제36회 한국정보처리학회 추계학술발표대회 논문집 제18권 제2호 (2011.11)

트리거 기반 XML 빈발 구조 추출

황정희*
*남서울대학교 컴퓨터학과
e-mail:jhhwang@nsu.ac.kr

Frequent Structure Extraction of XML based on Trigger

Jeong Hee Hwang*
*Dept of Computer Science, Namseoul University

요 약

유비쿼터스 컴퓨팅 환경에서 무한의 연속적으로 전송되는 데이터에 대한 처리가 요구되고 있다. 본 논문에서는 연속적이고 빠르게 발생하는 스트림 데이터로부터 유용한 정보를 발견하기 위한 기반 연구로써 트리거를 이용한 슬라이딩 윈도우 기반의 XML 빈발 구조 추출 방법을 제안한다.

1. 서론

스트림 데이터 마이닝은 데이터 스트림에 대한 실시간 분석 능력을 지원한다. 다시 말해, 어느 시점에서나 해당 시점까지의 모든 트랜잭션을 포함하는 현재 데이터 스트림에 대한 마이닝 결과를 얻을 수 있도록 지원한다. 그러나 스트림 데이터 마이닝은 최대 한번 탐색하고, 메모리 사용량은 무한히 증가되지 않으며 최신의 데이터에 대한 빠른 분석을 제공하는 것을 목적으로 한다[1,2,3,4]. 그러므로 XML 데이터로부터 빈발 구조를 탐색하는 기존의 연구방법[5]을 기반으로 스트리밍 XML 데이터의 특성을 고려하는 빈발 구조의 탐색방법에 대한 연구가 필요하다.

본 논문에서는 시간에 따라 변화되는 최신 XML 스트림 데이터로부터 빈발 구조 정보를 효율적으로 탐색할 수 있는 방법을 제안한다. 이를 위해 기존 스트림 마이닝 기법을 확장하여 XML 특성을 고려하는 트리거에 의한 슬라이딩 윈도우 기반의 마이닝 기법을 이용한다. 스트리밍 XML 데이터에 대한 빈발 구조 탐색은 스트림 데이터에 대한 효율적인 질의 처리 및 색인 구성에 효율적으로 이용될 수 있다.

2. 트리거 규칙

본 논문에서는 DBMS 내부적으로 트리거를 사용하여 연속적으로 삽입되는 스트림에 대한 윈도우를 유지한다. 이 과정에서 최초로 스트림 데이터가 삽입되는 워킹 테이블(Working_table)과 워킹 테이블 중에서 데이터 마이닝을 하기 위해 관심 있는 현재 시점 기준의 데이터를 유지하는 윈도우 테이블(Window_table)이 존재한다. 그리고 마이닝 테이블(Mining_table)은 윈도우 테이블에서 빈발구조를 발견하기 위한 대상이 되는 데이터를 유지한다.

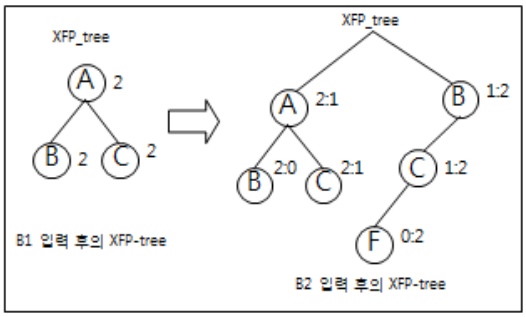
트리거 규칙의 예로써, 입력되는 스트림 데이터에 대해 일정한 튜플 수를 기준으로 탐색대상이 되는 데이터를 윈도우 테이블에 자동으로 삽입하는 트리거 규칙 1(Trigger1)을 정의한다. 워킹 테이블에 데이터 삽입이 발생하면 트리거 규칙1(Trigger1)에 의해 윈도우 테이블로 일정 수의 트랜잭션 묶음인 배치(batch)가 삽입된다. 워킹 테이블에 삽입된 트랜잭션에 대한 배치 시퀀스 번호를 순차적으로 부여하여 윈도우 테이블에 삽입한다. 워킹 테이블은 삽입되는 전체 데이터를 유지하는 백업 데이터의 역할도 하고, 필요시 오래된 데이터들은 삭제가능하다.

```
<Trigger 1>
CREATE OR REPLACE TRIGGER trigger_wnd_insert
AFTER INSERT ON wrk_table
FOR EACH ROW
DECLARE
w_SEQ2 NUMBER;
w_fSEQ NUMBER;
w_TRN NUMBER: = 5;
BEGIN
 SELECT (MAX(tn_bseq) +1) INTO w_SEQ2
 FROM  wnd_table
 IF (w_SEQ2 IS NULL) THEN
   w_SEQ2: = 1;
 SELECT MIN(tn_seq) INTO w_fSEQ
 FROM wrk_table
 GROUP BY tn_flg
 HAVING tn_flg = 0;
 // batch당 주어진 트랜잭션 수 만큼 이동
 FOR i IN 1..w_TRN LOOP
  INSERT INTO wnd_table
  (tn_id, tn_root, tn_seq, tn_bseq, tn_flg, tn_xml)
  SELECT tn_id, tn_root, tn_seq, w_SEQ2, tn_flg,
```

```
                tn_xml
  FROM wrk_table
  WHERE SEQ = w_fSEQ;
//이동한 트랜잭션 flg set
  UPDATE wrk_table
  SET (tn_bseq, tn_flg) = (w_SEQ2, 1)
  WHERE  tn_seq = w_fSEQ;
  w_fSEQ = w_fSEQ +1;
END LOOP
END
```

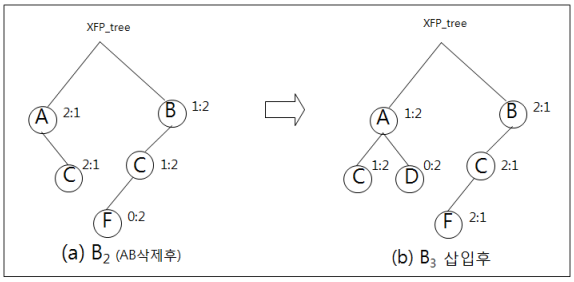
3. XFP_tree 구성 및 빈발구조 추출

본 논문에서는 트랜잭션들로 이루어진 스트림이 고정된 크기로 일괄(batch)처리되어 입력된다고 가정한다. 즉, 하나의 batch에는 일정한 수의 트랜잭션으로 구성되어 있다. XFP_tree의 빈발도 유지에 대한 예를 보이기 위해 각 batch에 포함되어 있는 트랜잭션을 B_1(t1, t2, t3), B_2(t4, t5, t6), B_3(t7, t8, t9)라 하고, 윈도우 사이즈를 2 batch라 가정한다. 그림 1은 순서대로 입력된 트랜잭션 T_1, T_2, T_3으로 구성된 배치 B_1에서 min_sup를 0.5로 하여 빈발구조와 경계구조(border), XFP_tree, 그리고 삽입된 배치 B_2의 빈발구조를 포함한 XFP_tree 를 보여준다.



(그림 1) B_1, B_2 입력의 XFP_tree

윈도우 사이즈를 2라 가정할 때 B_3이 입력되면 윈도우 사이즈를 고려하여 가장 오래전에 입력된 B_1의 정보를 XFP_tree로부터 삭제하고 최근 데이터인 B_2와 B_3의 빈발 구조정보를 XFP_tree에 유지한다. B_1의 삭제는 XFP_tree의 각 노드정보를 일괄적으로 이동하여 삭제하고 B_3의 빈발 구조를 XFP_tree에 추가한다. 그림 2의 (b)는 B_1의 정보를 삭제하고 B_3의 빈발 구조를 포함한 트리를 보인다.



(그림 2) B_1의 삭제 후 B_3의 입력에 대한 XFP-tree

한편 XFP_tree에서 빈발도를 만족하지 못하는 구조의 삭제와 더불어 경계 구조에 대한 관리도 추가적으로 이루어진다. B_1의 입력상태에서의 경계구조 집합은 {ABD, ABE, BC, BD}이고, B_2의 입력으로 인한 추가적인 경계구조항목은 {ACE, ACF, BA, BCG, BD, AD}이다. 스트림 데이터에서는 최근에 입력된 데이터를 중요하게 고려한다. 빈발구조 탐색은 현재 윈도우 범위에 속하는 스트림 XML 데이터들로 구성된 트랜잭션으로부터 빈발 구조를 마이닝 한다. 즉, 마이닝 요청시에 XFP_tree로부터 주어진 임계치 이상의 빈발 구조를 추출한다.

스트림 데이터가 입력된 임의의 시간 t에 대해 빈발 구조 집합을 나타내는 XFP_tree 상태를 그림 2의 (b)라 할 때, XFP_tree로부터 일정 임계치 이상의 빈발 구조만을 추출하고자 하는 최소 지지도를 min_support =3라 하면, 이를 만족하는 항목 및 이들의 빈발 지지도는 {A:3, B:3, C:6, D:3, F:3, AC:3, AD:3, BC:3, BF:3, CF:3, BCF:3}이다. 즉, 항목 C의 지지도는 에지 A-C에서의 1:2와 에지 B-C에서 2:1의 합을 나타낸다. 여기서 A-C의 1:2 의미는 B_2에서의 A-C빈발 지지도가 1이고 B_3에서는 2인 것을 나타낸다. 그러므로 현재 시점에서의 AC 빈발 지지도는 1과 2의 합인 3이 된다. 같은 방법으로 에지 B-C의 빈발 지지도는 3이다. XFP_tree로부터 주어진 지지도를 만족하는 경로 및 에지를 마이닝 할 때는 FP-growth와 같은 방법으로 빈발한 경로 및 에지를 발견한다.

4. 결론

본 논문에서는 트리거에 의한 슬라이딩 윈도우 기반의 XML 빈발 구조 추출 방법을 제안하였다. 스트림 데이터에 대한 빈발 구조의 집합의 XFP_tree를 구성하고, 트리거에 의한 최신의 데이터로부터 마이닝 결과를 추출하기 위해 일정한 윈도우 사이즈에 기준한 오래된 데이터에 대하여 일괄적으로 삭제하여 트리를 운영한다. 향후 연구에서는 제안된 기법과 기존 연구와의 정량적인 평가를 통해 트리거 이용에 대한 효율성에 대한 분석을 수행할 것이다.

참고문헌

[1] G. Chen, X. Wu, and X. Zhu, "Mining Sequential Patterns Across Data Streams", Univ. of Vermont Computer Science Technical Report(CS-05-04), 2005.

[2] T. Asai, K. Abe, S. Kawasoe, H.Sakamoto, et al., "Online Algorithms for Mining Semi-Structured Data Stream", In.Proc. ICDM, 2002.

[3] G.S. Manku, R. Motwani, "Approximate Frequency Counts over Data Streams", VLDB 2002.

[4] M.C. Hsieh, Y.H. Wu, A.L. Chen, "Discovering Frequent Tree Patterns over Data Stream", In Proc of SIAM, 2006.

[5] J.H. Hwang, M.S. Gu, "Finding Frequent Structures in XML Stream Data", Computational Science and Its Applications, ICCSA, 2009.