韓國컴퓨터情報學會 論文誌
第17卷 第2號, 2012. 2.

2012-17-2-4-2

스트림 데이터에서 슬라이딩 윈도우를 사용한 조인 연산의 효율에 관한 연구

양 영 휴*

A Study on the Efficiency of Join Operation On Stream Data Using Sliding Windows

YoungHyoo Yang *

요 약

이 논문은 슬라이딩 윈도우를 사용하는 스트림 데이터에서 모든 조인 연산의 상태를 저장하기에 메모리가 충분하지 않을 경우에, 연속적인 슬라이딩 윈도우 조인 연산의 근사치 답을 구하는 문제에 대한 연구이다. 근사치를 구하는 두 가지 방법으로는 최대 부분집합으로 근사치를 구하는 방법과 조인 결과에서 임의의 결과를 택하는 방법이 있다. 전자는 잃어버리는 튜플의 수를 최소화 하고, 후자는 조인의 결과가 집계로 나타날 때 사용된다. 이 논문에서는 임의의 입력 데이터에 슬라이딩 윈도우가 사용되는 경우 두 가지 방법으로 얻는 근사치 모두 효율적이지 못함을 보여준다. 기존의 최대 부분집합에 의해 근사치를 구하는 모델에서는 빈도-기반 모델을 사용하였는데. 샘플링이 문제가 되었다. 오히려 스트림 도착한 이후의 연령-기반 모델이 많은 응용분야에서 더 적절하게 사용 될 수 있음을 보여주고 있다. 이 논문에서는 최대 부분 집합과 임의의 결과라는 두 가지 근사치 측정법을 분석, 그 효율성을 비교하여 보여 준다. 또한, 메모리가 제한 되어있는 환경에서 다중 조인 연산이 수행 될 경우에, 어떤 경우에도 근사치 측정을 최적화할 수 있도록, 조인 연산 전체에 필요한 메모리를 적절하게 할당하는 알고리즘의 효율성을 분석한다.

▸ Keyword : 슬라이딩 윈도우, 조인, 최대 부분 집합, 임의의 결과, 빈도-기반 모델, 연령-기반 모델, 다중 조인 연산

Abstract

In this thesis, the problem of computing approximate answers to continuous sliding-window joins over data streams when the available memory may be insufficient to keep the entire join state. One approximation scenario is to provide a maximum subset of the result, with the objective

• 저자 : 양 영휴
• 투고일 : 2011. 12. 23, 심사일 : 2012. 01. 10, 게재확정일 : 2012. 01. 18.
* 한양여자대학 정보경영과(Dept. of Information Management, Hanyang Women's University))

of losing as few result tuples as possible. An alternative scenario is to provide a random sample of the join result, e.g., if the output of the join is being aggregated. It is shown formally that neither approximation can be addressed effectively for a sliding-window join of arbitrary input streams. Previous work has addressed only the maximum-subset problem, and has implicitly used a frequency based model of stream arrival. There exists a sampling problem for this model. More importantly, it is shown that a broad class of applications for which an age-based model of stream arrival is more appropriate, and both approximation scenarios under this new model are addressed. Finally, for the case of multiple joins being executed with an overall memory constraint, an algorithm for memory allocation across the join that optimizes a combined measure of approximation in all scenarios considered is provided.

▸ Keyword : sliding-window, join, maximum subset, arbitrary result, frequency-based model, age-based model, multiple join operation

I. 서 론

데이터 스트림 시스템은 많은 데이터 량에 비례한 만큼의 메모리 사용이 현실적으로는 불가능함에도 불구하고 이 많은 량의 데이터에 대한 여러 가지 질의에 대하여 즉각적인 온라인 결과를 요구하게 된다[1,2]. 이를 해결하는 데에는 두 가지 접근법이 있다. 최상의 수행결과를 내기 위하여 메모리를 사용, 정확한 결과 대신 근사치의 질의 결과를 제공하는 방법[1,3,4]과, 입력 데이터 속도는 무시하고 정확한 결과를 위하여 디스크를 사용하는 방법[1,5]이 있다. 이 논문에서는 데이터 스트림 시스템에서 근사치 결과를 제공하는데 중점을 둔 슬라이딩 위도우 조인 연산[2]의 메모리 제한적인 실행에 대하여 논한다.

두 개의 스트림 S_1 과 S_2 에서의 연속 슬라이딩 윈도우 조인 연산 S_1 [W_1]⋈ΘS_2 [W_2]를 고려해 보자. 윈도우 W_1 와 W_2 는 각각의 스트림에서 튜플 기반(최근 1000개의 튜플)이거나 시간 기반(최근 10분 동안 도착한 튜플)의 가장 최근의 튜플들로 이루어져 있을 것이다. 이때 조인 연산의 결과는 스트림 S_1 과 S_2 의 각각의 윈도우에 동시에 나타나있는 튜플들 중에서 조인 프레디키트 Θ 를 만족시키는 튜플들의 쌍으로 이루어진다. 일반적으로 조인연산을 정확히 수행하기 위해서는 두 개 윈도우의 전체 내용이 항상 보관 되어야 한다. 그러나 대용량의 데이터 스트림의 큰 사이즈 윈도우에 대한 조인 연산을 하게 된다면 당연히 전체 윈도우의 내용들을 메모리에 보관하기가 어려워진다.

1. “최대 부분 집합” 결과: 어플리케이션이 결과의 최대 부분 집합을 얻는 것이 더 유용하다면 튜플을 선택적으로 버릴 수 있다 (로드-쉐딩[1,6]이라 알려진 이 방법은 생성되는 조인 결과의 크기를 최대화 하는데 목적이 있다).
2. 샘플링 한 결과: 대용량의 조인 결과보다는 결과에 대한 임의의 부분집합이 더 나을 경우가 있다. 예를 들면, 집계를 내는 조인이 진행되고 있다면, 일관적이고 치우치지 않는 참 집계 값을 측정하기 위해 샘플을 대신하여 사용 할 수 있다.

이 논문에서는 연령-기반 모델에서의 최대 부분 집합 문제와 빈도-기반 모델과 연령-기반 모델에서의 샘플링 문제에 대한 비교를 한다. 또한 여태까지의 2차원 슬라이딩 윈도우 조인 연산만이 아니라 현실적으로는 동시에 여러 질의들이 실행하여야 하므로, 다중 조인에서의 메모리 할당 문제도 논의 한다. 메모리 할당량에 따른 실험을 통하여 모든 조인에 있어서 최대 근사치 에러를 최소화 할 수 있는 최적의 메모리 할당 구조는 어떤 것인지를 제시한다.

II. 관련 연구

이전까지의 메모리-제한 조인 연산에서는 스트림 도착의 빈도-기반 모델을 사용하여 그 결과에 최대 부분 집합만을 포함 하였다. 이 모델에서는 각 조인-속성의 값들은 스트림의 실현 치에 있어서 일반적으로 적용 되는 고정 빈도가 있었다. 이러한 빈도는 조인 연산 결과의 크기를 최대화 시키기 위하여 튜플의 보유 여부를 결정짓는 부하 차단을 결정하는데 쓰인다. 그러나 빈도-기반 모델이 최대 부분 집합의 근사치 문제를 해결하기 위한 최적의 방법은 아니다. 메모리 제약이 있는 환경에서 임의의 스트림에 대하여 슬라이딩 윈도우 조인이 실행될 때는 효율적인 근사치를 얻어내기 어렵기 때문이다 .

최대 부분 집합 문제에서는 오프라인의 최적의 알고리즘에 비해서 모든 온라인 알고리즘은 상대적으로 작은 부분집합 결과를 돌려주며, 샘플링 문제의 경우 어떤 알고리즘도 조인 결

과의 논제로(nonzero)의 균일한 임의의 샘플을 보장하지는 못한다. 이 문제의 해결을 위해서는 스트림 도착에 관한 모델을 필요로 한다. 지금까지의 연구에서는 빈도-기반 모델이 외래키 조인등의 연산에서는 적절하지 못함을 보여주었다. 이에 로드-쉐딩 결정에 더 적합한 연령-기반 모델이 제안 되었다. 연령-기반 모델에서 조인의 다중도는 조인 속성의 값 보다는 도착 후 시간 즉, 연령에 따라 결정된다.

데이터 스트림에 관한 많은 연구가 있어 왔는데, 이 논문에서는 사용가능한 메모리가 제한되어 있을 때 주어진 질의에 대하여 근사치 결과를 주는 경우만 고려하도록 한다.

이 작업은 크게 두 가지로 분류 된다. 하나는 조인 연산의 최대 부분 집합 근사치 계산에서 사용되는 로드-쉐딩 기법으로 가장 간단한 랜덤 로드-쉐딩은 [2]에서 논의 되었다. [2]에서는 미래에 도착할 튜플들이 이미 알려져 있는 오프라인 로드-쉐딩들 다루고 있으며, 빈도-기반 모델위의 온라인에 대한 발견적 해법을 제공한다. 로드-쉐딩의 또 다른 스트림 모델은 확률과정을 사용한다. 비록 이 모델이 더 일반적이긴 하지만, 기본적인 초점은 하나의 스트림에 도착한 튜플은 또 다른 스트림에 이미 도착해 있는 튜플들과는 무관함을 가정한다. 그러나 대부분의 시나리오는 이런 독립성을 보장하지 않으며, 더욱이 스트림을 관찰하는 것만으로 일반적인 확률 과정을 이끌어 낸다는 것은 적절하지 않다.

다른 하나는 조인연산의 집단화 질의에 대해 확률적 에러가 있는 근사치 값을 제공해 주는 랜덤화된 스케치-기반 해결책이다. 이 기법은 많은 응용에서 필요로 하는 슬라이딩 윈도우 조인이나 윈도우 된 집적 연산으로 까지 확장되지는 않는다. 비록 스트림에서의 명백한 삭제는 다룬다 할지라도 슬라이딩 윈도우에 의해 발생되는 내포된 삭제는 처리하지 못한다. 데이터 스트림 시스템은 CPU 처리 속도보다 더 빠르게 튜플 들이 입력되는 CPU 제한적인 시스템도 있을 수 있다. CPU 제한적인 환경에서의 로드-쉐딩은 [6,7]에서 연구되었다. 윈도우 시스템에서의 샘플링에 관한 연구는 많이 있어 왔지만, 기존의 샘플링 기법은 하나의 릴레이션에 대해 반복적인 액세스 혹은 색인이 요구되므로 스트림 환경에서는 적합하지 못하다.

III.최대 부분집합, 샘플링과 메모리 할당

1. 기본 개념과 모델

이 절에서는 데이터 스트림에서 실행되는 연속 질의 처리의 기본 모델을 설명한다. 스트림 Si, i=1,2 에는 매시간 여러 개의 튜플이 들어오게 되지만, 여기서는 시간당 평균 ri의 속도로 튜플이 도착한다고 가정하자. Si[Wi]는 스트림 Si에서의 윈도우를 나타내며, Wi는 시간당 윈도우의 길이를 표시하는 시간-기반 윈도우이다. 튜플 s가 시각 t를 기준으로 [t-Wi, t] 주기에 스트림 Si에 도착했다면, s는 Si[Wi]에 속한다. 또한 s가 t-k에 도착했다면 s는 연령 k로 나타낸다. 일반적으로는 시간-기반 윈도우를 고려하게 되지만, 시간당 하나의 튜플이 스트림에 도착한다고 가정 했을 때 이는 튜플-기반 윈도우와 같아진다.

그림 1.에서 나타낸 바와 같이 여기서 고려하는 기본 질의는 S1[W1]⋈A S2[W2]로 표시되는 공통의 속성 A를 가지는 두 개의 스트림 S1, S2에 대한 슬라이딩 윈도우 등가조인이다 . 이 조인 연산의 결과는 s1∈S1, s2∈S2인 튜플들이 시간 t에 s1∈S1[W1], s2∈S2[W2]에 속하면서 s1.A=s2.A를 만족하는 모든 튜플들의 쌍으로 이루어진다.

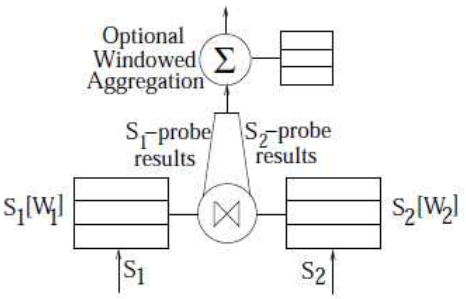

그림 1. 집적연산이 있는 슬라이딩-윈도우 조인
Figure 1. Sliding-window join with aggregation

1. When a new tuple s arrives on S_1
2. Update $S_2[W_2]$ by discarding expired tuples
3. Emit $s \bowtie_A S_2[W_2]$
4. Add s to $S_1[W_1]$

그림 2. 슬라이딩-윈도우 조인 실행
Figure 2. Sliding-window join execution

그림 2.에서는 슬라이딩-윈도우 조인 연산을 할 때, S1에 새롭게 도착하는 튜플에 대해 어떤 단계별 연산이 따라야 하는지를 보여 주고 있다. 스트림 조인 결과에는 윈도우에서의 집적 연산 결과도 더해진다. 이때 메모리 사용량은 고려되고 있지 않으며, 스트림 데이터의 조인 연산에 있어서 메모리 할당과 집적 연산의 상충 관계를 비교 분석하는 것은 좋은 연구 과제가 될 것이다.

1.1 빈도-기반 스트림 모델

슬라이딩 조인 연산 S1[W1]⋈A S2[W2]를 계속 고려해 보자. 조인 속성 A의 도메인을 D라고 했을 때, 빈도-기반 모델은 다음과 같이 정의 할 수 있다.

정의 1.1 (빈도-기반 모델) 값 v∈D에 있어서, S1에 도착하는 튜플들의 고정 분할 f1(v)와 S2에 도착하는 튜플들의 고정 분할 f2(v)는 속성 A에 대해 v의 값을 가진다.

단위 시간 내에 S2에 도착하는 튜플들의 평균 속도가 r2라고 가정 했을 때, 튜플 s1∈S1에 대해 생성되는 S1-probe 조인 튜플의 수는

$$E[n1(s)] = r2 \cdot W1 \cdot f2(s.A) \quad \cdots\cdots (1)$$

로 나타내어진다.

1.2 연령-기반 스트림 모델

온라인 옥션과 같은 경우에 빈도-기반 모델은 적합하지 않다. 이런 경우의 질의는 S1(seller), S2(bid) 스트림에 대하여 이루어진다. 예를 들어 지난 5일간 하나의 옥션에 대한 bid 수에 대한 질의가 들어오면, 옥션이 끝날 때까지의 S1에 대한 슬라이딩 윈도우와 S2의 슬라이딩 윈도우 조인이 이루어져야 하며, 그 후 바로 5일간 윈도우에 대한 집계 연산이 수행 된다.

만일 S1 윈도우의 모든 튜플들을 저장할 만큼 메모리가 충분하지 않고, 이러한 환경에서 전력 균등 분배 결정을 위해 빈도-기반 모델의 사용을 가정해 보자. 옥션 id는 고유하므로 스트림 S1에서 한번씩만 나타난다. 스트림 S2에 도착하는 옥션 id들은 현재 진행 중인 옥션들의 id 이므로 이 집합은 시간에 따라 달라진다. 모니터링을 통해서도 고정된 빈도 분배는 추측이 불가능한 것이다. 이런 경우 빈도-기반 모델의 전력 균등 분배 구조는 단순하게 새로운 튜플은 저장하고 오래된 튜플은 버린다. 바로 이것이 잘못된 방법인데 이는 마감되려는 옥션이 제일 많은 bid를 가지게 되는데, 이것들은 상대적으로 제일 오래된 것들이기 때문이다. 이러한 시나리오를 고려해서 연령-기반 모델을 다음과 같이 정의한다.

정의 1.2 (연령-기반 모델) 튜플 s∈S1에 대해서, S1-probe 조인 튜플은 다음 두 가지 조건을 따른다.

1. s에 의해 생성되는 S1-probe 조인 튜플의 수는 s와는 무관한 상수이며, n1으로 표기된다.
2. s의 S1-probe 조인 튜플 n1중에서 s의 연령이 k-1에서 k 사이에 있을 때 p1(k)가 생성된다.

튜플 s′∈S2에 의해 생성되는 S2-probe 조인 튜플들도 마찬가지이다.

이 모델에 따르면 하나의 튜플이 생성하는 조인의 수는 조인-속성의 값과는 무관하며, 윈도우 내에 있는 튜플의 연령과 함수 관계에 있다. 옥션과 같은 환경에서는 조인-속성 집합의 값이 시간에 따라 변한다. 그러므로 튜플 s∈S1의 ni(s) 값이 s.A에 의존적이라 할지라도 스트림을 모니터링 함으로써 그 값을 추측해내기는 어렵다.

다양한 연령 커브: 연령 커브는 Si[Wi]내의 튜플들이 점점 오래 되어가면서 어떻게 조인 튜플을 생성해내는 지를 나타낸다. 연령-기반 모델을 고수하는 응용분야에 특성에 따라 각기 다른 연령 커브를 보이게 된다.

- Increasing - 옥션 등의 응용 분야
- Decreasing - 주문/공급의 응용 분야
- Bell -두 개의 센서로 읽은 스트림의 조인 연산

2. 최대 부분 집합

그림 2.에서 보여준 S1[W1]⋈A S2[W2] 연산의 기본 알고리즘에서 만일 메모리 용량이 모자란다면 두 가지 방향으로 이 알고리즘을 수정해야 한다. 만료된 튜플들이 저장되어 있는 메모리를 비우기 위하여 S_1 [W_1]과 S_2 [W_2]을 갱신하여야 하며, 중요한 것은 S_1 [W_1]에 s를 추가하기에 메모리가 충분하지 않다는 것이다. 이때에 필요한 것은 s를 S_1 [W_1]내에 받아들일 것인지 아니면 버릴 것인지를 결정 하는 것이다. 만약 받아들인다면 기존의 어떤 튜플을 버릴 것인지도 결정해야 하는 것이다. 이런 결정을 하는 알고리즘을 로드-쉐딩 전략이라 부른다. 이 전략으로 인하여 오직 참 값의 결과 일부만이 실제로 생성되는 것이며, 이를 리콜 (recall)이라 하고 다음과 같이 표시한다.

$$Recall(t) = \frac{\text{Number of result tuples produced up to time } t}{\text{Number of actual result tuples up to time } t}$$

정의 2.1 (최대 부분 집합 Problem). 슬라이딩 윈도우의 조인 S1[W1]⋈A S2[W2]을 위한 메모리의 양이 고정되어 있다고 가정 할 때, limt→∞ Recall(t)를 최대화 할 온라인 로드-쉐딩 전략을 의미한다.

2.1 경도 결과(Hardness Result)

같은 용량의 메모리를 사용하면서 결과적으로 최대의 리콜을 생성해 낼 수 있다면, 로드-쉐딩 전략은 최적화 된 것이다. 유계의 시스템에서 오프라인 전략은 미래에 도착할 모든 튜플 들을 인지한 후에 로드-쉐딩을 결정할 수 있도록 되어 있다. 임의의 스트림에 대해서 어떤 온라인 전략도 최적화된 오프라인 전략을 넘어설 수는 없다.

S가 스트림 S1과 S2에 도착한 순서대로의 튜플 들의 집합이라 할 때, Ron(M,S)은 순차 S에 대해 M 메모리에서 수행한 온라인 전략의 결과로 얻은 리콜을, Roff(M,S)는 최적화된 오프라인 전략의 결과로 얻은 리콜이라면, Ron(M,S)/Roff(M,S)≤k를 만족할 때 이 온라인 전략은 k-competitve하다고 말한다.

정리 2.2 최대 부분 집합 문제에서 어떤 온라인 전략도 입력 순서의 데이터 길이와 상관없이 어떤 k에 대해 k-competitive할 수 없다.

표1. 두 모델의 조인 튜플 생성
Table 1. Rate of join tuples produced in two models

모델	Ri: Si-probe 조인튜플 생성률	
연령-기반	ri=1	$R_i = \begin{cases} M_i \frac{C_i(k_i^{opt})}{k_i^{opt}} & \text{if } M_i \le k_i^{opt} \\ C_i(M_i) & \text{if } M_i > k_i^{opt} \end{cases}$
	ri〉1	$R_i = \begin{cases} M_i \cdot \frac{C_i(k_i^{opt})}{k_i^{opt}} & \text{if } M_i/r_i \le k_i^{opt} \\ r_i \cdot C_i(M_i/r_i) & \text{if } M_i/r_i > k_i^{opt} \end{cases}$
빈도-기반	$R_1 = r_1 r_2 W_1 \sum_{j=1}^{t} f_1(v_j) f_2(v_j)$	

표2. 두 모델에서 조인의 리콜
Table 2. Recall of the join in two models

모델	조인의 리콜	
연령-기반	ri=1	
	ri〉1	$\frac{R_1+R_2}{r_1 n_1 + r_2 n_2}$
빈도-기반	$\frac{R_1+R_2}{r_1 r_2 (W_1+W_2) \sum_{v_i \in D} f_1(v$	

ri : 단위시간에 도착하는 튜플의 수
kiop t: Ci(k)/k가 최대화 되는 k 값
Ci(k) : 최대 Si-probe 조인 튜플의 수

이 결과에서 보듯이 임의의 스트림에서 최대 부분 집합 문제를 해결하는 효과적인 로드-쉐딩 전략을 찾기가 어렵다.

2.2 연령-기반 모델 vs 빈도-기반 모델

최대-부분 집합 문제를 생각해보자. 한정된 용량의 메모리가 S1[W1]를 위해 주어져 있고, S1-probe 조인에 의해 생성되는 튜플과 같은 방법으로 S2[W2]에 대한 S2-probe 조인의 결과로 생성되는 튜플의 수를 최대화 시키는 문제를 고려해 보자. 더불어서 전체 조인 연산의 리콜을 최대화하기 위해서 한정된 전체 메모리를 어떻게 할당해야 할지를 보인다. 최소치를 가지는 연령 커브의 일반적인 경우, 최적화된 전략은 존재하지 않는다. 그러므로 좀 더 완전한 결과를 얻기 위해 S가 Si-probe 조인 튜플을 생성하는 가장 빠른 속도에 따라 우선순위를 부여한다. 메모리 부족으로 인하여 튜플을 버려야 한다면 우선순위가 가장 낮은 튜플이 버려진다. 그리디(greedy) 전략을 사용하여 최적의 해결책을 찾는 것이다.

빈도-기반 모델에서는 조인 연산을 위해 할당되는 총 메모리 량이 주어지면, M=M1+M2의 제약조건하에서 조인의 전체 리콜이 최대화 되는 M1과 M2를 추출해 낼 수 있다.

3.랜덤 샘플링

메모리에서 S1[W1]⋈A S2[W2] 조인 연산 결과의 랜덤 샘플링을 하는 문제를 생각해 보자. 임의의 스트림에서 균일한 랜덤 샘플링을 한다는 것은 아주 힘든 일이지만(3.1절), 연령-기반 모델과 빈도-기반 모델 양쪽에 다 적용되는 균일한 랜덤 샘플링 알고리즘을 보인다(3.2절). 마지막으로 3.3절에서는 응용분야의 특성상 직접적인 랜덤 샘플링은 불필요하지만, 조인 연산 결과의 집계를 측정하기 위해서 랜덤 샘플링이 요구되는 경우를 고려해 본다. 이런 경우에는 균일한 랜덤 샘플링 보다는 수행하기가 쉬운 통계적으로 약한 샘플링인 클러스터 샘플링을 고려해 보는데 때로는 이 샘플링에 의해 더 정확한 집계 측정이 이루어지기도 한다.

3.1 경도 결과 (Hardness Result)

임의의 스트림에서 윈도우 조인 결과에 대한 샘플링은 다음과 같은 부정적인 측면이 있다.

정리 3.1 주어진 메모리 용량이 전체 윈도우를 다 포함 하지 못할 때, 분수 값 0이상의 균일한 랜덤 샘플링이 항상 보장되지는 않는다.

메모리가 차 있어서 S1[W1]의 튜플 s를 버리는 경우에는 모든 S1-probe 조인에 의해 생성되는 튜플 속의 s가 이 샘플링에서는 필요하지 않다는 것이 확실해야 하지만, 임의의 스트림에 대해서는 이를 보장 할 수가 없다. 그러므로 최소한의 가능성을 위해서도 s를 버리지 말고 저장해 두어야 하는 것이다.

결론적으로, 제한된 메모리을 가진 임의의 스트림에서의 조인 연산 결과에 대한 균일한 랜덤 샘플링의 효율적인 절차는 기대하기 힘들다.

s_1 : Tuple arriving on S_1
$n_1(s_1)$: Number of S_1-probe join tuples that s_1 produces
p : Sampling fraction

DecideNextJoin(s_1):
1. pick $X \sim \mathcal{G}(p)$
2. $s_1.next = s_1.num + X$
3. **if** $(s_1.next > n_1(s_1))$
4. discard s_1

Join(s_1, s_2):
1. $s_1.num = s_1.num + 1$
2. **if** $(s_1.num = s_1.next)$
3. output $s_1 \bowtie_A s_2$
4. DecideNextJoin(s_1)

그림 3. UNIFORM 알고리즘
Figure 3. UNIFORM Algorithm

3.2 균일한 랜덤 샘플링

랜덤 샘플링에 있어서는 빈도-기반 모델과 연령-기반 모델을 동시에 고려 할 수 있다. 여기서는 동전 뒤집기와 같은 의미를 가지는 Bernoulli Sampling을 가정한다. p의 확률로 튜플의 집합으로부터 샘플링을 한다는 것은 집합 내의 모든 튜플이 다른 튜플들과 무관하게 p의 확률로 샘플링 된다는 것을 의미한다.

3.2.1 샘플링 알고리즘

한정된 메모리에서 슬라이딩 조인 연산에 대한 랜덤 샘플링 알고리즘은 그림 3.과 같다.

• 빈도-기반 모델

정리 3.2 빈도-기반 모델에서, S1[W1]의 예측 메모리 사용량은, q=1-p로 놓았을 때 다음과 같다.

$$r_1 W_1 \sum_{v \in \mathcal{D}} f_1(v) \left(1 - \frac{q(1-q^{r_2 W_1 f_2(v)})}{p r_2 W_1 f_2(v)}\right)$$

• 연령-기반 모델

정리 3.3 연령-기반 모델에서, S1[W1]의 예측 메모리 사용량은, q=1-p로 놓았을 때 다음과 같다.

$$S_1[W_1] \; is \; r_1 \sum_{i=1}^{n_1} p(1-p)^{n_1 - i} C^{-1}(i)$$

두 모델의 경우 S2[W2]의 사용량도 같은 형태로 나타내어진다. 이 둘을 더하면 S1[W1]⋈A S2[W2]의 총 메모리 사용량을 나타낼 수 있다.

3.3 클러스터 샘플링(Cluster Sampling)

클러스터 샘플링은 통계적으로는 취약하지만, 응용에서 요구하는 조인 연산이 직접 균일한 랜덤 샘플링을 요구하지 않고, 조인 결과의 전반적인 측정을 위하여 집계된 샘플이 이용될 때 적용 한다. 또한, 일반적으로 샘플링 하는 튜플 들이 그룹화 되거나 클러스터링이 가능하며, 또한 단일 튜플을 샘플링 하거나 클러스터 전체를 샘플링 하는 비용이 비슷할 경우 클러스터 샘플링을 적용할 수 있다. 임의로 몇 개의 클러스터를 선택하며 선택된 클러스터내의 모든 샘플들이 클러스터 샘플에 포함된다. 다양한 종류의 원소들이 동일한 확률로 클러스터에 포함되기 때문에 클러스터 샘플은 편중 되지 않는다. 그러나 균일한 샘플과는 달리 각 튜플 들이 서로 의존적 연관성을 가지고 있다.

다음의 표3.에서는 두 가지 방법의 클러스터 샘플링 기법을 비교하고 있다.

표3. EQ-Cluster 대 PPS-Cluster
table 3. EQ-Cluster vs. PPS-Cluster

클러스터	튜플 추가	duration	적합한 모델
EQ-CLUSTER	확률 p	만기까지	연령-기반 모델
PPS-CLUSTER	n1(s1)에 비례	만기까지	빈도-기반 모델

3.4 두 가지 샘플링의 비교

집계에서 얻어지는 수치의 에러를 최소화 하는 측면에서 두가지 샘플링 기법인 균일 샘플링과 클러스터 샘플링을 비교해 본다. 표4. 에서 볼 수 있듯이 대부분의 경우 클러스터 샘플링의 수행 결과가 좀 더 낫다는 것을 알 수 있다.

표4. UNIFORM 대 CLUSTER 샘플링
table 4. UNIFORM vs. CLUSTER Sampling

샘플링 기법	모델 인수의 정확성	클러스터간의 변형	클러스터 크기	
UNIFORM	빈약함	빈약함	적용불가	
CLUSTER	사용하기 좋음	변형이 적을수록 좋음	PPU	다른 사이즈
			EQ	동일 사이즈

위의 표에서 EQ 클러스터링은 일정한 확률로 클러스터에 입력 튜플을 더해주는 기법이며, PPU 클러스터링은 입력 스트림의 크기에 비례하는 확률로 클러스터에 튜플을 더해주는 기법이다.

4.다중 조인을 위한 메모리 할당

데이터 스트림 시스템에서 여러 개의 연속 질의가 실행되고 각각의 질의들은 슬라이딩 조인 연산을 수행한다고 가정했을 때, 다중 조인들 간의 메모리 할당의 문제가 고려되어야 한다. 모든 조인 연산의 우선순위가 동일하다면 메모리 할당의 목표는 어느 하나의 조인 연산에 치우치게 에러 값이 크지 않게 하는 것이다. 즉, 모든 조인 연산에서 최대 측정 에러를 최소화 하는 것이다. 여기서 같은 비율의 메모리를 사용한다고 하더라도 각기 다른 조인 연산은 측정의 정확도가 다를 수

있다는 것을 간과해서는 아니 된다. 그러므로 조인들 간에 같은 비율의 메모리를 할당해 주는 단순한 방법은 적절하지 못하다.

특정 메모리 할당에 있어서 q_i를 i번째 조인에서 얻어지는 리콜이라 하자. 가장 최적의 메모리 할당 구조는 $\min 1 \le i \le n$ q_i를 최대화 하는 것이다.

정리 4.1 최소 리콜을 최대화하기 위한 최적의 메모리 할당은 모든 조인에 대하여 균등한 리콜을 생성해 주는 할당이다.

5. 실험적 평가

5장에서는 각각의 기법에 대한 실험적 평가를 보인다. 먼저 연령-기반 모델에 두 개의 데이터 스트림과 서버를 함께 실행한다. 스트림 1과 서버는 동일 컴퓨터에서, 스트림 2는 WAN으로 연결되어 있는 각기 다른 지역의 노드들에서 실행된다. 각 스트림은 1초에 50개씩 튜플들을 생성하고 각 튜플은 소스의 지역 클럭에 의한 타임스탬프를 포함하고 있다. 튜플들은 무연결 UDP 채널을 통해 서버로 스트림 되며, 스트림 2에서 스트림 1의 튜플과 1분 차이의 타임스탬프를 가진 튜플들의 인과 상관관계를 찾기 위해 슬라이딩 윈도우 조인을 실행한다. 특히 WAN으로 연결된 스트림 2의 튜플들이 지연 도착하는 것을 고려, 스트림 1의 매치 될 튜플이 먼저 만료되지 않도록 스트림 1의 윈도우는 2분 크기로 설정한다. 스트림 2의 조인닝 튜플들은 언제나 스트림 1의 튜플들 보다 늦게 도착 하므로 스트림 2의 윈도우는 저장할 필요가 없다.

그림 4.는 63초에 가장 많은 조인 튜플을 생성하게 되는데 이는 조인 프레디키트의 실행이 이때 이루어짐을 나타낸다. 그림 5.는 다양한 로드-쉐딩에 의한 리콜의 결과이다. 실험에 적용된 각 기법별 튜플의 메모리 저장 기준은 표 5.와 같다. 빈도-기반과 같은 모델인 RECENT는 메모리 할당량이 40% 이하일 때 까지는 리콜이 0이다. 그림 6.은 다양한 모양의 연령 커브에 따라 리콜이 얼마나 발생하는지를 보여준다. 메모리가 충분하지 못한 대부분의 경우, 연령-기반 기법이 빈도-기반 기법보다 최소의 리콜을 최대화함을 보여준다. 또한 다중 조인이 있는 경우 단순히 메모리 할당량을 증가해주는 것만으로는 충분치 못함을 보여준다. 그림5.와 그림 6.에서 볼 수 있듯이 현실적인 환경 요소들이 연령-기반 기법과 더 잘 부합됨을 고려할 때, 연령-기반 모델도 기존의 단순한 기법보다는 AGE기법이 더 좋은 결과를 가져다준다는 실험 결과인 것이다.

표5. 각 기법별 튜플의 메모리 저장 기준
Table 5.Variant memory-retained tuples

AGE	주어진 타임 유닛동안 메모리에 저장 후 삭제. 빈 공간 없으면 도착 무시.
until-EXPIRY	메모리 여유 있으면 저장, 만료 때까지 저장 후 삭제. 빈 공간 없으면 도착 무시.
RECENT	새 튜플 도착 시 윈도우내의 가장 오래된 튜플이 삭제. (빈도-기반과 유사)
Theoretical-AGE	AGE기법에 따른 이론적인 리콜을 표현

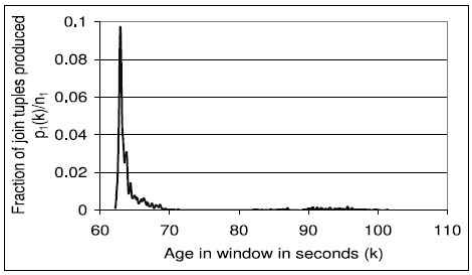

그림 4. WAN 시험의 연령 커브
Figure 4. Age curve for WAN experiments

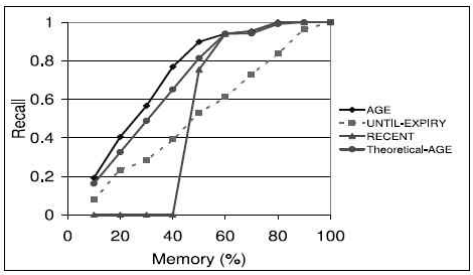

그림 5. WAN 실험에 의해 얻어진 리콜
Figure 5. Recall obtained on WAN experiments

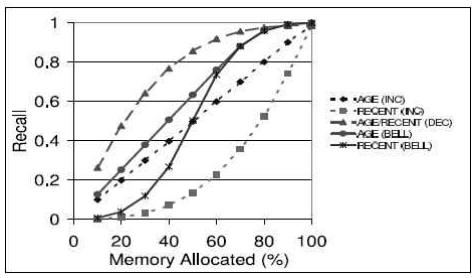

그림 6. 합성된 연령-기반 데이터에서 얻은 리콜
Figure 6. Recall obtained on synthetic age-based data

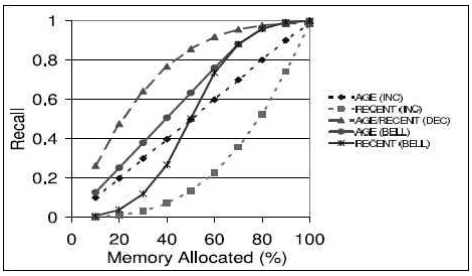

그림7. 빈도-기반 모델들의 집계 에러 vs 할당된 메모리
Figure 7. Aggregated error vs. memory allocated, IC frequency-based model

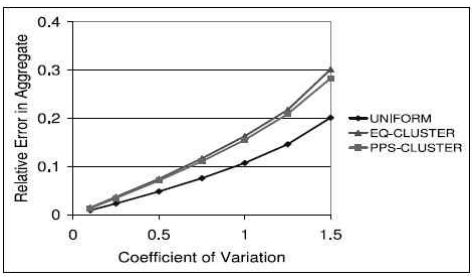

그림 8. 빈도-기반 모델들의 집계 에러 vs. 밀도 변형, 메모리 = 10%
Figure 8. Aggregation error vs. population variance, UC frequency-based model ,memeory=10%

그림7.에서는 PPS 클러스터가 EQ 클러스터 기법보다 우월함을 보여준다. 심지어 EQ 클러스터에서는 몇 개의 비교적 큰 클러스터를 놓치고 있다. 메모리 할당량이 10% 미만일 때에는 균일 클러스터의 에러가 가장 적다.

그림 8.에서는 메모리 할당량을 10%로 고정 시키고 클러스터내의 밀도에 변화를 주어 다양한 샘플링 기법에 따른 집계 에러의 변화를 보여준다. 밀도가 높아져 클러스터안의 모든 튜플 들이 같은 값을 가지게 되면 균일 클러스터링 기법에 비하여 클러스터 샘플링 기법의 에러가 증가함을 보여준다. 그러나 메모리 할당 비율이 높아지게 되면 점차적으로 클러스터 샘플링의 에러가 줄어드는 것은 그림 7.에서와 같다.

위의 실험 결과들을 정리하면 다음과 같다.

1. 빈도-기반 모델보다 연령-기반 모델의 사용이 사용 환경에 더 부합되는 현실에서는 연령-기반 기법 중에서도 AGE 알고리즘이 기존의 단순한 연령-기반 기법에 비해서 최소한의 리콜을 최대화 한다.
2. 메모리 할당량이 제한된 환경에서 빈도-기반 모델을 사용하는 경우, UNIFORM과 PPS-CLUSTER 샘플링 기법은 조인 연산 결과에 대해 윈도우를 사용한 집적 계산을 할 때 에러가 매우 낮다. 그러므로 위의 두 가지 기법이 다 유용하게 사용 될 수 있으며 조인 연산이 이루어지는 환경 요소(메모리 할당량, 클러스터의 밀도)에 따라 선택하여 사용하면 된다.

IV. 결 론

이 논문에서는 메모리 사용이 한정되어 있을 때 이러한 환경에서 이루어지는 슬라이딩 윈도우 조인 연산에 대해 논하였다.

기존의 빈도-기반 모델과 보다 더 효율적으로 최대-부분 집합 문제를 처리할 수 있는 연령-기반 모델을 사용하는 기법을 비교 하였고, 제한 된 메모리를 사용할 때 조인 연산 결과에서 랜덤 샘플을 추출하는 문제를 다루었다. 실험을 통하여 메모리 할당 비율에 따라 가장 적합한 샘플링 알고리즘을 선택하여 사용하는 방안도 제시하였다.

앞으로의 연구 방향은 이 논문에서 제안된 근사치 측정 기술, 조인 연산을 위한 메모리 할당이 제한 되어지는 환경 내에서도 근사치가 아닌 정확한 결과를 계산해 낼 수 있도록 확장하는 것이다. 현재는 로드-쉐딩 대신에 디스크에 선택된 데이터를 저장하는데, 이러한 환경에서 빈도-기반 모델이나 연령-기반 모델을 사용하여 디스크 I/O를 최소화 하는 알고리즘을 찾을 수 있을 것이다. 메모리 할당 전략 역시 좀 더 다양한 질의와 플랜 연산자를 다루기 위해 일반화 될 수 있을 것이다. 마지막으로 현재까지는 정적 데이터를 위한 알고리즘들이 스트림 데이터에 사용 되어 왔지만, 휘발성 데이터들에도 적용 될 수 있는 알고리즘이 개발되어야 할 것이다.

참고문헌

[1] A. Das, J. Gehrke, and M. Riedewald. “Approximate join processing over data streams”, In Proc. of the 2003 ACM SIGMOD Intl. Conf. on Management of Data, June 2003.

[2] J. Kang, J. F. Naughton, and S. Viglas. “Evaluating window joins over unbounded streams”, In Proc. of the 2003 Intl. Conf. on Data Engineering, March 2003.

[3] B. Babcock, S. Babu, M. Datar, R. Motwani, and

J.Widom. “Models and issues in data stream systems”, In Proc. of the 2002 ACM Symp. on Principles of Database Systems, pp. 1-16, June 2002.

[4] A. Dobra, M. Garofalakis, J. Gehrke, and R. Rastogi. "Processing complex aggregate queries over data streams". In Proc. of the 2002 ACM SIGMOD Intl. Conf. on Management of Data, pp. 61-72, 2002.

[5] T. Urhan and M.J. Franklin. Xjoin, "A reactively-scheduled pipelined join operato"r. IEEE Data Engineering Bulletin, 23(2):pp.27-33, June 2000.

[6] N. Tatbul, U. Cetintemel, S. Zdonik, M. Cherniack, and M. Stonebraker. "Load-shedding in a data stream manage"r. In Proc. of the 2003 Intl. Conf. on Very Large Data Bases, September 2003.

[7] B. Babcock, M. Datar, and R. Motwani. "Load-shedding for aggregation queries over data streams". In Proc. of the 2004 Intl. Conf. on Data Engineering, 2004.

[8] N. Alon, P. Gibbons, Y. Matias, and M. Szegedy. "Tracking join and self-join sizes in limited storage". In Proc. of the 1999 ACM Symp. on Principles of Database Systems, pp. 10-20, 1999.

[9] B. Babcock, M. Datar, and R. Motwani. "Sampling from a moving window over streaming data". In Proc. of the 2002 Annual ACMSIAM Symp. on Discrete Algorithms, pp. 633-634, 2002.

[10] S. Chaudhuri, R. Motwani, and V.R. Narasayya. "On random sampling over joins". In Proc. of the 1999 ACM SIGMOD Intl. Conf. on Management of Data, pp. 263-274, June 1999.

[11] W. G. Cochran. "Sampling Technique"s. John Wiley & Sons, 1977.

[12] M. Datar, A. Gionis, P. Indyk, and R. Motwani. "Maintaining stream statistics over sliding windows". In Proc. of the 2002 Annual ACMSIAM Symp. on Discrete Algorithms, pp. 635-644, 2002.

[13] A. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. "Muthukrishnan, and M. Strauss. Fast, small-space algorithms for approximate histogram maintenance". In Proc. of the 2002 Annual ACM Symp. on Theory of Computing, 2002.

[14] L. Golab and M. Ozsu. "Issues in data stream managemen"t. SIGMOD Record, 32(2):pp.5-14, June 2003.

[15] S. Guha, N. Koudas, and K. Shim. Data-streams and histograms. In Proc. of the 2001 Annual ACM Symp. on Theory of Computing, pp. 471-475, 2001.

[16] S. Krishnamurthy et al. "TelegraphCQ: An Architectural Status Repor"t. IEEE Data Engineering Bulletin, 26(1):pp. 11-18, March 2003.

[17] R. Motwani and P. Raghavan. "Randomized Algorithms". Cambridge University Press, 1995.

[18] The STREAM Group. "STREAM: The Stanford Stream Data Manage"r. IEEE Data Engineering Bulletin, 26(1):pp. 19-26, March 2003.

[19] Hong Shen, Yu Zhang, “Improved Approximate Detection of Duplicates for Data Streams Over Sliding Windows”, Journal of computer science and technology, Volume 23, Number 6, pp.973-987 ISSN 1666-6046 , 2008.

[20] YoungHyoo Yang, "An Efficient Query Processing in Stream DBMS using Query Preprocessor", Journal of The Korea Society of Computer and Information, Vol. 13, No. 1, pp. 65-73, 2008.

[21] Dongeon Lee et al., " A Multi-dimensional Query Processing Scheme for Stream Data Using Range Query Indexing", Journal of The Korea Society of Computer and Information, Vol. 14, No. 2, pp. 69-77, 2009.

저 자 소 개

양 영 휴

1981 : 이화여자대학교 영어영문학과 문학사

1987 : Northeastern University School of Computer Science M.S.(Artificial Intelligence)

1992 : 서강대학교 컴퓨터공학과 박사 과정 수료 (Database)

현재 : 한양여자대학 정보경영과 교수

관심분야 : 데이터베이스, 소프트웨어 공학, 운영체제, Telematics

Email: yhdh@chol.com