하는 질의처리기 모델의 수행과정을 살펴보기 위해 다음 예를 적용한다.

```
< employee department= "d_z002">
      <employeeID> = 101>
      <firstName>미경</firstName>
      <lastName>민</lastName>
      <title>President</title>
      <dateStarted>2000-11-12</dateStarted>
      <salary>8K</salary>
</employee>
<employee department = "d_z001">
      <employeeID> = 102>
      <firstName>길동</firstName>
      <lastName>홍</lastName>
      <title>Chief Technical Officer</title>
      <dateStarted>2002-9-16</dateStarted>
      <salary>4K</salary>
</employee>
<department deptID = "d_z001">
      <deptName>Research</deptName>
      <location>Seoul</location>
</department>
-- 이하생략 --
```

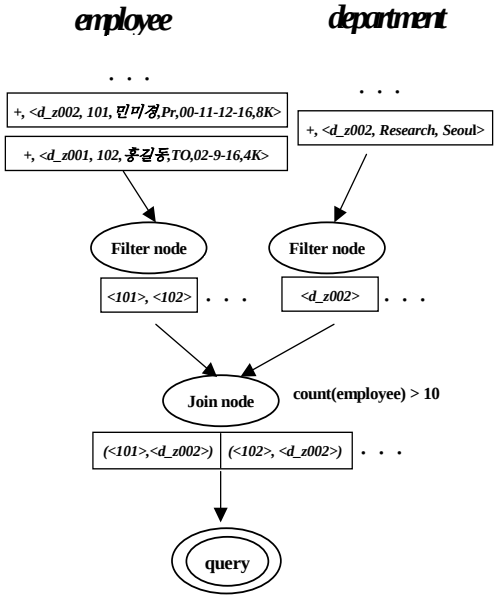
```
for $d in fn:doc("depts.xml")/depts/deptno
let $e :=
fn:doc("emps.xml")/employees/employee[department
        = $d]
where fn:count($e) >=10
return
    <big-dept>
      {
      $d,
      <headcount> {fn:count($e)}<headcount>,
      <avgsal>{fn:avg($e/salary)}</avgsal>
      }
    </big-dept>
```

(그림 3) XML 데이터와 질의의 예

(그림 3)은 XML[12] 데이터와 XQuery[13]의 질의의 예이다. employee의 수가 10명 이상인 department 사원의 평균 월급을 구하는 질의로서, XQuery 1.0으로 작성되었다. for 와 let 절의 결과는 튜플 스트림이고 각 튜플 스트림에는 $d

와 $e의 쌍이 들어있다. where 절은 이 튜플 스트림 중에서 조건을 만족하는 스트림만 걸러낸다.

이 예에서 질의는 다음과 같은 질의계획으로 변환된다. 입력데이터 XML 문서는 질의계획에 입력되어 질의가 실행되고 결과로서 <big-dept> 라는 엘리먼트를 생성한다.



(그림 4) XML 데이터처리

5. 결 론

최근 많은 연속적인 스트림 데이터에 대한 연속적 질의를 처리할 필요성이 증가하였다. 본 논문에서는 이러한 목적의 질의처리 기법을 제안하고 여기에 XML 데이터의 예를 적용하였다. 제안한 모델은 데이터구동형으로 작동하며 스트림 데이터를 처리하기 위한 목적으로 구성되었다. 본 논문에서 제안한 질의처리 기법의 특징은 다음과 같다.

첫째, 데이터구동형으로 동작한다. 질의가 들어오면 데이터를 검사하는 방식의 요구구동형 질의처리방식은 많은 양의 데이터가 연속적으로 입력되는 스트림 데이터의 경우는 적절하지 않다. 본 질의처리 모델에서는 새로운 데이터가 입력되면 질의가 실행된다. 이때 부분질의가 실행된 부분결과는 질의계획의 중간부에 저장되어 있어 추후 같은 부분을 실행할 때 이를 재사용한다.

따라서, 한번 검사된 부분에 대해서는 다시 검사하지 않으므로 질의실행 시간이 감소한다.

둘째, 다중 질의가 동시에 처리되며 질의 간에 질의계획의 일부가 공유된다. 따라서 많은 질의를 빠른 시간 내에 처리할 수 있으며 질의간에 질의계획을 공유함으로써 질의실행 시간을 단축시킨다.

향후 다양한 스트림 데이터에 대한 수행시간 측정 등 질의처리 기법에 대한 평가가 연구과제로 남아있다.

참 고 문 헌

[1] D. J. Abadi et al., "Aurora: a New Model and Architecture for data stream management", The VLDB Journal, Vol 12(2), pp. 120-139, 2003.

[2] M. Altinel and M.J. Franklin, "Efficient Filtering of XML Documents for Selective Dissemination of Information", Proc. of 26th International Conf. on VLDB, pp. 53-64, 2000.

[3] A. Arasu, S. Babu, and J. Widom, The CQL Continuous Query Language: Semantic Foundations and Query Execution. Technical Report, Stanford University, 2003.

[4] A. Arasu et al., STREAM: The Stanford Data Stream Management System. Technical Report, Stanford University, 2004.

[5] S. Bose and L. Fegaras, "Data Stream Management for Historical XML Data", Proc. of the ACM SIGMOD International Conf. on Management of Data, pp. 239-250, 2004.

[6] C. L. Forgy, "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", Artificial Inteligence, Vol. 19(1), pp. 17-37, 1982.

[7] A. K. Gupta and D. Suciu, "Stream Processing of XPath Queries with Predicates," Proc. of the 18th International Conf. on Data Engineering, pp. 341-342, 2002.

[8] C. Jin and J. Carbonell, Argus: Rete+DBMS= Efficient Continuous Profile Matching on Large-Volume Data Streams, Technical Report, Carnegie Mellon Univ

ersity, 2004.
[9] T. Kim and M. Min, ",A Pattern Matching Method for Object-Oriented Query Processing, International Conf. on Information and Knowledge Engineering, pp. 643-649, 2002.
[10] S. Krishnamurthy et al., TelegraphCQ: An Architectural Status Report, IEEE Data Engineering Bulletin, Vol. 26(1), 2003.
[11] D. P. Miranker, TREAT: A New and Efficient Match Algorithm for AI Production Systems, Morgan Kaufmann Publishers, 1990.
[12] F. Peng and S. Chawathe, "XPath Queries on Stream Data," Proc. of the 2003 ACM SIGMOD International Conf. on Management of Data, pp. 431-442, 2003.
[13] W3C, Extensible Markup Language (XML) 1.0, W3C, 2006, www.w3.org/TR/EC- xml.
[14] W3C, XQuery 1.0: An XML Query Language, W3C, 2007, www.w3.org/TR/xquery.
[15] I. Wright, "The Execution Kernel of RC++: RETE*, A Faster Rete with TREAT as a Special Case", International Journal of Intelligent Games and Simulation, volume 2(1), pp. 36-48, 2003.
[16] 민준기, 박명제, 정진완, "스트리밍 XML 데이터를 위한 효율적인 다중질의 처리 기법", 정보과학회논문지: 데이터베이스, 제 34권 제 3호, pp. 270-281, 2007.

민 미 경

1987년 : 서울대학교 계산통계학과 졸업(학사)
1989년 : 서울대학교 대학원 전산과학과 졸업(석사)
1993년 : 서울대학교 대학원 전산과학과 졸업(박사)
2005년~2007년 : 미국 USC 대학교 Department of Computer Science 교환교수
1994년~현 재 : 서경대학교 컴퓨터과학과 교수
관심분야 : 데이터베이스, 디지털 컨텐츠, 컴퓨터 교육