데이터 스트림 상에서 다중 연속 질의 처리를 위한 속성기반 접근 기법

이 현 호[†] · 이 원 석[††]

요 약

데이터 스트림은 빠르게 연속적으로 발생하는 무제한의 데이터 튜플의 집합이다. 이러한 데이터 스트림에 대한 질의 처리 또한 연속적이고 신속해야 하며 엄격한 시공간적 제약이 요구된다. 대부분의 데이터 스트림 관리시스템(DSMS)에서는 시공간적 제약사항을 효과적으로 지키기 위해서 등록된 연속 질의들의 선택 조건(selection predicate)들을 그룹화하거나 색인처리 한다. 본 논문에서는 연속 질의들의 선택 조건들을 속성별로 그룹화한 새로운 구조체인 속성 선택체(Attribute Selection Construct)를 제안한다. 속성 선택체에는 해당 속성이 특정 질의조건에 사용되는지 여부, 부분적으로 미리 계산된 질의결과 정보, 그리고 해당 속성의 선택률 통계 등 효율적인 질의 처리를 위한 유용한 정보들이 포함된다. 또한, 대상 질의집합을 구현한 속성 선택체들 간의 처리 순서는 전체적인 질의성능에 많은 영향을 미칠 수 있기 때문에 효과적으로 속성 선택체 처리 순서를 결정할 수 있는 전략도 함께 제안된다. 마지막으로, 기존의 방법들이 포함된 다양한 실험을 통하여 제안된 방법론의 성능을 여러 각도에서 비교 검증한다.

키워드 : 데이터 스트림, 다중 연속 질의, 속성 선택체, 매칭 알고리즘, 질의 최적화

Attribute-based Approach for Multiple Continuous Queries over Data Streams

Hyunho Lee[†] · Wonsuk Lee[††]

ABSTRACT

A data stream is a massive unbounded sequence of data elements continuously generated at a rapid rate. Query processing for such a data stream should also be continuous and rapid, which requires strict time and space constraints. In most DSMS (Data Stream Management System), the selection predicates of continuous queries are grouped or indexed to guarantee these constraints. This paper proposes a new scheme called an ASC (Attribute Selection Construct) that collectively evaluates selection predicates containing the same attribute in multiple continuous queries. An ASC contains valuable information, such as attribute usage status, partially pre calculated matching results and selectivity statistics for its multiple selection predicates. The processing order of those ASC's that are corresponding to the attributes of a base data stream can significantly influence the overall performance of multiple query evaluation. Consequently, a method of establishing an efficient evaluation order of multiple ASC's is also proposed. Finally, the performance of the proposed method is analyzed by a series of experiments to identify its various characteristics.

Key Words : Data Stream, Multiple Continuous Queries, Attribute Selection Construct, Matching Algorithm, Query Optimization

1. 서 론

데이터 스트림에 대한 연구는 인터넷 상의 고객행위 흐름, 통화기록, 멀티미디어데이터 흐름 그리고, 상품유통 흐름 등 데이터 스트림으로 정의될 수 있는 거대한 데이터집합을 이용하는 새로운 경향의 어플리케이션들에 의해 요구되기 시작했다. 데이터 스트림은 빠르게 연속적으로 발생하는 무제한의 데이터 튜플의 집합으로 정의될 수 있는데, 데이터 스트림의 특성으로 인해 데이터 스트림 관리시스템(DSMS)에서 사용되는 질의는 필요 시 수행되는 개념이 아니라 연속적으로 계속 수행되는 개념으로 사용된다. 이러한 질의를 연속 질의(continuous query)라고 한다. DSMS에서의 연속 질의는 미리 등록되고 대상 데이터 스트림의 새로운 튜플이 도착될 때 마다 반복적으로 수행된다. 그러므로, 연속 질의는 실시간으로 처리되어야 하며 엄격한 시공간적 제약을 수반한다. 이러한 점에서, 다중 연속 질의의 효율적인 처리를 위해서는 등록된 질의들을 개별적으로 처리하기 보다 질의들의 공통 선택 조건(selection predicate)들의 공유를 통하

※ 본 과제(결과물)는 교육인적자원부, 산업자원부, 노동부의 출연금 및 보조금으로 수행한 최우수실험실지원사업의 연구결과입니다.
† 정 회 원 : 연세대학교 컴퓨터과학과 박사과정 (교신저자)
†† 종신회원 : 연세대학교 컴퓨터과학과 교수
논문접수 : 2006년 12월 14일, 심사완료 : 2007년 5월 8일

여 집단적으로 처리하는 것이 유리하다. 이러한 방법은 이미 대부분의 DSMS에서 적용하고 있다. 본 논문에서는 연속 질의의 집단적인 처리를 위해서 속성 선택체(Attribute Selection Construct)라는 새로운 구조를 제안한다. 주어진 연속 질의 집합에서 적어도 하나 이상의 선택 조건에 사용되는 데이터 스트림의 속성을 참여 속성(participant attribute 또는 p-attribute)라고 정의한다. 속성 선택체는 참여 속성들 각각에 대해서 하나씩 만들어진다. 속성 선택체는 주어진 다중 연속 질의들의 선택 조건들 중에서 해당 참여 속성에 대한 선택 조건들을 통합적으로 처리하는 구조체로서 여러 개의 연속 질의들 사이에 동일한 속성에 대한 선택 조건들이 해당 속성 선택체에 의해 공유된다.

속성 선택체에는 해당 참여 속성이 주어진 연속 질의 집합에서 어느 연속 질의의 선택 조건에 사용되었는지를 나타내는 정보가 저장된다. 또한, 속성이 가질 수 있는 값의 범위를 속성의 도메인(domain)이라고 할 때, 참여 속성의 도메인은 선택 조건들에 의해 배타적 영역(region)으로 분할되고 각 영역별로 스트림 튜플을 만족시키는지 여부를 나타내는 정보가 포함된다. 이러한 정보는 데이터 스트림의 내용이 아니라 대상 질의들의 특성에 의해 얻어지기 때문에 질의수행 전에(질의등록 시에) 미리 만들어 질 수 있다. 이는 시간적 제약이 많은 스트림 환경에서의 질의 처리에 관한 실시간 부하를 줄이는데 기여한다. 제안된 방법의 또다른 기여점은 속성 선택체가 CACQ[7]나 PSoup[9] 등 기존의 방법들과는 달리 비동등 비교 연산(non-equal comparison operation)의 수행속도와 동등 비교 연산(equal comparison operation)의 수행속도가 거의 차이가 없다는 점이다. 이것은 처리대상 스트림 튜플의 속성값은 반드시 해당 속성 선택체의 나뉘어진 영역들 중 하나에만 속하게 되고, 각 영역들은 선택 조건의 비교 연산의 유형에 상관없이 만족하는 질의들의 정보를 동일한 방식으로 가지고 있기 때문이다.

주어진 연속 질의들에 대한 하나 이상의 속성 선택체들의 처리 순서는 전체적인 질의성능에 많은 영향을 미칠 수 있다. 왜냐하면, 속성 선택체에 대응되는 참여 속성의 선택률은 서로 간에 차이가 있기 때문이다. 질의 최적화의 본질인 불필요한 연산수행을 막기 위해서 취해지는 일반적인 질의 최적화 전략은 선택 연산(selection operations)을 가능한 빨리 수행하며, 여러 개의 선택 조건들 중에는 선택률이 가장 낮은 선택 조건을 먼저 적용하는 것이다. 본 논문에서는 연속 질의들의 처리시간을 줄이거나 어떤 질의에도 만족되지 않는 스트림 튜플을 가능한 한 빨리 제거하기 위해서 적응적으로 속성 선택체들의 처리 순서를 결정하는 속성 선택체 정렬 전략을 제안한다. 이러한 전략은 만족하지 않는 튜플이나 질의들에 의해 수행되는 불필요한 연산들을 차단함으로써 질의수행비용을 줄이는데 기여한다.

본 논문의 목적은 다음과 같이 요약될 수 있다.

주어진 데이터 스트림 D와 연속 질의 집합 Q에 대해서, 질의집합 Q에 속한 질의들의 공통 선택 조건들을 공유하고 Q에 속한 질의들을 연속적이면서 정시적인 형태로 동시처리하기 위한 효과적인 데이터 구조와 매칭 알고리즘을 제안하고 이에 대한 적응적 최적화 방안을 제시하는 것이다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 본 논문과 관련된 기존 연구들을 살펴본다. 3장에서는 참여 속성에 대한 속성 선택체가 어떻게 만들어지고 사용되는지 자세히 살펴본다. 4장에서는 질의 최적화를 위한 속성 선택체 정렬 전략을 소개한다. 5장에서는 제안된 방법의 성능을 일련의 실험들을 통해 분석 검증하고, 6장에서 결론을 맺는다.

2. 관련연구

데이터 스트림에 대한 질의 처리는 최근 큰 관심을 받고 있고, NiagaraCQ[4], CACQ[7], PSoup[9] 그리고 STREAM[6]을 비롯한 많은 연구들이 제안되었다. 발표된 연구들의 대부분은 다중 연속 질의 처리를 위한 그룹기반의 처리기법을 사용한다[4, 7, 9]. *NiagaraCQ*[4]는 질의들 간의 유사구조를 공유한다는 원칙 하에 웹 기반의 다중 연속 질의를 그룹화함으로써 인터넷 환경에서의 다중 연속 질의 처리를 가능하게 한다. 이 시스템은 XML-QL과 비슷한 질의언어를 사용하고, 질의 최적기(query optimizer)가 연속 질의들의 포맷을 분석하여 유사구조를 그룹화한다. 연속 질의들의 선택 연산이 스트림 튜플과 그룹상수 테이블(group constant table) 간의 조인을 통하여 수행되고 질의의 중간결과가 저장되어야 하기 때문에, 대상 질의들이 그룹화되어 공통연산을 공유한다고 하더라도 질의들의 특성에 따라 연산비용과 I/O비용이 크게 증가될 수 있다. 질의 수행을 스트림 데이터와 질의 간의 조인의 관점에서 바라본 또다른 연구로 [18]가 있다. 이 연구에서는 연속 질의의 문제를 다차원 공간 조인(multi-dimensional spatial join)의 문제로 변환하였다. 즉, 스트림의 데이터요소(data element)와 질의 각각에 대하여 색인(index)를 구축하여 이를 다차원 공간에서의 영역들로 나타내고, 이 영역들 간의 공유 영역(overlapping region)을 공간 조인 연산을 이용하여 구함으로써 질의를 수행한다. 이 방법은 쿼리주도의 질의(ad-hoc query)와 데이터 주도의 질의(continuous query)에 동일한 관점으로 적용할 수 있는 장점이 있으나, 스트림 튜플들을 일정한 정도로 모은 다음 한꺼번에 처리하는 것이 효율적이므로 스트림 튜플의 신속한 필터링이나 각 튜플에 대한 빠른 질의응답시간을 요하는 곳에는 적절하지 않을 수 있다.

CACQ[7]는 다중의 선택 조건들을 동시에 처리하기 위해서 그룹필터(Grouped filter)라는 기법을 사용한다. 이 기법은 속성기반 그룹화 기법이라는 측면에서 본 논문의 방식과 유사하다. 그룹필터는 다중 질의의 공통 선택 조건들의 연산 결과를 공유함으로써 연산비용을 줄이는 조건 색인 연산자(predicate indexing operator)로 정의할 수 있다. 조건 색인(predicate index)는 선택 조건에 사용된 각 속성마다 만들어지며, 비교 연산자에 따라 해쉬 테이블(hash table), 비교상위 트리(greater-than tree) 그리고 비교하위 트리(less-than tree) 등 다양한 데이터 구조로 표현된다. 그러나, 비동등 연산의 경우, IBS(Interval Binary Search)트리 형태로 표현된

조건 색인(predicate index)의 특정범위를 검색해야 하기 때문에 질의의 성능가 우려된다. Telegraph[3] 프로젝트에서 제안된 PSoup[9]은 질의의 형태를 애드혹(ad-hoc), 연속(continuous) 그리고 혼합(hybrid)의 세가지 유형으로 나누고, 새로운 질의가 과거 데이터에도 적용되는 적응적 기법을 제시하였다. 또한, 등록된 다중 연속 질의들을 색인하기 위해서 일련의 홍흑 트리(red-black tree)군을 활용한다. 홍흑 트리는 또한 본 논문의 속성 선택체(ASC)처럼 선택 조건에 사용된 각 속성마다 만들어진다. PSoup의 핵심은 선택 작업들의 처리를 다수의 연속 질의들과 스트리밍되어 발생되는 데이터들 간의 조인작업으로 수행함으로써 스트림 데이터와 연속 질의 모두에 대해 적응적 관리가 가능하다는 점이다. 그러나, 본 논문과의 주요 비교관점인 선택 연산 처리의 효율성 측면에서 PSoup은 CACQ와 마찬가지로 비동등 연산에서 홍흑 트리의 순차적인 검색이 필요하기 때문에 효율성이 떨어진다는 약점을 가지고 있다.

실시간 부하를 줄 수 있다는 점에도 불구하고 다양한 적응적 최적화 전략이 제안되었다[6, 14, 17]. Eddy[14]는 데이터 스트림의 대기 큐(waiting queue)로부터 새로운 튜플을 받아서 라우팅 정책에 따라 선택모듈(selection module) 중 하나에 보냄으로써 질의를 처리한다. 라우팅 정책에는 랜덤 라우팅(random routing)과 티켓 라우팅(ticket routing)이 있다. 랜덤 라우팅은 새로운 튜플을 임의로 라우팅 하는 방식이다. 반면, 티켓 라우팅에서는 각각의 선택모듈이 자신의 티켓수를 가지며, 새로운 튜플이 특정 선택모듈에서 제거되면 그 선택모듈의 티켓 수를 증가시키고 그렇지 않으면 감소시킨다. 결과적으로 많은 티켓을 가진 선택모듈이 많은 튜플을 제거했다는 뜻이므로 Eddy는 그 선택모듈로 새로운 튜플을 먼저 보내게 된다. 티켓 라우팅의 단점은 티켓 라우팅 방식의 스케줄링은 오직 티켓의 수를 근거로 실시간으로 이루어지기 때문에, 선택 조건의 선택률 등 연속 질의 처리에 대한 통계들을 질의 최적화에 이용할 수 없다는 점이다. 또한, 질의 최적화를 위한 정보를 실시간에 계속 유지해야 하고 빈번한 스케줄 변경이 발생할 수 있다는 점이 질의성능의 부담으로 작용할 수 있다. STREAM[6]에서는 필터링 순서를 정하기 위해서 조건부 필터 선택률(conditional filter selectivity)에 기초한 A-Greedy 알고리즘을 사용한다. A-Greedy의 특징은 실시간으로 모니터 된 선택률 통계에 따라 적응적으로 필터링 순서를 재정렬한다는 것이다. 그러나, 이 알고리즘은 단일 질의 최적화에만 적용된다는 면에서 본 논문의 방법과는 근본적으로 다르다.

[17]에서는 다중 연속 질의에서의 선택 조건(필터)들의 처리 순서를 최적화하기 위해 고비용 필터(expensive filters)에 대한 공유 수행 전략(shared execution strategy)를 제안하였다. 제안된 전략은 비용 모델(cost model)에 근거한 필터의 개념적 공유 모델과 점진적이고 적응적인 필터의 순차 수행을 그 특성으로 한다. 개념적 공유 모델이라 함은 여러 질의에 존재하는 동일한 필터에 대해서 한 질의에서 수행된 필터의 수행 결과를 다른 질의들에서 재사용 함을 뜻하며, 적응적 순차 수행을 위해 처리되지 않은 필터들 중에 다음 처리될 필터를 실시간 비용분석을 통해 동적으로 결정한다. 이 논문은 필터의 공유 수행에 대한 개념적인 원칙을 질의 수행 알고리즘에 반영하였으나 본 논문의 속성 선택체와 같이 필터연산의 공유를 위한 구체적인 데이터 구조를 제안하지는 않았다.

3. 속성 선택체와 질의 처리 알고리즘

3.1 속성 선택체 (Attribute Selection Construct)

DSMS에 등록된 연속 질의의 선택 표현식(selection expression)은 하나 이상의 선택 조건(selection predicate)들의 논리곱으로 표현될 수 있으며, 대상 연속 질의들의 선택 표현식 모두를 만족시키지 못한 튜플들은 제거된다. n개의 속성을 지닌 데이터 스트림 $D(A_1, A_2, \cdots, A_n)$와 k개의 연속 질의들을 가진 질의집합 $Q=\{q_1, q_2, \cdots, q_k\}$가 주어졌을 때, Q에 대한 참여 속성들의 집합을 $A_p(Q)$라 하면 $A_p(Q)$는 다음을 만족한다.

$$A_p(Q) \subseteq \{A_1, A_2, \cdots, A_n\}$$

참여 속성은 다중 연속 질의들 사이에 공유될 수 있으므로 대상 질의들에 있는 선택 조건들을 참여 속성 단위로 그룹화하고 생성된 그룹을 차례로 처리한다면, 스트림의 현재 튜플을 만족시키지 못하는 질의들을 보다 빨리 종료시켜 전체적인 질의 최적화에 기여할 수 있다. 질의집합 Q에 대한 참여 속성 $A_i \in A_p(Q)$의 선택 조건들의 집합을 $P(A_i)$라 하면 $P(A_i)$는 다음과 같이 정의된다.

$$P(A_i) = \{p_1(A_i), p_2(A_i), \cdots, p_r(A_i) \mid p_j(A_i) = A_i \Theta C_l \ (1 \le j \le r, l \le r)\}$$ where C_l is a *comparison constant* and Θ is a *comparison operator*(=, ≠, >, <, *etc.*)

정리 1. 참여 속성 $A_i \in A_p(Q)$의 도메인을 $dom(A_i)$라 하고 A_i의 대한 선택 조건집합 $P(A_i)$에서 m개의 비교상수 *(comparison constant)*가 사용된다면, 도메인 $dom(A_i)$는 많아야 $(2m+1)$개의 상호배타적인 영역들로 분할된다. 먼저, 비교상수 값 자체로 자신의 영역을 구성한다. 이를 상수영역이라 한다. $dom(A_i)$의 나머지 영역은 m개의 상수영역에 의해 많아야 $(m+1)$개의 배타적 영역으로 분할된다. 이를 구간영역이라 한다.

질의 $q \in Q$가 선택 조건집합 $P(A_i)$에 속하는 선택 조건을 포함하고 있다면, 속성 A_i에 대한 질의 q의 매칭결과는 포함된 선택 조건의 비교 연산 형태에 따라 $dom(A_i)$의 영역별로 질의수행 이전에 결정될 수 있다. 이러한 매칭 정보를 효과적으로 저장하기 위해서 속성 선택체(Attribute Selection Construct)이라는 새로운 구조체를 제안한다. 속성 선택체는 주어진 질의집합의 개별 참여 속성 각각에 대하여 만들어지며 질의 처리과정에서 필요한 정보를 가지고 있다. 속성 선택체는 다음과 같이 정의된다.

정의 1. (속성 선택체) 데이터 스트림 $D(A_1, A_2, \cdots, A_n)$와

연속 질의 집합 $Q=\{q_1,q_2,\cdots,q_k\}$가 주어졌을 때, m개의 비교 상수 값들을 가진 Q의 참여 속성 A_i에 대한 속성 선택체 $ASC(A_i)$는 다음과 같이 구성된다.

- 질의 활용 비트맵(Query Usage Bitmap) : $ASC(A_i).qub$로 표기하며, qub의 j 번째 비트 qub_j는 참여 속성 A_i가 질의 q_j의 선택 표현식에 사용되었는지 여부를 나타낸다. 사용되었다면 $qub_j = 1$, 그렇지 않다면 $qub_j = 0$으로 설정한다.
- 영역 배열(Comparison Region Array) : $ASC(A_i).cra[1..2m+1]$로 표기하며, 참여 속성 A_i의 선택 조건들의 비교상수들에 의해 분할된 도메인 $dom(A_i)$내의 영역들의 배열이다. (정리 1)에 의하여, m개의 비교상수에 의해 분할된 영역의 총 개수는 $2m+1$이다. $ASC(A_i).cra$의 r 번째 영역 $ASC(A_i).cra[r](1\le r\le 2m+1)$은 다음과 같은 두 개의 필드(field)를 가진다.
  - 영역 식별자(Region Identifier) : $ASC(A_i).cra[r].rif$로 표기하며, 영역 $ASC(A_i).cra[r]$이 상수영역인지 구간영역인지를 구분한다. 만약, $ASC(A_i).cra[r]$가 상수 값 $c \in dom(A_i)$를 가지는 상수영역이라면 $ASC(A_i).cra[r].rif = c$이다. 그렇지 않고 구간영역이라면, $ASC(A_i).cra[r].rif = null$이다.
  - 질의 결과 비트맵(Query Result Bitmap) : $ASC(A_i).cra[r].qrb_{1..k}$로 표기하며, 영역 $ASC(A_i).cra[r]$에서의 미리 계산된 질의 매칭 결과를 저장한다. 영역비트맵 $ASC(A_i).cra[r].qrb$의 j 번째 비트 qrb_j는 질의 q_j의 속성 A_i에 대한 선택 조건(들)이 영역 $ASC(A_i).cra[r]$에서 질의 q_j을 만족하는지 여부를 나타낸다. 만족하거나 $ASC(A_i).qub_j = 0$이면 $ASC(A_i).cra[r].qrb_j = 1$이다. 그렇지 않다면, $ASC(A_i).cra[r].qrb_j = 0$이다. ($ASC(A_i).qub_j = 0$는 질의 q_j의 선택 표현식에 참여 속성 A_i가 사용되지 않는 경우이므로 q_j는 거짓이 되지 않는다. 이는 $ASC(A_i).cra[r].qrb_j = 1$과 같다.) □

질의 활용 비트맵은 4.1절에 소개될 개념인 질의집합 Q에 대한 최소 속성커버 집합(MACS)를 찾는데 활용된다. 또한, 질의 결과 비트맵을 통하여 데이터 스트림의 현재 튜플의 해당 속성값에 따라 각 질의의 그 속성에 대한 매칭 결과를 바로 확인할 수 있다.

(그림 1)은 질의집합 $Q = \{q_1, q_2, q_3, q_4\}$에 대한 참여 속성 A_i의 속성 선택체의 예를 보여준다. 질의 활용 비트맵 $ASC(A_i).qub = 1011$은 참여 속성 A_i가 질의 q_1, q_3, q_4의 선택 표현식에 사용되고 있음을 나타낸다. 또한, 영역배열에서 예를 들어 $ASC(A_i).cra[3].qrb = 1110$은 참여 속성 A_i 값이 세 번째 영역인 구간 $(12, 20)$에 존재하면, 질의 q_1, q_2, q_3에서 A_i에 대하여 참임을 나타낸다. 질의 q_1과 q_3는 A_i를 포함한 각각의 선택 조건(들)에 대해서 참이다. 질의 q_2의 경우는 선택 표현식에 속성 A_i가 참여하지 않기 때문에($ASC(A_i).qub_2 = 0$) A_i 값이 어떠한 영역에 속해도 q_2는 A_i에 대해 거짓이 될 수 없다. 그러므로, 영역 $ASC(A_i).cra[3]$ 뿐만 아니라 $ASC(A_i)$의 모든 영역에 걸쳐 비트 $qrb_2 = 1$이다. 반면, 질의 q_4는 $ASC(A_i).cra[3].qrb_4 = 0$이기 때문에 현재 튜플의 참여 속성 A_i 값이 구간 $(12, 15)$에 속할 경우 질의 q_4의 A_i를 포함한 선택 조건(들)이 거짓이 되므로 q_4는 종료된다.

참여 속성 A_i의 속성 선택체 $ASC(A_i)$를 만들고 유지하는 것은 실시간 질의 처리과정에는 거의 부하를 주지 않는다. 왜냐하면, $ASC(A_i)$의 구성요소인 질의 활용 비트맵이나 영역구조체 내의 영역 구별자 및 질의 결과 비트맵의 값은 데이터 스트림 D의 특성이 아니라 질의집합 Q의 특성에 따라 결정되므로 속성 선택체 $ASC(A_i)$의 모든 구성요소가 질의 처리 전에 만들어질 수 있기 때문이다.

3.2 속성 선택체를 이용한 매칭 알고리즘

본 논문에서의 질의 매칭은 참여 속성 각각에 대한 속성 선택체 단위로 수행된다. 수행대상 참여 속성에 대하여 먼저 데이터 스트림의 현재 튜플에 존재하는 해당 참여 속성의 값이 속성 선택체에 정의된 영역들 중 어디에 속하는가를 판단한다. 그리고, 소속 영역에서의 미리 계산된 대상 질의들의 매칭 결과에 따라 질의들의 성공여부를 판단하고 모든 질의들이 실패한 경우에는 현재 튜플 자체가 제거된다. 실패하지 않은 질의가 남아 있을 경우에는 수행될 다른 참여 속성에 대한 매칭 프로세스가 계속 진행된다. 이러한 방식으로 대상 질의집합에 대한 모든 참여 속성들의 매칭 프로세스를 끝낸 후, 현재 튜플에 대한 질의 처리는 최종적으로 살아남은 매칭에 성공한 질의들을 파악하고 종료된다.

(알고리즘 1)은 데이터 스트림 D에 대한 질의집합 Q의 매칭 프로세스를 기술한 것이다. 이 알고리즘에서는 매칭 과정에서 질의집합 Q의 질의중간결과를 저장하기 위해서 전역 비트맵 변수 GRB이 소개된다. 비트맵 변수 GRB의 모든 비트들은 모두 1로 초기화된다. 이는 질의집합 Q내의 모든 질의들이 대상 데이터 스트림 튜플을 만족한다는 가정에서부터 출발한다는 의미이다. 데이터 스트림 D의 튜플 t_D를 처리한 후, 비트맵 변수 GRB의 i번째 비트 r_i의 값이 여전히 1이면 질의 $q_i \in Q$는 튜플 t_D를 만족시킨 것으로 간주된다. (알고리즘 1)에서 보듯이, 매칭 프로세스는 질의집합

질의 활용 비트맵(qub)		*1011 ($q_1q_2q_3q_4$)*						
영역배열(cra)		*cra[1]*	*cra[2]*	*cra[3]*	*cra[4]*	*cra[5]*	*cra[6]*	*cra[7]*
	영역구별자(rif)		*12*		*20*		*35*	
	질의 결과 비트맵(qrb)	*1101*	*1101*	*1110*	*1100*	*0111*	*1111*	*1101*

(그림 1) 질의집합 $Q = \{q_1, q_2, q_3, q_4\}$의 참가속성 A_i에 대한 $ASC(A_i)$의 예

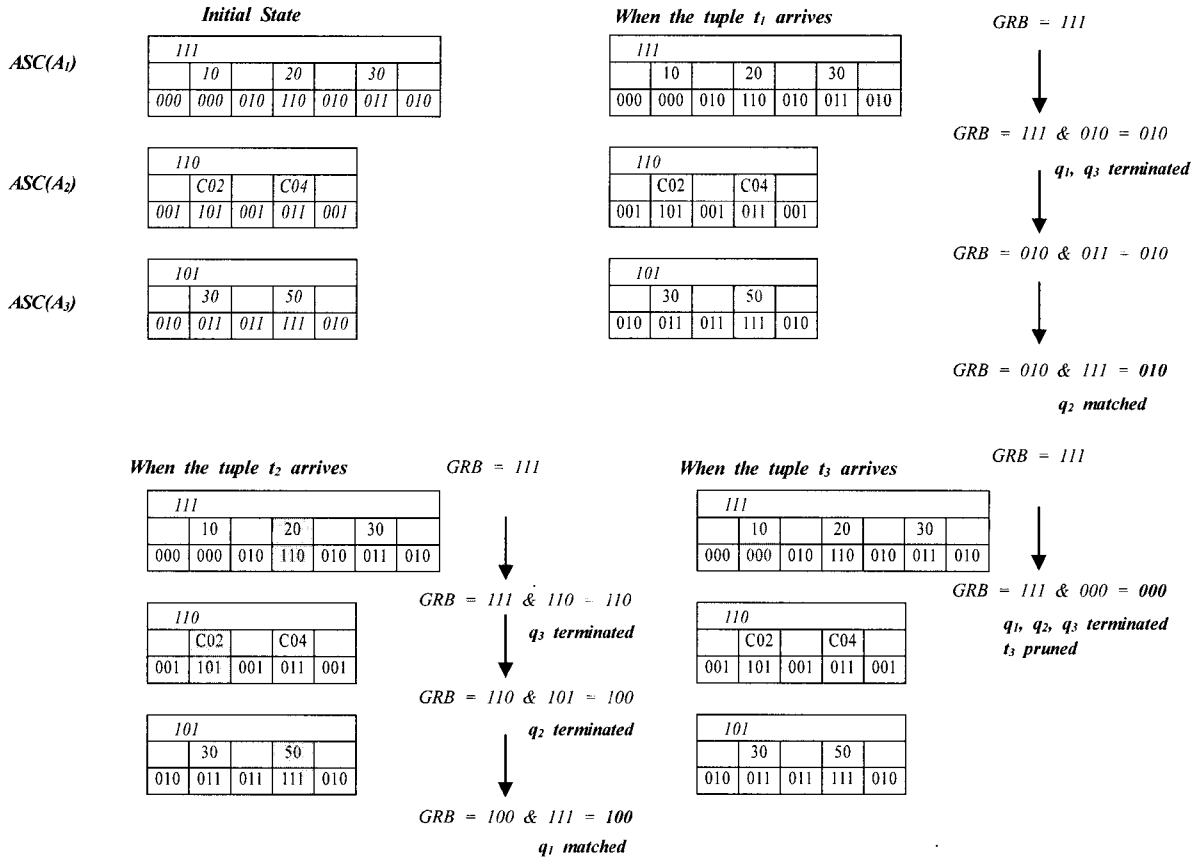
```
1  tD : the current tuple of a given data stream D.
2  V(tD, Ai) : the p-attribute Ai's value in the tuple tD.
3  GRB : the global result bitmap variable storing the intermediate results of the query set Q during the matching process.
4  find_region(ASC(Ai).cra, V(tD, Ai)) : returns the region (interval or constant) containing the value V(tD, Ai) in the list ASC(Ai).cra.
5  initializes GRB to the bitmap in which the bits are all 1 ;
6  for each tuple tD ∈ the stream D do
7    for each p-attribute Ai ∈ Ap(Q) do
8      ASC(Ai).cra[r] = find_region(ASC(Ai).cra, V(tD, Ai)) ;
9      if GRB = 0 then
10         tD is filtered out because tD does not match any queries in Q ;
11         exit for ; /* exit inner for-loop */
12     end if
13     GRB = GRB & ASC(Ai).cra[r].qrb ;
14   next Ai
15   output GRB ;
16 next tD
```

(알고리즘 1) 데이터 스트림 D와 질의집합 Q 간의 매칭 알고리즘

Given a query set Q = {q1, q2, q3} and a data stream D consisting of three attributes A1, A2, A3, and a global bitmap variable R

The predicates of q1, q2, q3 :
q1 : D.A1 = 20 AND D.A2 = 'C02' AND D.A3 = 50
q2 : D.A1 > 10 AND D.A2 = 'C04'
q3 : D.A1 = 30 AND D.A3 >= 30 AND D.A3 <= 50

The tuples t1, t2, t3 of the stream D :
t1 : {40, 'C04', 50}
t2 : {20, 'C02', 50}
t3 : {10, 'C04', 20}



(그림 2) 속성 선택체를 이용한 데이터 스트림에 대한 다중 연속 질의 수행 예

Q에 대한 각각의 참여 속성 $A_i \in A_p(Q)$에 대해 단지 세가지 기본 연산들로만 구성된 단순한 작업이다. 첫 번째 연산은 새로 발생하는 데이터 스트림 튜플 t_D에서 속성 A_i 값을 포함하는 영역을 영역배열 $ASC(A_i).cra$에서 찾는 것이다((알고리즘 1)의 8 번째 줄). 찾아진 영역을 $ASC(A_i).cra[r]$이라고 가정하면, 두 번째 연산은 비트맵 $ASC(A_i).cra[r].qrb$와

비트맵 변수 GRB을 *bitwise-AND* 방식으로 더한 결과로 비트맵 변수 GRB을 갱신하는 것이다((알고리즘 1)의 13번째 줄). 비트맵 변수 GRB의 모든 비트가 0이 되면 질의집합 Q 내의 어떠한 질의도 튜플 t_D를 만족하지 않는다는 뜻이므로, 튜플 t_D는 즉시 제거된다. ((알고리즘 1)의 9-12번째 줄)

(그림 2)는 (알고리즘 1)에 기반으로 데이터 스트림 D의 튜플 t_1, t_2 그리고 t_3에 대해 질의 집합 $Q = \{q_1, q_2, q_3\}$를 수행하는 예를 보여준다. 질의집합 Q의 참여 속성 A_1, A_2 그리고 A_3에 대해, 세 개의 속성 선택체 $ASC(A_1)$, $ASC(A_2)$ 그리고 $ASC(A_3)$가 각각 만들어진다. 새로운 튜플 t_k $(1 \le k \le 3)$가 도착할 때 마다, 각각의 속성 선택체 $ASC(A_i)$ $(1 \le i \le 3)$를 한번씩 방문하면서 (알고리즘 1)에 기술한 대로 튜플 t_k에서의 참여 속성 A_i의 값을 포함하는 영역에 따라 전역 비트맵 변수 GRB의 값을 변화시킨다. 세 개의 속성 선택체들을 방문하는 과정에서 GRB의 $j(1 \le j \le 3)$ 번째 비트가 0이 되면 질의 q_j는 즉시 종료된다. 튜플 t_1에 대한 질의 q_2 또는 튜플 t_2에 대한 질의 q_1의 경우처럼, 모든 속성 선택체들을 방문한 후에 GRB에서의 해당 비트가 여전히 1인 질의를 튜플 t_k를 만족하는 것으로 판단한다. 특히, 튜플 t_3의 경우는 단지 속성 선택체 $ASC(A_1)$만을 방문한 후에 질의 q_1, q_2 그리고 q_3를 모두 만족시키지 않는다는 사실을 알게 되므로 다른 속성 선택체를 수행하지 않고 튜플 t_3가 즉시 제거된다.

3.3 성능평가

데이터 스트림 $D(A_1, A_2, \cdots, A_n)$와 질의집합 $Q=(q_1, \cdots, q_k)$에 대하여, 질의집합 Q에 존재하는 참여 속성의 수를 n_p라 하고 각 참여 속성 $A_i \in A_p(Q)$에 존재하는 비교상수의 개수들의 평균을 m이라고 할 때, 데이터 스트림 D의 각 튜플에 대해 질의집합 Q를 처리하는데 필요한 공간과 시간은 (정리 2)과 (정리 3)와 같이 평가된다.

정리 2. 질의집합 Q에 존재하는 n_p개의 참여 속성들에 대한 속성 선택체들을 유지하는데 필요한 공간은 최대 $O(mkn_p)$이다.

(증명) (정리 1)에 의해 참여 속성 A_i에 대한 선택 조건집합 $P(A_i)$가 m개의 비교상수를 가질 때 최대 $(2m+1)$개의 영역이 존재한다. (정의 1)에 의하여 질의 활용 비트맵 $ASC(A_i).qub$를 위한 공간은 비트맵의 길이가 질의집합 Q 내의 질의의 수와 같기 때문에 k비트 필요하다. 영역배열 $ASC(A_i).cra$의 영역 식별자를 위한 공간은 $2mC$이다. 상수 C는 $ASC(A_i).cra$에서 한 상수영역의 도메인을 저장하는데 필요한 공간을 나타낸다. 질의 결과 비트맵 $ASC(A_i).cra[r].qrb(1 \le r \le 2m+1)$에는 길이 k인 비트맵이 영역의 수 만큼 필요하므로 $ASC(A_i).cra[r].qrb$를 위한 공간은 최대 $(2m+1)k$비트 필요하다. 이러한 속성 선택체가 n_p개 필요하므로, 총 최대 필요공간은 다음과 같이 정리된다.

$$n_p\{k+mC+(2m+1)k\} = n_p\{(2m+2)k+(2m+1)C\} = O(mkn_p). \quad \square$$

정리 3. 데이터 스트림 D의 각 튜플에 대해서 질의집합 Q를 처리하는데 필요한 시간은 최대 $O(n_p logm)$이다.

(증명) (알고리즘 1)에서 보듯이, 참여 속성 A_i에 대한 속성 선택체 $ASC(A_i)$를 처리하는데 있어서 병목지점은 영역배열 $ASC(A_i).cra$에서 현재 튜플의 속성 A_i 값을 포함하고 있는 영역을 찾는 것이다 (알고리즘 1의 함수 *find_region()*). 비트단위 연산의 처리시간은 무시할 만 하다. $(2m+1)$개의 영역을 가진 $ASC(A_i).cra$에서 목적 영역을 이진탐색방법으로 찾는다면, 검색 시간은 $log(2m+1)$이다. 질의 처리를 완성하기 위해서 최대 n_p개의 속성 선택체를 처리해야 하므로 총 최대 질의 처리시간은 다음과 같다.

$$n_p log(2m+1) = O(n_p logm). \quad \square$$

4. 속성 선택체 처리 순서

질의집합 Q의 참여 속성에 대한 속성 선택체의 처리 순서가 참여 속성의 선택률에 따라 질의성능에 심대한 영향을 끼칠 수 있기 때문에 효율적인 처리 순서를 결정하는 것은 질의성능의 측면에서 매우 중요하다. 전체적인 질의비용을 고려했을 때, 선택률이 낮은 참여 속성의 속성 선택체부터 처리하는 것이 바람직하나, 최소 속성커버 집합과 같이 질의 최적화를 위해 고려되어야 할 몇 가지 추가적인 요소가 있다.

4.1 속성 선택체 정렬 기준

4.1.1 최소 속성커버 집합

데이터 스트림 D의 튜플은 질의집합 Q에 속한 모든 질의들의 매칭 결과가 파악되지 않으면 제거할 수 없기 때문에, 빠른 튜플 필터링을 위해서는 질의집합 Q에 있는 모든 질의들에 대한 답을 줄 수 있는 참여 속성들을 먼저 처리하는 것이 필요하다. 이러한 참여 속성들의 집합을 속성커버 집합(Attribute Cover Set)이라 하고, 특히 최소한의 참여 속성들로 이루어진 속성커버 집합을 *최소 속성커버 집합(Minimum Attribute Cover Set)*이라 하고 다음과 같이 정의한다.

정의 2. (최소 속성커버 집합) 데이터 스트림 D와 질의집합 Q가 주어졌을 때, 질의집합 Q의 참여 속성들의 부분 집합 $B \subset A_p(Q)$에 대하여, $Q(B)$를 집합 B에 속하는 속성이 적어도 한 번 이상 질의의 선택 조건에 나타난 질의들의 집합으로 가정한다. $Q(B)=Q$일 때의 참여 속성집합 B를 질의집합 Q에 대한 *속성커버 집합*이라 하고, $ACS(Q)$로 표기한다. 또한, 속성커버 집합들 중 가장 적은 수의 참여 속성을 가지는 집합을 *최소 속성커버 집합*이라 하고, $MACS(Q)$로 표기한다. $\square$

4.1.2 참여 속성 시퀀스에 대한 질의 선택률

참여 속성의 질의 선택률은 해당 참여 속성을 포함한 선택 조건들을 적용하였을 시에 만족하는 질의의 비율로 정의한다. 전통적인 데이터베이스에서의 질의 최적화와 유사하게, 만족하지 않는 질의들을 가능한 한 일찍 종료하기 위해서는 참여 속성에 대한 속성 선택체의 처리 순서를 해당 참여 속성에 대한 질의 선택률의 오름차순으로 정하는 것이

효율적이다. 그러나, 속성 선택체의 최적화된 정렬을 위해서는 개개 참여 속성의 질의 선택률 보다는 속성 선택체의 처리 순서에 대한 질의 선택률이 중요하다. 왜냐하면, 각 속성 선택체의 처리결과가 다음 속성 선택체의 처리결과에 영향을 주기 때문이다. 그러므로, 본 논문에서는 참여 속성 시퀀스(p-attribute sequence)에 대한 질의 선택률을 정의한다. 이는 해당 시퀀스를 구성하는 참여 속성들의 속성 선택체들을 모두 처리하였을 시에 만족하는 질의의 비율로 다음과 같이 정의된다.

정의 3. (참여 속성 시퀀스의 질의선택률) 데이터 스트림 $D(A_1, A_2, \cdots, A_n)$에 대한 연속 질의 집합 $Q = (q_1, \cdots, q_k)$와 Q의 참여 속성 시퀀스 $\rho = A_x \rightarrow \cdots \rightarrow A_y$에 대하여, $EI_\tau(\rho)$는 τ 기간 동안 발생된 D의 튜플들을 시퀀스 ρ에 의해 처리한 결과 만족된 질의들의 총 수라 할 때, τ 기간의 시퀀스 ρ의 질의 선택률 $s_\tau(\rho)$는 다음과 같이 정의된다.

$$s_\tau(\rho) = \frac{EI_\tau(\rho)}{k^* |\tau|} \quad \text{(식 1)}$$

단, $|\tau|$는 τ 동안 발생한 튜플의 수를 의미한다. □

$EI_\tau(\rho)$는 (알고리즘 1)에 소개된 GRB를 이용하여 실시간에 쉽게 구할 수 있다. 즉, $EI_\tau(\rho)$는 τ 기간 동안 발생된 튜플에 대하여 시퀀스 ρ를 적용한 결과로 산출된 GRB에서 *1*의 개수를 더해 나간 결과와 같다. 이는 다음과 같은 식으로 표현된다.

$$EI_\tau(\rho) = \sum_{i=1}^{|\tau|} \eta(\rho, t_i) \quad \text{(식 2)}$$

단, $\eta(\rho, t_i)$는 튜플 t_i에 대해 시퀀스 ρ를 적용한 결과로 산출된 $GRB[1..k]$에서의 *1*의 개수이다.

4.2 속성 선택체 정렬 전략

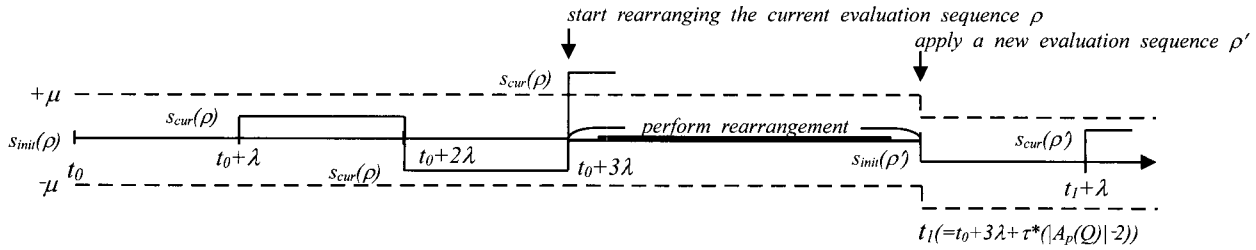
속성 선택체를 연산순서를 정할 때, 4.1절에서 제시한 최소 속성커버 집합은 목적 질의집합의 어떠한 질의도 만족시키지 못하는 튜플을 가능한 한 일찍 제거(filtering)하기 위한 기준이며, 참여 속성 시퀀스의 질의 선택률은 튜플에 의해 만족되지 않은 질의를 일찍 종료(terminating)시키기 위한 기준이다. 본 절에서는 이들을 이용한 속성 선택체 정렬 전략을 제시한다.

질의집합 Q의 참여 속성들의 속성 선택체 정렬을 위한 참여 속성 시퀀스를 찾는 첫 번째 단계는 빠른 튜플 제거를 위해 최소 속성커버 집합(MACS)을 찾는 것이다. 질의집합 Q에 대한 $MACS(Q)$를 찾기 위해 먼저 속성커버 집합 $ACS(Q)$를 찾는다. 이는 단순한 작업으로 참여 속성집합에 속하는 참여 속성의 수를 하나씩 늘려가면서, 그 집합에 속한 모든 참여 속성들에 대한 속성 선택체들의 질의 활용 비트맵(*qub*) 들에 대해 *bitwise-OR* 작업을 수행한다. 이러한 과정을 반복하여 *bitwise-OR* 연산의 결과 값이 모두 *1*로 구성된 비트맵이 될 때의 참여 속성집합이 $ACS(Q)$이 된다. 이렇게 찾은 여러 $ACS(Q)$ 중에서 가장 작은 수의 참여 속성을 가진 집합이 최소 속성커버 집합이다. 찾은 최소 속성커버 집합에 속한 참여 속성 수가 하나 이상일 때에는 질의 응답시간(response time)을 줄이기 위해서 최소 속성커버 집합을 구성하는 참여 속성의 시퀀스를 다음과 같이 구한다. 구하고자 하는 참여 속성 시퀀스를 ρ라 할 때, 우선 MACS에 속한 참여 속성 중 질의 선택률이 가장 낮은 참여 속성을 선택하여 길이 *1*인 ρ를 만든다. 그런 다음, 아직 ρ에 속하지 않은 MACS 내의 참여 속성들 각각을 붙여 길이가 *1* 확장된 후보 시퀀스 군을 형성한다. 이러한 후보 시퀀스들 중에 가장 낮은 질의 선택률을 가진 시퀀스가 참여 속성 하나가 확장된 시퀀스 ρ로 채택된다. 이러한 과정은 ρ에 MACS의 모든 참여 속성이 포함될 때까지 계속된다. 참여 속성 시퀀스를 구하는 마지막 단계는 MACS에 속하지 않은 참여 속성들의 시퀀스를 찾는 것이다. 이는 MACS 내의 참여 속성 시퀀스를 구하는 방식과 동일하다. 즉, MACS 내의 참여 속성 시퀀스를 초기 시퀀스 ρ로 하여 위에서 제시한 방식과 동일하게 ρ를 확장해 간다. 이러한 과정을 모든 참여 속성이 ρ에 속할 때까지 반복하면 종국적으로 최적화 참여 속성 시퀀스 ρ를 만들어 낼 수 있다.

이러한 방식으로 속성 선택체의 처리 순서를 결정하기 위해서는 질의 수행모듈 이외에 STREAM에서의 *StreaMon* [19]과 같은 모니터링 모듈이 필요하다. 모니터링 모듈에서는 참여 속성의 시퀀스를 하나씩 확장할 때, 후보 시퀀스 군에 속한 시퀀스들 각각에 대한 질의 선택률을 일정기간 동안 모니터링 하여 확장 시퀀스 결정을 위한 데이터로 활용한다. 즉, 수집된 후보 시퀀스들에 대한 질의 선택률들 중 최소값을 가진 시퀀스를 확장 시퀀스로 선택한다. 시퀀스들 각각의 질의 선택률은 (정의 3)에 따라 실시간에 쉽게 구할 수 있으며, 이들 중 최소값을 찾는 것 또한 후보 시퀀스들의 수가 많아야 참여 속성의 수를 절대 넘을 수 없으므로 비용의 측면에서 미미하다. 만약, 모니터링 기간에 스트림 튜플의 도착 비율(arriving rate)이 높아 실시간 모니터링에 부담이 있을 시에는 샘플링(sampling)을 통해 모니터링 대상 튜플을 제한한다.

정리 4. 참여 속성 시퀀스의 길이를 하나 확장하기 위한 모니터링 기간이 τ일 때, 전체 속성 선택체 처리 순서를 결정하기 위해 필요한 기간은 $\tau^*(|A_p(Q)|-2)$ 이다.

(증명) 참여 속성 시퀀스를 찾는 작업은 크게 $MACS(Q)$에서의 참여 속성 시퀀스를 찾는 작업과 $MACS(Q)$의 외부 즉, $A_p(Q) - MACS(Q)$에서의 참여 속성 시퀀스를 찾는 작업으로 나뉜다. 위에서 기술하였듯이, 이 두 작업은 공통적으로 대상 참여 속성들에 대한 전체를 한꺼번에 정렬하는 방식이 아니라, 모니터링 모듈에서 현재 모니터링 결과를 바탕으로 점진적으로 이루어 진다. 즉, 현재까지 찾은 시퀀스에서 하나 확장된 후보 시퀀스들 각각에 대한 질의 선택률을 일정기간 모니터링 하여 가장 낮은 질의 선택률을 가진 후보 시퀀스를 하나 확장된 시퀀스로 확정한다. 가정에서 시퀀스를 하나 확장하기 위한 모니터링 기간을 τ라고 가정하였으므로,



(그림 3) 모니터링 모듈에서의 속성 선택체 처리 순서에 대한 적응적 최적화

$MACS(Q)$에서의 참여 속성 시퀀스를 찾기 위한 기간은 $\tau*(|MACS(Q)|-1)$ 이고, $A_p(Q)-MACS(Q)$에서의 참여 속성 시퀀스를 찾기 위한 기간은 $\tau*(|A_p(Q)-MACS(Q)|-1)$ 이다. 그러므로, 전체 속성 선택체 처리 순서를 결정하기 위해 필요한 총 기간은 $\tau*(|MACS(Q)|-1)+\tau*(|A_p(Q)-MACS(Q)|-1) = \tau*(|A_p(Q)|-2)$ 이 된다.

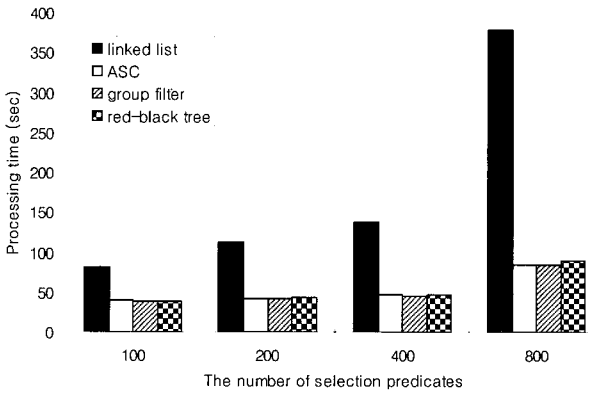
예를 들어, (그림 2)의 질의집합 $Q=\{q_1, q_2, q_3\}$의 참여 속성집합 $A_p(Q)=\{A_1, A_2, A_3\}$이다. 각 참여 속성에 대한 속성 선택체의 질의 활용 비트맵에 따라 $MACS(Q) = \{A_1\}$이 된다. $|A_p(Q)| = 3$이므로, 전체 참여 속성 시퀀스를 구하기 위해 τ 기간이 필요하다. 만약, 튜플 t_1, t_2, t_3가 모두 이 기간 동안에 발생되었다고 가정하면, $s(A_1 \rightarrow A_2) = 2/9$이고 $s(A_1 \rightarrow A_3) = 1/3$이므로 $A_1 \rightarrow A_2$가 시퀀스로 채택된다. 그러므로, 전체 참여 속성 시퀀스는 $A_1 \rightarrow A_2 \rightarrow A_3$를 찾을 수 있다.

4.3 적응적 재정렬 전략

연속 질의 집합 Q의 참여 속성에 대한 질의 선택률은 시간에 따라 변화하기 때문에, 속성 선택체의 처리 순서를 가능한 한 효율적으로 유지하기 위해서는 현재의 참여 속성 시퀀스의 질의 선택률이 주기적으로 모니터 되어야 한다. 이러한 주기를 질의 선택률 재계산 주기 λ라 한다. 속성 선택체의 처리 순서는 현재의 참여 속성 시퀀스의 질의 선택률이 일정한 임계치 이상의 변화를 보였을 때 재정렬된다. 이러한 임계치를 재정렬 임계치 μ라 한다. 현 참여 속성 시퀀스 ρ에 대하여, $s_{init}(\rho)$를 ρ가 현재의 시퀀스로 결정될 당시의 질의 선택률이라 정의하고 $s_{cur}(\rho)$를 현 시점에서의 ρ의 질의 선택률이라 정의한다. $s_{cur}(\rho)$는 질의 선택률 재계산 주기 λ 마다 갱신된다. 그리고, (식 3)을 만족할 때 마다, 현재의 참여 속성 시퀀스는 재정렬되고 $s_{init}(\rho)$의 값 또한 재계산되는 질의 선택률로 갱신된다.

$$\Delta s(\rho) = \left| \frac{s_{cur}(\rho) - s_{init}(\rho)}{s_{init}(\rho)} \right| \geq \mu \qquad \text{(식 3)}$$

재정렬 임계치 μ는 현 속성 선택체의 처리 순서의 비효율성에 대한 최대 허용한계로서 언제 재정렬을 시작하는지에 대한 기준을 제공한다. (그림 3)은 모니터링 모듈에서 현재 참여 속성 시퀀스의 선택률이 모니터링 되고 선택률 값에 따라 속성 선택체의 처리 순서가 재정렬되는 모습을 보여준다. 그림에서 $t_0+\lambda$와 $t_0+2\lambda$ 시점에서는 재정렬 작업이 일어나지



(그림 4) 동등 비교 연산에서의 질의성능 비교

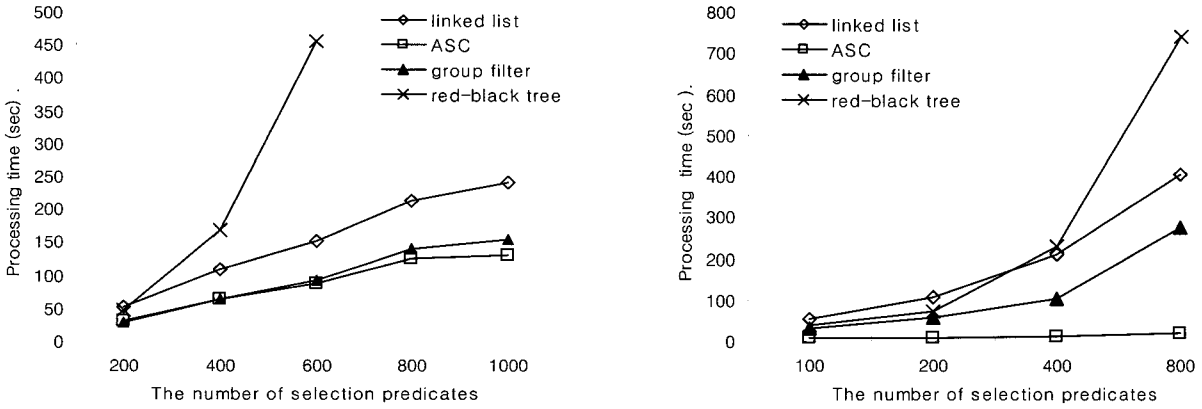
않는다. 왜냐하면, $\Delta s(\rho)$ 값이 임계치 μ 값보다 작기 때문이다. 재정렬 작업은 $t_0+3\lambda$ 시점에서 시작된다. 4.2절에 따라, 정렬 시에 참여 속성 시퀀스의 확장에 필요한 기본 모니터링 시간을 τ라 하면, 재정렬 작업은 $t_1(=t_0+3\lambda+\tau*(|A_p(Q)|-2))$ 시점에 완료되고 이 시점부터 새로운 시퀀스에 의해 속성 선택체가 처리된다. μ는 데이터 스트림의 변화를 얼마나 정확하게 속성 선택체 처리 순서에 반영하는가에 대한 척도이다. μ 값이 작을수록 속성 선택체의 처리 순서는 자주 재정렬되어 최적화 처리 순서를 찾지만 재정렬 부하 또한 증가한다.

5. 실험

본 장에서는 여러 실험을 통하여 이전에 제안된 연속 질의 처리방법과 본 논문에서 제안한 연속 질의 처리방법을 비교하여 성능을 검증하였다. 모든 실험은 1G의 램(RAM)을 가진 2.6 GHz 팬티엄 컴퓨터와 리눅스 7.3 환경에서 실험되었으며, C언어로 구현되었다.

5.1 속성 선택체의 효율성

실험 5.1에서는 연속 질의의 그룹처리 방식에 대한 비교 실험으로 CACQ[7]의 그룹필터(Grouped filter) 기법, PSoup[9]의 홍흑 트리(Red black tree) 기법 그리고 단순히 조건절을 연결한 연결리스트(Linked list) 기법을 본 논문에서 제안한 속성 선택체(ASC) 기법과 비교하여 효율성을 검증 하였다. 각 실험에서 사용한 데이터 집합은 하나의 속성을 가지는 5,000,000개의 튜플로 이루어졌으며 속성 값은 0~999의



(a) 조건형태 : $R.a >= C$ (b) 조건형태 : $R.a >= C_1$ and $R.a <= C_2$

(그림 5) 비동등 비교 연산에서의 질의성능 비교

〈표 1〉 실험 데이터 집합 명세

D_1	각 속성의 값이 일정한 분포를 가짐
D_2	정규분포 그래프와 유사한 형태로 중간 값의 분포가 높음
D_3	펄스의 형태와 비슷한 분포로 처음 1/2은 낮은 속성 값을 가지고 다음 1/2은 높은 속성 값을 가짐

〈표 2〉 실험 질의 집합 명세

	그림 6, 그림 9	그림 7	그림 8
연속 질의 수	50	30	40
참여 속성의 수	10	7	20
선택 조건의 수	204	104	236
최소 속성커버 집합의 수	1	2	1
최소 속성커버 집합 내의 참여 속성의 수	3	2	5

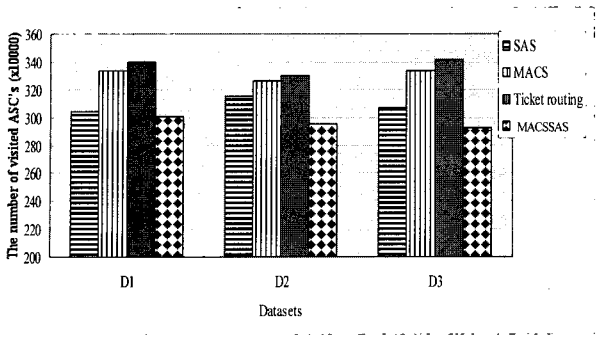
값을 무작위로 발생시켰다.

(그림 4)은 등록된 연속 질의들의 조건절의 형태가 동등 비교 연산(=)으로만 이루어진 선택 조건(selection predicates)의 수에 따른 수행시간의 변화를 보여준다. 조건절의 형태가 동등 연산인 경우 선택 조건의 수가 증가함에 따라, 가장 좋지 않은 성능을 보이는 연결리스트 기법을 제외하고는 그룹필터 기법, 홍흑 트리 기법 그리고 속성 선택체 기법이 분명한 차이를 보이지 않는다.

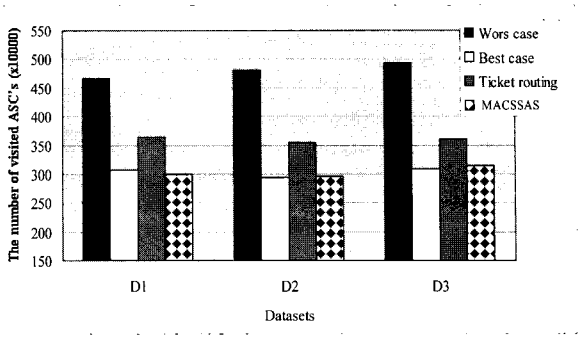
(그림 5)는 등록된 연속 질의들의 조건절의 형태가 비동등 비교 연산(>, <, >=, <=)으로만 이루어진 선택 조건(selection predicate)의 수에 따른 수행시간의 변화를 보여준다. (그림 5)의 (a)와 (b) 모두에서 선택 조건의 수가 증가함에 따라 속성 선택체 기법이 가장 좋은 성능을 보인다. 그룹필터 기법은 선택 조건의 비동등 비교 연산에 대해 비교상위 트리(greater-than tree) 또는 비교하위 트리(less-than tree)에 대한 순차 탐색을 한다. 홍흑 트리 기법에서는 튜플이 새로 들어 올 때마다 비교 상수의 개수(노드수) 만큼을 재귀적으로 순차 탐색하며 방문된 각 노드 내에서도 비교 연산자의 종류 만큼의 순차탐색을 하기 때문에 가장 나쁜 성능을 보인다. 이에 비해서 속성 선택체 기법은 선택 조건의 비교 연산 형태에 관계없이 튜플의 속성 값이 속한 영역을 찾기 위한 이진 탐색을 수행함으로 탐색 비용을 $O(log_2 k)$까지 낮출 수 있다. 결론적으로 (그림 4)와 같이 동등 비교 연산에 대해서는 분명한 성능의 차이를 보이지 않지만, (그림 5)과 같이 비동등 비교 연산에서는 속성 선택체 기법이 다른 기법들에 비해 월등히 좋은 성능을 보인다.

5.2 속성 선택체 순서에 따른 효율성

본 절에서는 속성 선택체(ASC)의 다양한 처리 순서에 따른 비교를 통하여 성능을 검증하였다. 각 실험에서 사용한 데이터 집합은 *20*개의 속성을 가지는 *5,000,000*개의 투플로 이루어져 있다. 속성 값은 무작위로 발생시킨 *0~99*의 값을 가지며 <표 1>과 같은 속성 값의 분포를 가지는 세 종류의 데이터 집합들을 생성하였고, <표 2>와 같은 특징을 가지는 연속 질의 집합들을 사용했다. 모든 실험 그래프의 Y축은 연속 질의의 처리비용을 나타내며, 처리비용은 각 방문된 속성 선택체의 처리 비용이 동일하다는 가정 하에 속성 선택체의 방문 횟수로 정의한다.



(a) 제안된 기법들과 티켓라우팅과의 비교



(b) 제안된 기법과 티켓라우팅 그리고, 최선의 경우와 최악의 경우 사이의 비교

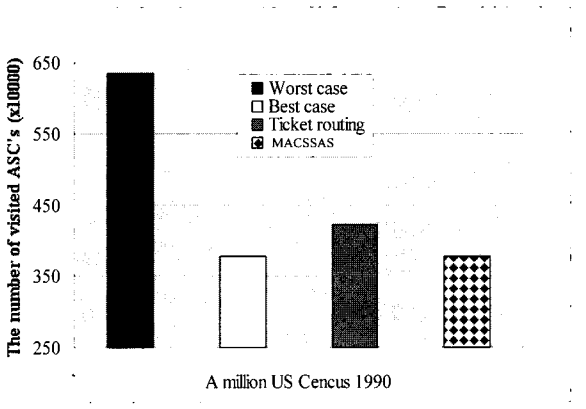
(그림 6) 속성 선택체의 처리 순서에 따른 질의 처리비용 비교

(그림 6)는 <표 1>에서 제시한 세 종류의 데이터 집합을 대상으로 4장에서 제안한 속성 선택체의 정렬 방법에 따른 질의 처리비용과 Eddy[14]의 티켓 라우팅(ticket routing)의 처리비용을 비교한 결과이다. (그림 6(a))는 최소 속성커버 집합에 의한 속성 선택체 정렬 방법(MACS), 참여 속성 시퀀스의 질의 선택률이 낮은 순으로 속성 선택체의 처리 순서를 결정하는 방법(SAS), 그리고 본 논문이 제시한 방법인 최소 속성커버 집합과 SAS를 결합한 방법(MACSSAS), 마지막으로 티켓 라우팅 방법을 비교하였다. 전체적으로 티켓 라우팅 방법보다 본 논문에서 제안된 방법이 좋은 성능을 보이며 특히 데이터집합 D_2와 D_3에 대해서는 MACSSAS가 훨씬 더 좋은 성능을 보인다. 이러한 결과가 나온 이유는 데이터 분포 형태가 D_2 또는 D_3와 같은 형태를 가지는 데이터 스트림에서는 질의를 처리하는 과정에서 많은 튜플 제거(tuple pruning)와 질의 종료(query terminating)가 일어날 가능성이 커서 상대적으로 더 적은 수의 속성 선택체를 방문하게 되기 때문으로 해석된다.

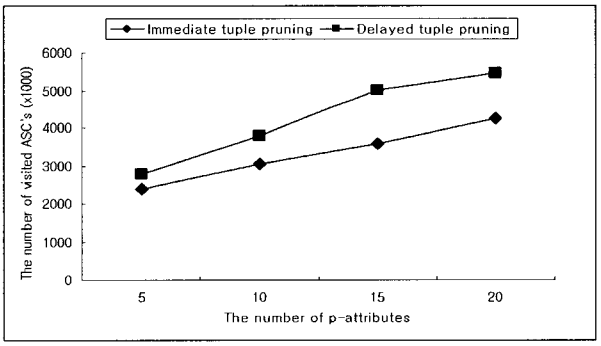
(그림 6(b))는 최악의 조건(worst case)과 최상의 조건(best case)에 대비하여 MACSSAS와 *Ticket Routing*의 처리비용을 비교하였다. 이 실험에서는 <표 2>에 기술된 7개의 연속 질의 집합들을 *2,000*번 수행하였으며 매 수행 시마다 속성 선택체의 처리 순서를 임의로 재비치 시켜 모든 가능한 경우가 도출되도록 하였다. 이렇게 해서 얻은 결과들 중에 가장 많은 속성 선택체를 방문한 경우를 최악의 경우로, 가장 적은 속성 선택체를 방문한 경우를 최상의 경우로 정의하였다. 이 그림에서 보듯이, MACSSAS와 티켓 라우팅은 최악의 경우와 최상의 경우 사이의 성능을 보이며, 데이터 집합들과 상관없이 MACSSAS가 최상의 경우에 가장 가까운 성능을 보여준다.

(그림 7)은 실제 데이터 집합인 million US Census 1990 데이터 투플 *1,000,000*건을 대상으로 실험하였으며, 실험 결과는 (그림 6(b))의 결과와 유사하게 MACSSAS가 최상의 경우에 근접한 성능을 보였다.

(그림 8)은 3.2절에서 소개한 튜플 제거(tuple pruning) 효과를 보여준다. 즉시 튜플 제거 (Immediate tuple pruning)는 (알고리즘 1) 상에서 전역비트맵 변수 *GRB*의 모든 비트



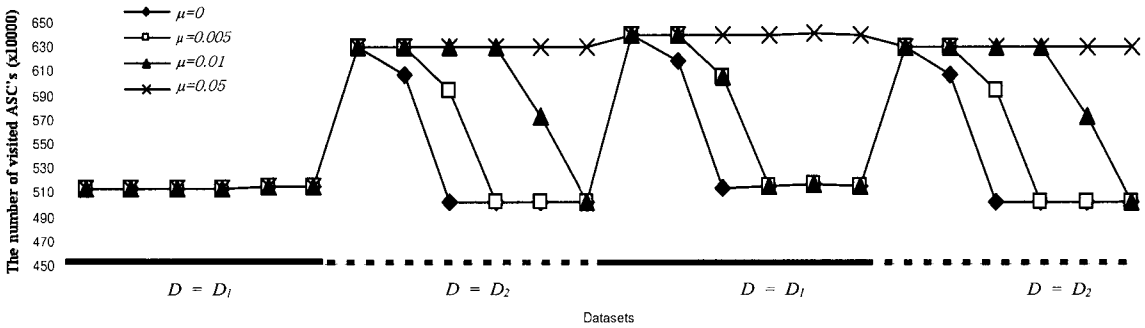
(그림 7) 실제 데이터집합을 대상으로 한 질의성능 비교



(그림 8) 튜플 제거 효과

가 *0*이 되면 어떤 질의에도 만족하지 않는 튜플로 판명된 결과이기 때문에, 처리 순서 상의 나머지 속성 선택체를 방문하지 않고 즉시 해당 튜플을 제거하는 방법이다. 반대로 연기 투플 제거(delayed tuple pruning)는 모든 속성 선택체의 방문을 완료할 때까지 만족하지 않은 튜플의 제거를 연기하는 방법이다. 이 그림에서 보듯이, 즉시 튜플 제거 방법이 훨씬 더 효율적이며, 참여 속성의 수가 늘어날수록 두 방법에 대한 질의성능의 차이는 확대된다.

(그림 9)은 재정렬 임계치 μ 값에 따른 속성 선택체 처리 순서의 적응적 재정렬 효과를 보여준다. 참여 속성 시퀀스의 질의 선택률에 대한 동적인 변화를 유도하기 위해서 데



(그림 9) 재정렬 임계치 μ 값에 따른 적응적 속성 선택체 처리 순서 재정렬 효과

이터분포가 다른 <표 1>의 데이터 셋 D_1과 D_2를 번갈아 사용하면서 전체 데이터 셋 D를 구성하였다. 질의셋은 (그림 6)의 질의셋과 동일한 것을 사용하였다. 스트림 튜플이 일정한 간격으로 도착한다는 전제하에 선택률 재계산 주기 λ는 *80,000* 튜플이 도착하는데 걸리는 시간으로 가정하였다. λ 주기마다 현 참여 속성 시퀀스의 질의 선택률을 계산하여 4.3절의 (식 3)을 만족하면 속성 선택체의 처리 순서를 재정렬하였다. 실험결과에서 보듯이, 서브 데이터 셋이 변화할 때 마다 질의 처리비용이 급격하게 증가한다. 이는 현 속성 선택체의 처리 순서가 데이터 분포가 다른 데이터 셋에서는 더 이상 최적화된 처리 순서가 아니기 때문이다. 그러나, 적응적인 처리 순서 재정렬 작업을 통하여 다시 최적화된 처리비용에 가까워짐을 알 수 있다. 또한, μ 값이 작을수록 재정렬 작업은 더 자주 수행되어 더 빨리 최적화 처리 순서를 찾아간다. 그러나, 그에 따른 재정렬 부하 또한 커진다.

6. 결 론

본 논문에서는 데이터 스트림 환경에서 다중 연속 질의를 효율적으로 처리하기 위해서 속성기반 구조체인 속성 선택체와 그것을 이용한 매칭 알고리즘을 제안하였다. 제안된 접근 방법은 대상 다중 연속 질의들의 공통된 선택 조건들을 공유함으로써 공간 사용량을 절약하였고, 질의의 선택 조건에 나타난 참여 속성의 비교상수를 기준으로 한 질의의 부분적인 매칭 결과를 미리 계산하여 속성 선택체에 저장함으로써 실시간 부하를 줄었다. 덧붙여서, 최소 속성커버 집합과 참여 속성의 질의 선택률에 따라 속성 선택체의 처리 순서를 정렬하는 질의 최적화 기법을 제안하였다. 본 질의 최적화의 목적은 만족하지 않는 질의와 튜플을 가능한 한 일찍 종료하고 제거함으로써 불필요한 연산을 줄이는데 있다. 또한, 제안된 방법과 최적화 기법들을 다양한 실험을 통하여 검증하였는데, 수행된 실험을 통해 본 논문의 방법론이 합성 또는 실제 데이터 집합을 포함한 다양한 데이터분포를 지닌 데이터 스트림에 대한 다중 연속 질의 처리에서 고루 효율적이라는 것을 보였다.

본 논문에서 제안된 방법론은 속성 기반의 다중 연속 질의 처리 방법론으로 요약될 수 있다. 이는 선택 연산(selection operation) 뿐만 아니라, 집단화(aggregation) 또는 조인(join) 과 같은 여러 가지 다양한 질의 연산에도 효과적으로 적용될 수 있다. 왜냐하면, 집단화나 조인 연산 또한 속성 기반의 연산 성격을 지니고 있기 때문이다. 본 논문에서 제안된 여러 가지 기법들을 바탕으로, 향후 집단화나 조인과 같은 좀 더 진전된 질의 연산 주제에 대한 처리방안을 연구하고, 나아가서는 이를 바탕으로 다중 스트림에 대한 다중 연속 질의 처리 시스템을 설계하는 것은 매우 의미 있는 일일 것이다.

참 고 문 헌

[1] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems," *In Proc. 2002 ACM Symp. on Principles of Database Systems*, pp.1-16, June, 2002.

[2] D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, and S. Zdonik, "Monitoring Streams-A new class of data management applications," *In Proc of the 28th International Conference on VLDB*, pp.15-226, August, 2002.

[3] S. Chandrasekaran et al, "TelegraphCQ: Continuous dataflow processing for an uncertain world," *In Proc. First Biennial Conf. on Innovative Data Systems Research*, pp.269-280, Jan, 2003.

[4] J. Chen, D. J. DeWitt, F. Tian and Y. Wang, "NiagaraCQ: A Scalable Continuous Query System for Internet Databases," *In proc.of ACM SIGMOD 2000 Conf.*, pp.379-390, May, 2000.

[5] D. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, S. Zdonik, "Aurora: A New Model and Architecture for Data Stream Management," *In VLDB Journal*, pp.120-139, August, 2003.

[6] Widom J, Babu S, "Continuous queries over data streams," In ACM SIGMOD Record, pp.109-120, September, 2001.

[7] Samuel R. Madden, Mehul A. Shah, Joseph M. Hellerstein and Vijayshankar Raman, "Continuously Adaptive Continuous Queries over Streams," *In proc. 2002 ACM SIGMOD Conf*, June, 2002.

[8] Motwani R, Widom J, Arasu A, Babcock B, Babu S, Datar M, Manku G, Olston C, Rosenstein J, Varma R, "Query Processing, Resource Management, and Approximation in a

Data Stream Management System," *In Proc. of the 2003 CIDR*, January, 2003.

[9] Sirish Chandrasekaran and Michael J. Franklin, "Streaming Queries over Streaming Data," *In Proc. of the 28th Intl. Conf. on Very Large Data Bases*, August, 2002.

[10] D. Carney, U. Çetintemel, A. Rasin, S. Zdonik, M. Cherniack, M. Stonebraker, "Operator Scheduling in a Data Stream Manager," *In proc of the 29th International Conference on Very Large DataBases*, 2003.

[11] N. Tatbul, U. Çetintemel, S. Zdonik, M. Cherniack, M. Stonebraker, "Load Shedding in a Data Stream Manager," *In proc of the 29th International Conference on Very Large Data Bases*, pp.309-320, 2003.

[12] Jianjun Chen, David J. DeWitt, and Jeffrey F. Naughton, "Design and Evaluation of Alternative Selection Placement Strategies in Optimizing Continuous Queries," *In proc of the 18th International Conference on ICDE*, pp.345-356, Feb, 2002.

[13] Jianjun Chen, David J. DeWitt, "Dynamic Re grouping of Continuous Queries," *In Proc of the 28th VLDB Conference*, pp.430-441, 2002.

[14] Ron Avnur, Joe Hellerstein, "Eddies: Continuously Adaptive Query Processing," *In proc. of the 2000 ACM SIGMOD Intl. Conf. on Management of Data*, Dallas, pp. 261-272, 2000.

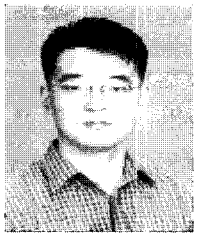
[15] Vijayshankar Raman, Amol Deshpande, and Joseph M. Hellerstein, "Using State Modules for Adaptive Query Processing" *In ICDE*, 2003.

[16] S. Babu, R. Motwani, K. Munagala, I. Nishizawa, and J. Widom, "Adaptive Ordering of Pipelined Stream Filters," *In SIGMOD*, pp.407-418, June, 2004.

[17] K. Munagala, U. Shrivastava, and J. Widom, "Optimization of Continuous Queries with Shared Expensive Filters," *In Proc of the 32th VLDB Conference*, Sep, 2006.

[18] H.S. Lim, J.G. Lee, M.J. Lee, K.Y. Whang, I.Y. Song, "Continuous Query Processing in Data Streams Using Duality of data and Queries," *In Proc of the 32th VLDB Conference*, Sep, 2006.

[19] S. Babu and J. Widom, "StreaMon: An Adaptive Engine for Stream Query Processing," *In proc.of ACM SIGMOD 2004 Conf.*, June, 2004.



이 현 호

e-mail : hhlee@ianyang.ac.kr

1993년 연세대학교 전산과학과 (학사)

1995년 연세대학교 컴퓨터과학과 (석사)

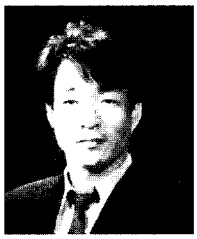
1996년~1999년 육군사관학교 교수부 전산학과 전임강사

2000년~2001년 ㈜엔코아정보컨설팅 선임컨설턴트

2001년~현 재 안양과학대학 컴퓨터정보학부 조교수

2002년~현 재 연세대학교 컴퓨터과학과 박사과정

관심분야 : 데이터 스트림, XML 스트림, 연속 질의 처리, 질의 최적화



이 원 석

e-mail : leewo@database.yonsei.ac.kr

1985년 미국 보스턴대학교 컴퓨터공학과 (공학사)

1987년 미국 퍼듀대학교 컴퓨터공학과 (공학석사)

1990년 미국 퍼듀대학교 컴퓨터공학과 (공학박사)

1990년~1992년 삼성전자 선임연구원

1993년~1999년 연세대학교 컴퓨터과학과 조교수

1999년~2004년 연세대학교 컴퓨터과학과 부교수

2004년~현 재 연세대학교 컴퓨터과학과 교수

관심분야 : 분산데이터베이스, 미디에이터시스템, 데이터마이닝, 데이터스트림