튜플 데이터 스트림에서 과부하 처리 기법

장중혁°, 박남훈, 이원석
연세대학교 컴퓨터과학과
e-mail : {jhchang, zyonix, leewo}@amadeus.yonsei.ac.kr

A Load Shedding Technique over a Data Stream of Tuples

Joong Hyuk Chang°, Nam Hun Park, Won Suk Lee
Dept. of Computer Science, Yonsei University

요 약

수행 과정에서의 성능 측면에서 기존의 데이터 스트림 처리 방법들은 주로 수행 과정에서의 저장 공간 사용량 및 데이터 객체당 수행 시간을 줄이는데 초점을 맞추고 있다. 이들 방법들에서 일정 시간 내에 처리될 수 있는 데이터 객체의 수보다 많은 데이터 객체가 발생된다면, 그들 중 일부는 실시간으로 처리되지 못한다. 본 논문에서는 튜플 데이터 스트림에서 발생빈도 기반의 중요 튜플 선별 방법을 제안한다. 이는 해당 데이터 스트림 처리 과정에서 전처리 과정으로 간주할 수 있다. 제안된 방법에서는 데이터 스트림의 변화를 고려하여 중요 튜플 선별을 위한 임계값을 적응적으로 조절한다. 이를 지원하기 위해서 튜플의 발생빈도 예측 방법을 제시한다.

1. 서론

최근 들어 데이터베이스 연구 그룹들의 주요 관심 대상이 이전과 같은 한정적인 데이터 집합에서 매우 빠른 속도록 발생되는 무한정의 데이터 집합인 데이터 스트림으로 옮겨가고 있다[1]. 특히, 센서 데이터 처리[2,3] 및 네트웍 트래픽 분석[4,5] 등의 분야에서 데이터 스트림 형태로 발생되는 데이터를 분석하기 위한 방법들이 다양하게 연구되고 있다.

데이터 스트림 처리를 위한 이전의 방법들은 처리 과정에서의 메모리 사용량을 일정범위로 한정적으로 제한하고 해당 데이터 스트림을 구성하는 단위 정보의 처리 시간을 최소화하는 것이 중요한 고려사항이다. 하지만, 각 데이터 스트림을 구성하는 단위 정보의 처리 시간을 최소화 하더라도 각 방법의 단위 시간당 처리 능력을 무한정 향상 시킬 수는 없다. 즉, 일정 단위 시간에 처리할 수 있는 단위 정보의 양은 한계를 가진다. 따라서, 데이터 스트림을 구성하는 단위 정보가 이들 방법들의 처리 능력보다 빠른 속도록 발생되는 경우에 이들 방법들은 해당 단위 정보를 실시간으로 처리할 수 없는 약점을 갖는다. 윈도우 접근법에서 분석 대상 범위가 제한된다 하더라도 하나의 윈도우 범위에서 발생되는 단위 정보의 양이 급격히 증가되는 경우 이러한 약점(drawback)은 여전히 존재한다.

본 논문에서는 튜플들로 구성되는 데이터 스트림에서 의미적으로 중요한 튜플들을 실시간으로 선별하여 해당 데이터 스트림을 위한 본처리 작업으로 전달하는 전처리 작업을 제안한다. 데이터 스트림에서 발생되는 튜플들의 중요성은 각 응용 분야의 특성에 따라 각기 다른 기준으로 측정될 수 있으며 본 논문에서는 튜플들의 발생빈도를 기준으로 중요성을 판단한다. 즉, 논문에서 제안하는 방법은 데이터 스트림에서 빈번히 발생하는 중요 튜플을 선별하여, 본처리 작업으로 전달한다. 또한, 제안된 방법에서는 선별되는 튜플의 수가 본처리 작업의 처리 능력에 근접하도록 지원하기 위해서 매개변수를 데이터 스트림의 변화 및 본처리 작업에서의 처리 능력에 따라 조절하는 동적 조절 과정을 제시한다.

2. 튜플 데이터 스트림

튜플 데이터 스트림 $D_k[A_1, ..., A_n]$는 해당 시점까지 발생된 모든 튜플들의 집합으로 정의된다. 튜플 $u_t=(t, a_1, a_2, ..., a_n)$ $(l \le k, 1 \le t \le l)$는 n 개의 속성으로 구성될 때, 데이터 스트림은 $D_k[A_1, ..., A_n]=<u_1, u_2, ..., u_l>$로 정의된다. 데이터 스트림 $D_k[A_1, ..., A_n]$에 포함되는 튜플의 총 수는 $|D|_k$로 나타낸다.

본 논문에서는 일정 임계값 이상의 지지도를 갖는 튜플을 ***중요 튜플***이라 정의하며, 이때 기준이 되는 임

계값을 ***선별 지지도*** $\Lambda \in (0,1)$라 지칭한다. 하나의 데이터 스트림에서 일정 시간동안 해당 데이터 스트림의 본처리 과정에서 처리되는 튜플의 수를 해당 ***본처리 과정의 처리 능력***이라 지칭하고 μ로써 나타낸다.

3. 데이터 스트림에서 중요 튜플 선별

본 장에서는 ***FBS*** (***Frequency-Based Selection***)이라 불리는 데이터 스트림에서의 중요 튜플 선별 방법을 제안한다.

3.1 출현빈도 수 관리

튜플 데이터 스트림에서 발생되는 각 튜플들의 발생빈도를 관리하기 위한 출현빈도 수 관리 구조는 서로 다른 튜플들 사이에 공통된 속성 정보들은 하나의 노드에서 표현하는 트리 구조이며, 하나의 튜플을 표현하는 단말노드는 (*Val, tStamp, cCnt, aCnt, pCnt, pVel, aAcc*) 정보를 관리한다.

- *Val* : 노드가 표현하는 값
- *cCnt* : 튜플의 현재 단위 시간(time slot)에서의 출현빈도 수 (단위 시간은 사용자에 의해 정의된 일정 크기의 시간을 의미)
- *aCnt* : 튜플의 단위 시간당 평균 출현빈도 수
- *tStamp* : *cCnt* 값을 마지막으로 갱신한 튜플의 시간 정보(timestamp)
- *PCnt* : 현재 단위 시간을 제외한 가장 최근 단위 시간에서의 튜플의 출현빈도 수
- *Pvel* : 튜플의 현재 단위 시간을 제외한 가장 최근 단위 시간에서의 *튜플 속도* (***튜플의 속도***는 해당 단위 시간에서의 지지도와 직전 단위 시간에서의 지지도 차이)
- *pAcc* : 튜플의 현재 단위 시간을 제외한 가장 최근 단위 시간에서의 *튜플 가속도* (***튜플의 가속도***는 해당 단위 시간에서의 튜플 속도와 직전 단위 시간에서의 튜플 속도 차이)

3.2 튜플의 발생 빈도 예측

t 번째 단위 시간에서 정보목록 (*Val, tStamp, cCnt, aCnt, pCnt, pVel, aAcc*)를 갖는 출현빈도 수 관리 트리의 단말 노드에 대해서, 해당 노드에 의해 표현되는 튜플의 다음 단위 시간(즉, *t+1* 번째 단위 시간)에서의 출현빈도 수 $cCnt'$는 현재 단위 시간에서의 *cCnt*, *pVel* 및 *aAcc* 정보를 이용하여 예측할 수 있다. 먼저 해당 튜플의 *t+1* 번째 단위 시간에서의 튜플 속도 예측값 V'는 해당 튜플의 현재 단위 시간에서의 튜플 속도 및 *aAcc* 값을 이용하여 다음과 같이 구할 수 있다.

$$V' = (\text{cCnt} - \text{pCnt}) + aAcc$$

다음으로 해당 튜플의 *t+1* 번째 단위 시간에서의 출현빈도 수 예측값 $cCnt'$는 *cCnt* 및 V' 정보를 이용하여 다음과 같이 구할 수 있다.

$$cCnt' = cCnt + V'$$

끝으로 해당 튜플의 *t+1* 번째 단위 시간에서의 출현빈도 수 예측값 *PS* 는 다음과 같이 구할 수 있다.

$$PS = \frac{\{aCnt \times (t-1)\} + cCnt + cCnt'}{|D|_k + M_{t-1}}$$

여기서 M_{t-1} 는 *t-1* 번째 단위 시간까지의 단위 시간당 평균 튜플 수를 나타낸다.

만약 하나의 튜플이 데이터 스트림의 첫번째 단위 시간에 발생했다면, 해당 튜플의 튜플 속도, 튜플 가속도 및 단위 시간당 평균 튜플 수가 정의되지 않는다. 따라서, 해당 튜플의 두번째 단위 시간에서의 출현빈도 수는 예측할 수 없다. 즉, 해당 튜플의 두번째 단위 시간에서의 지지도를 구할 수 없다. 이러한 상황을 고려하여 본 논문에서는 첫번째 단위 시간에서 발생한 튜플에 대해서 해당 튜플의 두번째 단위 시간에서의 지지도는 첫번째 단위 시간에서의 지지도와 동일한 것으로 간주한다.

3.3 지지도 분포 관리

선별 지지도 Λ를 동적으로 조절하기 위해서 각 단위 시간에서 튜플들의 지지도 분포를 ***지지도 분포 배열***을 이용하여 관리한다. 지지도 분포 배열은 다수의 버켓(bucket)으로 구성되며, 버켓의 크기가 r 로 설정되었을 때 t 번째 단위 시간에서의 지지도 분포 배열 B_t 는 $(m_i, N_i, \alpha_i, \beta_i)$로 구성되는 정보목록(entry)들의 집합으로 정의된다. 해당 지지도 분포 배열의 i 번째 버켓인 $B_t[i]$는 $\{1-r\times i\}$보다 크거나 같고 $\{1-r\times(i-1)\}$보다 작은 지지도를 갖는 튜플들의 수를 관리한다. 이때, 해당 튜플들의 수는 $B_t[i]$의 네가지 정보들 중에서 N_i 값에 의해 표현된다. 지지도 분포 배열의 각 버켓들에 의해 관리되는 튜플들의 지지도 범위는 상호간에 중복되지 않는다. 따라서, 지지도는 0 이상 1 이하의 값을 가지므로 지지도 분포 배열은 $\lceil 1/r \rceil$개의 버켓으로 구성된다($B_t = \{(m_i, N_i, \alpha_i, \beta_i) \mid 1 \le i \le \lceil 1/r \rceil\}$). i 번째 버켓인 $B_t[i]$에서 관리되는 지지도의 하한값(lower bound)은 $B_t[i]$의 네가지 정보들 중에서 m_i 에 의해 표현된다. 다시 말해서, 하나의 지지도 분포 배열 $B_t = \{(m_i, N_i, \alpha_i, \beta_i) \mid 1 \le i \le \lceil 1/r \rceil\}$에 대해서 $m_i = 1 - r \times i$ $(1 \le i \le \lceil 1/r \rceil - 1)$ 및 $m_{\lceil 1/r \rceil} = 0$ 을 만족한다. 지지도 분포 배열의 한 버켓에서 관리되는 튜플들 중에서 하나의 단위 시간에서 발생되지 않거나 또는 발생빈도가 낮은 튜플이 존재한다면 해당 튜플의 지지도는 감소되며, 경우에 따라서 해당 튜플이 속하는 버켓이 변경되어야 한다. 즉, 보다 낮은 지지도를 갖는 튜플을 관리하는 버켓으로 이동되어 관리되어야 한다. 본 논문에서는 이러한 튜플을 *이동 튜플*(*shifting tuple*)이라 지칭한다. i 번째 버켓인 $B_t[i]$에서 관리되는 N_i 개의 튜플들 중에서 현재 단위 시간에서 이동 가능성이 있는 이동 튜플의 수는 $B_t[i]$의 네가지 정보들 중에서 α_i 에 의해 표현된다. 이 값은 N_i 보다 작거나 같은 값을 가지며 이전 단위 시간 (즉, *t-1* 번째 단위 시간)에서 구해진다. 한편, 현재 단위 시간에서의 이동 가능성은 이전 단위 시간에서 구하는 예측 값으로서, 해당 값을 구하는데 있어서 현재 단위 시간에서 발생되는 튜플의 수는 이전 시점까지의 단위 시간당 평균 튜플 수와 동일한 것으로 간주한다. 더불어, $B_t[i]$의 네가지 정보들 중에서 β_i 는 다음 단위 시간인 *t+1* 번째 단위 시간에서의 이동 튜플 수 예측 값을 표현하며, 이 값은 현재 단위 시간이 종

료되고 다음 단위 시간을 시작 되는 시점에 α_i 값으로 부여된다.

3.4 중요 튜플 선별

튜플 데이터 스트림에서 중요 튜플 선별 방법인 *FBS* 방법은 네 단계로 구성된다. 즉, 발생빈도 갱신 단계, 튜플 선별 단계, 지지도 분포 배열 갱신 단계 및 임계값 갱신 단계로 구성된다. 속성 순서 ρ 및 연관된 출현빈도 수 관리 트리를 갖는 데이터 스트림 D_k 에서 새로운 튜플 $u_k=(k, a_1, a_2, ..., a_n)$가 t ($t\geq 2$)번째 단위 시간에서 발생되었을 때, 해당 튜플의 정렬 튜플이 FBS 방법에 의해 처리되며 임계값 갱신 단계를 제외한 모든 단계가 순차적으로 수행된다. 임계값 갱신 단계는 각 단위 시간이 종료됐을 때에만 수행된다. 선별 지지도 초기값 Λ_1 은 사용자에 의해 설정된다.

[발생빈도 갱신 단계] 튜플 u_k 에 연관된 단말 노드가 출현빈도 수 관리 트리에 존재한다면 해당 노드의 각 정보들이 갱신된다. 반면, 해당 튜플에 연관된 노드가 존재하지 않는다면 새로 추가한다. (각 정보의 자세한 갱신 과정은 지면 관계로 생략하였다.)

[튜플 선별 단계] 출현빈도 수 관리 트리에서 정보목록 (*Val, tStamp, cCnt, aCnt, pCnt, pVel, aAcc*)를 갖는 단말 노드에서는 튜플의 선별 여부를 결정하기 위해서 해당 노드가 표현하는 튜플의 지지도를 다음과 같이 구한다.

$$\text{튜플 } u_k\text{의 갱신된 지지도} = \frac{\{aCnt\times(t-1)\}+cCnt}{|D|_k}$$

갱신된 지지도가 선별 지지도 Λ보다 크거나 같으면 해당 튜플은 중요 튜플로 간주되며, 선별되어 해당 데이터 스트림의 본처리 과정으로 전달된다. 반면에 갱신된 지지도가 Λ보다 작다면 해당 튜플은 본처리 과정으로 전달되지 않고 무시된다. 한편, 하나의 단위 시간에서 발생되는 튜플들 중에서 Λ보다 크거나 같은 지지도를 갖는 튜플의 수가 본처리 과정의 처리 능력 μ보다 많은 경우에는 시간적으로 먼저 발생되는 μ개의 튜플이 선별된 이후에 발생되는 튜플들은 비록 Λ보다 큰 지지도를 갖는 중요 튜플이라 할지라도 선별되지 않고 무시된다.

[지지도 분포 배열 갱신 단계] 이어서 $t+1$ 번째 단위 시간에서의 해당 튜플의 발생빈도가 3.2 절에서 기술한 바와 같이 예측된다. 해당 튜플의 지지도 변화로 인해서 해당 튜플을 관리하는 버켓이 변경돼야 하는 경우 지지도 분포 배열에 대한 갱신 작업이 수행된다. 현재 단위 시간의 지지도 분포 배열 B_t에서 p 번째 버켓에 속하는 튜플이 다음 단위 시간에서 q 번째 버켓에 포함될 것으로 예측될 때, $B_t[q].N$ 의 값은 1 만큼 증가되는 반면 $B_t[p].N$ 의 값은 1 만큼 감소된다. 더불어, 만약 $p<q$ 인 경우 (즉, p 번째 버켓에서 관리되는 튜플의 지지도 하한값이 q 번째 버켓에서 관리되는 튜플의 지지도 하한값보다 큰 경우), $B_t[p].\alpha$값이 1 만큼 감소된다. 하지만, 해당 값이 0 이라면 더 이상 감소되지 않는다. 다음 단위 시간에서의 지지도 예측값이 변경되더라도 해당 튜플이 관리돼야 할 버켓이 변경되지 않는 경우 지지도 분포 배열에 대한 갱신 작업을 수행되지 않는다. 또한, 해당 튜플의 지지도를 관리하는 연관된 버켓이 현재 지지도 분포 배열에 존재하지 않는다면 해당 버켓이 새로 추가된다. 이 경우에는 해당 튜플이 이전에 속하던 버켓이 없으므로 새로 추가되는 버켓의 N 값만 1 로 설정된다.

한편, 해당 튜플이 다음 단위 시간에서 출현하지 않는 경우의 지지도 예측값 PS'을 *aCnt*, *cCnt* 및 *t-1* 번째 단위 시간까지를 기준으로 구해진 단위 시간당 평균 튜플 수 M_{t-1} 값으로부터 다음과 같이 구해진다.

$$PS' = \frac{\{aCnt\times(t-1)\}+cCnt}{|D|_k + M_{t-1}}$$

PS' 값을 이용하여 해당 튜플이 다음 단위 시간에서 전혀 발생되지 않는 경우에 속하게 되는 버켓이 변경되는지를 판단할 수 있다. 이러한 상황에서 해당 튜플이 속하는 버켓이 변경될 것으로 판단된다면 다음 단위 시간에서의 q 번째 버켓의 이동 튜플 수(즉, $B_t[q].\beta$)가 1 만큼 증가된다. 현재 단위 시간에서 발생한 모든 튜플에 대해서 지지도 분포 배열에 대한 갱신 작업이 완려된 후 $t+1$ 번째 단위 시간을 위한 지지도 분포 배열 B_{t+1}을 얻게 된다.

앞서 기술한 바와 같이 첫번째 단위 시간에서는 하나의 튜플을 위한 정보목록 (*Val, tStamp, cCnt, aCnt, pCnt, pVel, aAcc*) 중에서 *aCnt*, *pCnt*, *pVel* 및 *aAcc* 값이 0 이며, 단위 시간당 평균 튜플 수가 정의되지 않는다. 따라서, 각 튜플에 대해서 두번째 단위 시간에서 해당 튜플이 출현하지 않는 경우의 지지도 예측값 PS'를 구하는 것이 불가능하다. 하지만, 두번째 단위 시간에서 첫번째 단위에서와 동일한 수의 튜플들이 발생하는 것으로 가정하는 경우 각 튜플의 지지도가 절반으로 감쇠된다. 이에 근거하여, 첫번째 단위 시간에서 발생한 각 튜플들은 두번째 단위 시간에서 이동 튜플로 간주한다. 따라서, 첫번째 단위 시간에서는 지지도 분포 배열의 각 버켓들에 있어서 β 값이 N 값과 동일한 것으로 간주한다.

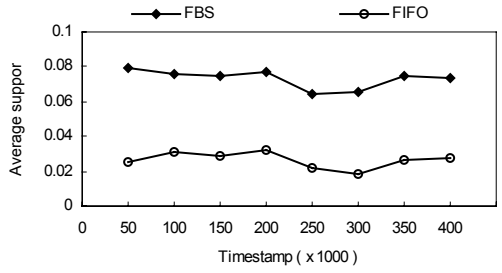
[임계값 갱신 단계] t 번째 단위 시간이 종료되었을 때 $t+1$ 번째 단위 시간에서의 지지도 분포 배열 $B_{t+1}=\{(m_i, N_i, \alpha_i, \beta_i) \mid 1\leq i\leq\lceil 1/r\rceil\}$이 구축된다. 해당 시점에서 B_{t+1} 의 각 버켓에서 유지되는 정보들 중에서 α 값은 각 버켓에서 관리되는 튜플들 중에서 t 번째 단위 시간에서 실제로는 발생하지 않은 이동 튜플의 수를 나타낸다. 해당 튜플들은 t 번째 단위 시간에서 전혀 발생되지 않는 경우에 소속 버켓이 변경될 것으로 예측된 이동 튜플이며, 해당 단위 시간에서 실제 발생하지 않고 $|D|_k$ 의 증가로 인해 지지도가 감소되었다. 따라서, 해당 튜플이 속할 버켓이 변경되어야 한다. 결론적으로, B_{t+1} 의 i 번째 버켓에 대해서 해당 버켓의 N 값이 α 값 만큼 감소되어야 한다. 즉, $B_{t+1}[i].N=B_{t+1}[i].N-B_{t+1}[i].\alpha$ ($1\leq i<\lceil 1/r\rceil$). 반면에 $i+1$ 번째 버켓의 N 값은 i 번째 버켓의 α 값 만큼 증가된다. 즉,

$B_{t+1}[i+1].N = B_{t+1}[i+1].N + B_{t+1}[i].\alpha\ (1 \le i < \lceil 1/r \rceil)$.

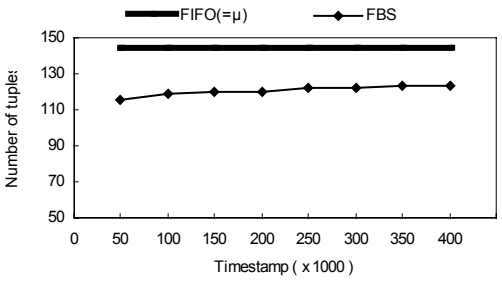
이어서 지지도 분포 배열 B_{t+1} 및 본처리 단계의 처리 능력 μ에 근거하여 $t+1$ 번째 단위 시간에서의 선별 지지도 Λ_{t+1} 값이 새로 구해진다. 과부하 처리를 위한 중요 튜플 선별 작업에 의해 선별되는 튜플의 수는 본처리 단계의 처리 능력 μ을 넘지 않아야 한다. 따라서, $\sum_{l=1}^{k} B_{t+1}[l].N \le \mu$ 및 $1 \le k \le \lceil 1/r \rceil$을 만족하는 k 에 대해서 $B_{t+1}[k].m$ 값이 새로운 선별 지지도 Λ_{t+1} 로 구해진다. 끝으로, 지지도 분포 배열의 각 버켓에서 β 값이 α 값으로 부여된다. 다시 말해서, $t+1$ 번째 단위 시간에서의 이동 튜플의 수가 B_{t+1} 의 각 버켓의 α 값으로 표현되며, 해당 β는 $t+2$ 번째 단위 시간에서의 이동 튜플 수 예측값을 관리 하기 위해서 0 으로 초기화 된다. 이러한 과정을 통해서 새로운 선별 지지도 Λ_{t+1} 가 구해지며, 지지도 분포 배열도 정제된다.

4. 실험 결과

FBS 방법의 성능을 평가하기 위해서 웹포털 사이트의 사용자 접근 로그로부터 생성된 *WebLog* 데이터 집합을 사용하였다. 하나의 웹 페이지에 대한 접속 기록을 하나의 튜플로 정의하였으며, 해당 데이터 집합은 10 개의 속성을 갖는 5,056,000 개의 튜플로 구성된다. 각 튜플들은 시간 순서에 따라 정렬돼 있으며 단위 시간당 튜플 생성률은 시간 변화에 따라 변한다. 한편, 10 개의 속성들 중에서 출현빈도 수 관리를 위해서 4 개의 속성을 활용하였으며, 이때 튜플의 지지도 평균값은 0.02917 이다. 모든 실험에서 선별 지지도 초기값은 0 으로 설정하였다.



(그림 1) 선별된 튜플의 평균 지지도



(그림 2) 선별된 튜플의 수

FBS 방법에 의해 선별되는 튜플의 수 및 지지도 평균 값을 일반적인 선별 방법인 *FIFO*(first-in-first-out) 방법과 비교하였다. 본 실험에서 단위 시간의 크기는 100 으로 설정되었다. 단위 시간당 튜플의 최소 발생 수는 1,216 개 이며, 최대 발생 수는 1,664 개이다. 해당 데이터 집합을 위한 본처리 과정의 처리 능력 μ 는 145 로 가정한다. 즉, 단위 시간당 145 개의 튜플을 처리할 수 있는 것으로 가정한다. 지지도분포 배열의 버켓 크기는 0.001 로 설정하였다. 해당 데이터 집합에서 발생되는 일련의 튜플들은 발생 시간을 기준으로 8 개의 구간으로 구분하였다. 그림 1 은 두 가지 방법에 의해 선별되는 튜플의 평균 지지도를 각 구간별로 보여준다. 그림에서 보듯이 *FBS* 방법에 의해 선별되는 튜플들의 평균 지지도가 FIFO 방법에 의한 경우의 평균 지지도보다 크다. 그림 2 는 선별되는 튜플의 수를 보여준다. FBS 방법에 의해 선별되는 튜플의 수는 본처리 과정의 처리 능력 μ 에 미치지 못한다. 하나의 단위 시간에서 선별 지지도는 이전 단위 시간까지의 튜플 발생 상황을 고려하여 설정된 값으로서 해당 단위 시간에서 지지도가 높은 튜플이 적게 발생되는 경우 실제로 선별되는 튜플의 수는 μ 보다 작다.

5. 결론

본 논문에서는 튜플 데이터 스트림에 대한 전처리 과정으로서 발생빈도를 기반으로 중요 튜플을 선별하는 적응형 선별 방법을 제안하였다. 논문에서 제안된 방법은 데이터 스트림 처리를 위한 본처리 과정의 종류에 무관하게 적용될 수 있으며, 특히나 데이터 스트림에 대한 조인 질의나 마이닝 작업에서 매우 효과적으로 적용될 수 있다. 특히, 제안된 방법에서는 중요 튜플 선별의 기준이 되는 임계값을 데이터 집합에서 발생되는 튜플들의 발생빈도를 고려하여 동적으로 조절한다. 선별 지지도의 동적 조절 과정을 통해 선별되는 튜플의 수를 본처리 과정의 처리 능력에 최대한 근접하도록 지원한다.

참고문헌

[1] J. Kang, J.F. Naughton, and S. D. Viglas. Evaluating Window Joins over Unbounded Streams. In *Proceedings of the 19th International Conference on Data Engineering*, 2003, pp. 341-352.

[2] D.J. Abadi, D. Carney, U. Cetinternel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S.B. Zdonik. Aurora: A New Model and Architecture for Data Stream Management. VLDB Journal, Vol. 12, No. 2, 2003, pp. 120-139.

[3] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varma. Query Processing, Approximation, and Resource Management in a Data Stream Management System. In *Proceedings of the 1st Biennial Conference on Innovative Data Systems Research*, 2003, pp. 245-256.

[4] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. QuickSAND: Quick Summary and Analysis of Network Data. *DIMACS Technical Report 2001-43*, 2001.

[5] M. Sullivan and A. Heybey. Tribeca: A System for Managing Large Databases of Network Transactions. In *Proceedings of USENIX Annual Technical Conference*, 1998.