데이타와 질의의 이원성을 이용한 데이타스트림에서의 연속질의 처리

(Continuous Query Processing in Data Streams Using Duality of Data and Queries)

임 효 상 [†] 이 재 길 [††] 이 민 재 [†††] 황 규 영 [††††]

(Hyo-Sang Lim) (Jae-Gil Lee) (Min-Jae Lee) (Kyu-Young Whang)

요 약 본 논문은 데이타스트림 환경에서 연속질의를 효율적으로 처리하는 방법을 다룬다. 먼저, 기존의 질의 처리 방법을 데이타 엘리먼트와 질의 중에서 어느 것을 먼저 선택하고 수행을 시작하느냐에 따라서, 서로 이원적인 두 가지 방법인 데이타-이니셔티브(data-initiative)와 질의-이니셔티브(query-initiative)로 분류한다. 이러한 분류는 기존의 질의 처리 연구에서 데이타와 질의를 서로 다르게(asymmetrically) 취급하였다는 것에 기인한다. 기존의 연속질의 처리에서는 이원적인 질의 처리 방법 중에서 데이타-이니셔티브 방법만이 사용되었기 때문에, 질의-이니셔티브 방법에서 얻을 수 있는 성능 상의 이점이 간과되었다. 이러한 문제를 해결하기 위해, 데이타와 질의를 동등하게(symmetrically) 볼 수 있다는 점에 착안한다. 본 논문에서는 **데이타와 질의의 이원성 모델**(*Duality Model of Data and Queries*)을 제안하고 이 모델에 기반하여 연속질의 처리 문제를 다차원 공간에서의 공간조인 문제로 변환하는 새로운 관점을 제시한다. 그리고, 공간조인 기반 연속질의 처리 알고리즘인 Spatial Join CQ를 제안한다. *Spatial Join CQ*는 다차원 공간상에 영역으로 표현된 데이타 엘리먼트들의 집합과 질의들의 집합으로부터 서로 겹치는 쌍을 찾음으로써 연속질의를 처리한다. 제안하는 알고리즘은 대칭적인(symmetric) 연산인 공간조인으로 겹치는 영역들을 찾아냄으로써 서로 이원적인 두 가지 질의 처리 방법의 효과를 동시에 얻는다. 성능 평가 결과, 제시하는 알고리즘은 기존의 방법에 비해서 단순 선택 연속질의는 최대 36배, 슬라이딩 윈도우 조인 연속질의는 최대 7배의 성능 향상을 보였다.

키워드 : 데이타스트림, 연속질의 처리, 이원성, 공간 조인, 배치 처리

Abstract In this paper, we deal with a method of efficiently processing continuous queries in a data stream environment. We classify previous query processing methods into two dual categories - *data-initiative* and *query-initiative* - depending on whether query processing is initiated by selecting a data element or a query. This classification stems from the fact that data and queries have been treated asymmetrically. For processing continuous queries, only data-initiative methods have traditionally been employed, and thus, the performance gain that could be obtained by query-initiative methods has been overlooked. To solve this problem, we focus on an observation that data and queries can be treated symmetrically. In this paper, we propose the *duality model of data and queries* and, based on this model, present a new viewpoint of transforming the continuous query processing problem to a multi-dimensional spatial join problem. We also present a continuous query processing algorithm based on spatial join, named *Spatial Join CQ*. Spatial Join CQ processes continuous queries by finding the pairs of overlapping regions from a set of data elements and a set of queries defined as regions in the multi-dimensional space. The algorithm achieves the effects of both of the two dual

· 본 연구는 첨단정보기술연구센터를 통하여 과학기술부/한국과학재단으로부터 지원을 받았음

† 학생회원 : 한국과학기술원 전산학과
lorien@mozart.kaist.ac.kr

†† 정 회 원 : 한국과학기술원 전산학과
jglee@mozart.kaist.ac.kr

††† 정 회 원 : (주)네오위즈 연구소 연구원
mjlee@mozart.kaist.ac.kr

†††† 종신회원 : 한국과학기술원 전산학과 교수
kywhang@mozart.kaist.ac.kr

논문접수 : 2005년 8월 12일
심사완료 : 2006년 2월 2일

methods by using the spatial join, which is a symmetric operation. Experimental results show that the proposed algorithm outperforms earlier methods by up to 36 times for simple selection continuous queries and by up to 7 times for sliding window join continuous queries.

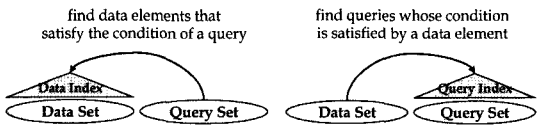
Key words : Data stream, Continuous query processing, Duality, Spatial join, Batch processing

1. 서 론

데이타스트림이란 순차적으로 생성되는 데이타 엘리먼트들의 연속(sequence)이다[1,9,16]. 데이타스트림의 대표적인 예로는 센서 데이타와 네트웍 패킷 데이타 등이 있다. 데이타스트림의 주요한 특징은 실시간적이고(real-time), 연속적이고(continuous), 빠르고(rapid), 그리고 무한히(unbounded) 새로운 데이타가 생성된다는 것이다[9,16]. 이러한 특징으로 인하여 데이타스트림 환경에서는 데이타의 도착 순서를 통제하거나 모든 데이타를 저장하는 것이 불가능하다[16]. 데이타의 이러한 특성 때문에 데이타스트림에서의 질의는 데이타를 모두 저장해 놓은 상태에서 수행하는 일회성-질의(one-time query) 형태가 아니라 질의를 미리 등록해 놓고 새로 입력된 데이타에 대해서 즉시 혹은 주기적으로 질의를 수행하여 결과를 돌려주는 **연속질의**(*continuous query*)의 형태를 가진다[16,21].

이러한 새로운 형태의 데이타와 질의를 효율적으로 관리하고 수행하기 위하여 많은 연구가 진행되었으며, NiagaraCQ[5], TelegraphCQ[4], Aurora[25], 그리고 STREAM[1] 등과 같은 데이타스트림 시스템들이 개발되었다. 기존의 데이타스트림 연구에서는 주로 저장된 연속질의들에 대해서 색인을 생성[4,5]하거나, 연속질의들 간의 공통된 연산을 공유[1,5,25]하는 등의 방법을 적용함으로써 연속질의를 빠르게 처리하고자 하였다.

데이타스트림 시스템에서의 연속질의 처리는 데이타베이스 시스템에서의 질의 처리와 이원적인 형태를 가진다. 데이타베이스 시스템에서는 질의 처리를 위해 그림 1(a)와 같이 저장된 데이타 엘리먼트들의 집합에 대한 색인을 생성하고, 각각의 질의에 대해서 이 색인을 사용하여 조건을 만족하는 데이타를 빨리 찾는 형태를 취하고 있다. 우리는 이러한 질의 처리 방법을 **질의-이니셔티브**(*query-initiative*)라고 부른다. 반면, 데이타스트림 시스템에서는 연속질의 처리를 위해 그림 1(b)와 같이 저장된 질의들의 집합에 대한 색인을 생성하고, 각각의 입력 데이타에 대해서 이 색인을 사용하여 조건을 만족하는 질의들을 빨리 찾는 형태를 취하고 있다. 우리는 이러한 질의 처리 방법을 **데이타-이니셔티브**(*data-initiative*)라고 부른다. 하지만 이러한 각각의 방법은 데이타나 질의 한쪽에만 최적화 기법을 적용함으로써, 나머지 한쪽에도 최적화 기법을 적용하여 얻을 수 있는 성능상의 장점들을 간과하고 있다.



(a) 질의-이니셔티브 방법 (b) 데이타-이니셔티브 방법

그림 1 서로 이원적인 두 가지 질의 처리 방법

본 논문에서는 서로 이원적인 질의-이니셔티브와 데이타-이니셔티브 방법 양쪽의 이점을 동시에 얻을 수 있는 대칭적인(symmetric) 연속질의 처리 방법을 제안한다. 이러한 방법은 그림 2에서 표현한 것과 같이 데이타 집합과 질의 집합에 모두 색인을 구성하고, 이를 동시에 사용하여 질의와 조건을 만족하는 데이타 엘리먼트의 쌍을 찾는다.

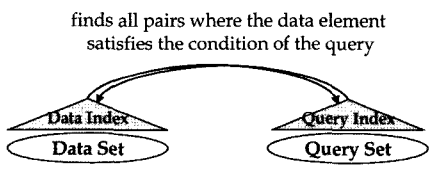


그림 2 데이타-이니셔티브와 질의-이니셔티브의 이점을 동시에 얻는 대칭적인 질의 처리 방법

본 논문의 공헌은 다음과 같다. 첫째, 데이타와 질의를 동등하게 취급하는 **데이타와 질의의 이원성 모델**(*Duality Model of Data and Queries*)을 제안한다. (이후부터는 간단히 **이원성 모델**이라 부른다.) 이원성 모델은 질의와 데이타를 다차원 공간상의 영역으로 동등하게 표현하고, 서로 겹치는 질의와 데이타의 영역들을 찾는 연산으로 질의 처리를 정의한다. 이 모델은 서로 이원적인 데이타-이니셔티브와 질의-이니셔티브 방법을 하나의 질의 처리 모델 안에서 통합한다.

둘째, 이원성 모델에 기반하여 연속질의 처리 문제를 다차원 공간에서의 공간조인 문제로 변환하는 새로운 관점을 제시한다. 공간조인이란 주어진 공간 관계(예를 들어, 본 논문에서는 서로 겹치는 관계)를 만족하는 모든 공간 객체들의 쌍들을 찾는 연산이다[3]. 이원성 모델에서는 서로 겹치는 영역의 쌍을 찾는 연산으로 질의 처리를 정의하므로, 공간조인을 연속질의 처리에 그대로 적용할 수 있다. 이러한 변환을 통하여, 데이타스트림에

서의 연속질의 처리라는 새로운 문제를 이미 많이 연구된 공간조인 기법들을 사용하여 풀 수 있다.

세째, **공간조인 기반 연속질의 처리 알고리즘**(*Spatial Join-Based Continuous Query Processing Algorithm*)을 제안하고 실험을 통하여 우수성을 보인다. 이 알고리즘은 대칭적인 연산인 공간 조인을 사용하여 데이타와 질의에 대해 모두 최적화를 수행할 수 있도록 함으로써 서로 이원적인 두 가지 질의 처리 방법의 효과를 동시에 얻는다. 이러한 특성으로 인하여, 데이타 집합에 대한 최적화의 이점을 얻을 수 있는 배치 처리에서 특히 우수한 성능을 보인다.

본 논문의 구조는 다음과 같다. 제2장에서는 본 논문에서 다루고자 하는 데이타스트림과 연속질의의 형태에 대해서 설명한다. 제3장에서는 기존의 연속질의 처리 방법에 대해서 살펴본다. 제4장에서는 이원성 모델을 제안한다. 제5장에서는 공간조인 기반 연속질의 처리 알고리즘을 제안하고, 제6장에서는 슬라이딩 윈도우 조인을 수행할 수 있도록 확장한다. 제7장에서는 성능 평가 결과를 제시한다. 마지막으로, 제8장에서는 결론을 맺는다.

2. 배 경

본 장에서는 데이타스트림과 연속질의에 대한 기본 지식으로서 데이타스트림의 형태, 연속질의의 형태, 연속질의의 처리 형태를 소개한다. 또한, 본 논문에서 다루고자 하는 데이타스트림과 연속질의의 형태를 정의한다.

2.1 데이타스트림의 데이타 형태

TelegraphCQ[4], Aurora[25], STREAM[1] 등 대부분의 기존 데이타스트림 시스템들은 입력 데이타로서 데이타스트림 소스별로 고정된 스키마를 가지는 관계형 튜플 형태의 데이타를 다룬다. 본 논문에도 마찬가지로 이와 같은 형태의 데이타를 취급한다.

본 논문에서는 설명의 편의를 위하여 모든 속성의 도메인이 0부터 1사이의 실수라고 가정한다. 여기서 0은 도메인의 최소값을 의미하고, 1은 최대값을 의미한다. 모든 값은 해쉬 등을 사용하여 실수 값으로 변환될 수 있으므로, 이러한 가정은 일반성을 잃지 않는다. 예를 들어, 스트링 데이타의 경우에는 Fox 등[8]이 제안한 순서가 보장되는(order-preserving) 해쉬 함수를 사용하여 실수 값으로 변환될 수 있다.

데이타스트림의 입력 데이타는 튜플의 속성 값 이외에도 타임스탬프 값을 가진다. 타임스탬프 값은 데이타스트림 시스템에 데이타가 입력된 시간을 의미한다. 이러한 타임스탬프 값을 사용하여 데이타들 간의 입력 순서를 파악할 수 있고, 오래된 데이타를 폐기할 수 있다.

2.2 연속질의의 형태

대부분의 기존의 데이타스트림 시스템들[1,5,25]은 이렇게 관계형 튜플 형태로 표현된 데이타를 대상으로 SPJ(Select-Project-Join)형태의 질의를 연속질의로서 다루고 있다.[1] 본 논문에서도 이와 동일한 형태의 수식(1)과 같은 연속질의를 다룬다.

$$\Pi_{projection\_attribute}\ (\ \sigma_{selection\_predicate}\ (D)\ ) \qquad (1)$$

여기서 D는 데이타스트림 소스들의 집합을 의미한다. *selection_predicate*은 질의의 조건(condition) 이다. 질의 조건은 데이타스트림 소스가 하나만 있는 경우, >, ≥, =, <, ≤의 연산자로 이루어진 선택(selection)조건의 논리곱들의 논리합(disjunction of conjunctions), 즉, DNF(Disjunctive Normal Form)로 표현된다. 모든 질의 조건은 DNF로 변환될 수 있기 때문에, 이러한 질의 표현은 일반성을 잃지 않는다. 데이타스트림 소스가 두 개 이상인 경우, *selection_predicate*은 소스들간의 조인(join) 조건을 포함할 수 있다. 여기서 조인 조건은 >, ≥, =, <, ≤의 연산자로 이루어진 임의의 쎄타 조인일 수 있다. 그리고, *projection_attribute*는 질의 결과로 반환되는 데이타 속성 이름들의 리스트이다.

데이타스트림에서는 데이타가 무한히(unbounded) 입력된다는 성격을 가지기 때문에 질의에 데이타의 범위를 한정하는 **슬라이딩 윈도우** 개념을 사용한다[1,9,16]. 슬라이딩 윈도우는 시간-기반 슬라이딩 윈도우(time-based sliding window)와 개수-기반 슬라이딩 윈도우(count-based sliding window)로 나누어진다. 시간-기반 슬라이딩 윈도우는 그 크기가 w로 주어졌을 때 최근 w시간 안에 도착한 데이타만을 질의의 대상으로 한정한다. 개수-기반 슬라이딩 윈도우는 최근 도착한 w개의 데이타만을 질의의 대상으로 한정한다.

2.3 연속질의의 처리 형태

연속질의를 처리하는 방법은 질의 수행 시점에 따라서 즉각적인(immediate) 처리 방법과 배치(batch) 처리 방법으로 나누어진다. 즉각적인 처리 방법은 데이타스트림으로부터 데이타 엘리먼트가 입력될 때 마다 연속질의를 처리하여 결과를 돌려주는 방법이다. 배치 처리 방법은 일정 기간 동안 모았다가 한꺼번에 연속질의를 처리하는 방법이다. 본 논문에서는 이러한 두 가지 형태의 연속질의를 모두 다룬다.

즉각적인 처리 방법은 센서 감시 등과 같은 실시간 모니터링이 필요한 응용에서 주로 사용된다. 이 방법은 데이타가 입력되자 마자 지연(delay)없이 바로 질의 결과를 얻을 수 있다는 장점이 있지만, 입력 데이타에 대

1) 데이타스트림에서의 집계 함수 처리는 데이타가 무한히(unbounded) 입력되는 성질로 인하여 통계 정보 등을 사용하여 근사치를 구하는 방법으로 수행된다[1,9]. 이러한 집계 함수의 처리는 본 논문에서 다루는 기본적인 SPJ 형태의 질의를 수행한 후에 후처리로서 수행할 수 있으며, 추후 향후 연구에서 다루도록 한다.

해서 매번 질의 처리를 수행해야 하기 때문에 데이타가 매우 빨리 입력되는 환경에서는 사용하기 어렵다는 단점이 있다.

배치 처리 방법은 교통량 분석 등과 같은 주기적인 모니터링이 필요한 응용에서 주로 사용된다. 이 방법은 입력된 데이타를 모았다가 한꺼번에 처리하기 때문에 매번 질의 처리를 수행하는 즉각적인 처리 방법에 비해서 단위 시간당 더 많은 데이타에 대해서 연속질의를 수행할 수 있다는 장점이 있지만, 데이타가 입력되었을 때 부터 질의 결과를 얻을 때 까지 지연이 있기 때문에 실시간성이 중요한 응용에서는 사용하기 어렵다는 단점이 있다.

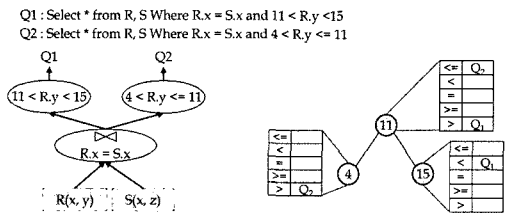
3. 관련 연구

NiagaraCQ[5], TelegraphCQ[4], Aurora[25], 그리고 STREAM[1] 등과 같은 기존 데이타스트림 시스템에서는 연속질의를 처리하기 위하여 데이타-이니셔티브 방법을 취한다. 즉, 데이타스트림 소스로부터 데이타가 새로 도착하였을 때 저장된 연속질의들의 집합에 대해서 그 데이타를 결과로 갖는 질의들을 검색한다. 이러한 시스템들은 단지 질의 집합만을 고려하여 최적화를 수행한다. TelegraphCQ[4] 에서도 데이타 집합과 질의 집합에 각각 색인을 구성하기 때문에 우리가 제안한 방법과 유사해 보이지만, 두 방법 사이에는 큰 차이가 있다. TelegraphCQ는 새로운 질의가 입력되었을 때는 질의-이니셔티브 방법을 사용하고 새로운 데이타가 입력되었을 때는 데이타-이니셔티브 방법을 사용한다. 즉, 두개의 색인을 각각 사용할 뿐이지 함께 사용하지는 않는다.

Golab 등[9]은 기존의 데이타스트림 시스템에서 데이타-이니셔티브 방법으로 연속질의를 처리할 때 사용하는 최적화 기법들을 수행계획 공유 방법(query plan sharing)[4,5]과 프레디킷 색인 방법(query predicate indexing)[5]으로 분류하였다. 질의 수행계획 공유 방법은 질의들의 수행계획에서 공통된 연산을 공유하는 방법이다. 질의 프레디킷 색인 방법은 질의의 조건들을 색인하여 데이타를 만족하는 질의를 빠르게 찾는 방법으로, IBS(Interval Binary Search) 트리[11]에서와 같이 질의 조건을 만족하도록 하는 속성값의 범위를 색인한다.

3.1 질의 수행계획 공유(query plan sharing)

질의 수행계획 공유 방법[4,5]은 등록된 질의들의 공통된 연산을 식별하고, 이러한 공통된 연산을 한번만 수행하는 방법이다. 따라서, 질의들 간에 공통된 연산들을 중복 수행하는 것을 제거할 수 있다. 데이타스트림 시스템에서는 주로 수행 비용이 큰 조인 연산의 수행을 공유하기 위하여 이 방법을 사용한다.



(a) 질의 Q1과 Q2의 수행계획 공유의 예 (b) 속성 R.y에 대한 질의 프레디킷 색인의 예

그림 3 데이타-이니셔티브 방법에서의 질의 최적화 기법

그림 3(a)는 질의 Q1과 Q2의 수행계획을 공유하는 예이다. 질의 Q1과 Q2는 조인 연산 R.x = S.y를 공통으로 가지고 있으므로, 이를 한번만 수행하도록 질의 수행 계획을 공유함으로써 질의를 빠르게 처리할 수 있다.

3.2 질의 프레디킷 색인(query predicate indexing)

질의 프레디킷 색인 방법[5]은 주어진 데이타 엘리먼트에 의해서 조건이 참이되는 질의들을 빠르게 찾는 방법이다. NiagaraCQ[5]와 TelegraphCQ[4]에서는 질의 프레디킷 색인 방법으로 IBS(Interval Binary Search) 트리[11]의 변형을 사용하고 있다. IBS 트리는 프레디킷에 나타난 각 속성들 별로 구성되며, 프레디킷의 상수 값들을 노드로 하는 균형(balanced) 이진 탐색 트리이다. IBS 트리의 각 노드에는 프레디킷에 사용된 상수 값과, 그 상수 값에 사용된 프레디킷들의 식별자가 >, ≥, =, <, ≤ 연산자 별로 저장되어 있다.

그림 3(b)는 질의 Q1과 Q2에서 속성 R.y에 대한 프레디킷을 IBS 트리를 사용하여 색인한 예이다. 데이타가 주어지면 IBS 트리를 해당 속성 값으로 탐색(traverse)하면서 연산자를 비교하는 방법으로 참이되는 프레디킷들을 찾는다. 예를들어 R.y의 값이 14인 데이타 엘리먼트가 입력되면, 노드 11에서는 입력된 데이타가 11보다 크기 때문에 Q1이 조건을 만족하는 프레디킷으로 선택된다. 그리고, 노드 15에서는 입력된 데이타가 15보다 작기 때문에 Q1이 선택된다. 결과적으로 저장된 모든 노드에서 Q1이 선택되었으므로, R.y = 14인 데이타 엘리먼트는 Q1의 R.y 속성에 대한 조건을 만족한다.

IBS 트리를 사용하는 방법은 데이타의 속성별로 색인을 구성하기 때문에 연속질의 처리시 여러 개의 IBS 트리를 탐색해야하며, 최종 결과를 얻기 위하여 각각의 색인으로부터 얻은 결과를 병합하는 과정을 거쳐야 한다. 그리고, 조인 조건과 같이 두개 이상의 속성들간의 관계로 표현된 프레디킷은 색인할 수 없다는 단점이 있다.

4. 데이타와 질의의 이원성 모델

본 장에서는 연속질의 처리를 위한 **이원성 모델**을 제안한다. 제4.1절에서는 이원성 모델에서의 데이타와 질

의의 정의에 대해서 설명한 후, 제4.2절에서는 질의 처리에 대해서 설명한다.

4.1 데이타 엘리먼트와 질의의 정의

이원성 모델에서는 질의와 데이타가 다차원 공간상의 영역으로 동일하게 표현된다. 정의 1과 정의 2에서 데이타 엘리먼트와 질의를 각각 다차원 공간상의 점(point)과 영역(region)으로 정의한다. 점은 영역의 특수한 형태이므로, 앞으로는 혼동의 우려가 없으면 점 까지도 영역이라고 부른다.

정의 1. (이원성 모델에서의 데이타 엘리먼트) n개의 속성을 가지고 i번째 속성의 도메인이 Dom_i인 데이타 엘리먼트 $d = (v_1, v_2, ..., v_n)$는 도메인 공간 $Dom_1 \times Dom_2 \times ... \times Dom_n$ 에서의 점 $(v_1, v_2, ..., v_n)$으로 정의한다. 즉, 데이타 엘리먼트 d는 각 속성들을 차원축으로 하는 n차원 공간에서 각 축마다 해당 속성 값을 좌표 값으로 하는 점이다. 그리고, 데이타 엘리먼트들의 집합을 $\mathbb{D}$로 표시한다. □

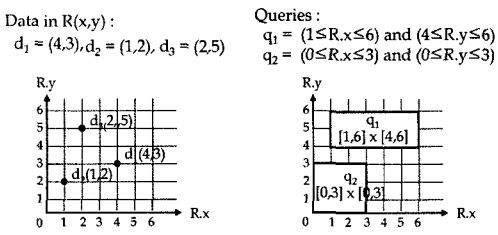
정의 2. (이원성 모델에서의 질의) 질의 q가 $\phi_q(attr_1, attr_2, ..., attr_n)$와 같이 n개의 속성에 대한 조건으로 주어지고 i번째 속성의 도메인이 Dom_i 일 때, 질의 q는 도메인 공간 $Dom_1 \times Dom_2 \times ... \times Dom_n$ 에서의 영역 $\{(v_1, v_2, ..., v_n) \mid \phi_q(v_1, v_2, ..., v_n), v_1 \in Dom_1, ..., v_n \in Dom_n\}$ 으로 정의한다. 즉, 질의 q는 질의의 조건을 만족시키는 n차원 공간상의 점들의 집합으로 이루어진 영역이다.[2] 그리고, 질의들의 집합을 $\mathbb{Q}$로 표시한다. □

예 1: 그림 4는 이원성 모델에서의 데이타와 질의를 다차원 공간에 표현한 예이다. 릴레이션 R(x,y)의 튜플(tuple)인 데이타 엘리먼트 d_1, d_2, d_3는 그림 4와 같이 속성 x와 y를 좌표축으로 하는 이차원 공간상의 점으로 표현된다. 예를 들어, 데이타 엘리먼트 d_1은 x 좌표 값이 4이고, y 좌표 값이 3인 점으로 표현된다. 릴레이션 R(x,y)에 대한 질의 q_1과 q_2는 그림 4(b)와 같이 이차원 공간상의 영역으로 표현된다. 예를 들어, 질의 q_1은 x축에서 [1,6]의 범위이고 y 축에서 [4,6]의 범위인 영역으로 표현된다. □

4.2 질의 처리

이원성 모델에서의 질의 처리는 다차원 공간에서 서로 겹치는 데이타 엘리먼트와 질의의 쌍을 찾아냄으로써 질의 조건을 만족하는 데이타를 찾아낼 수 있으며, 보조정리 1은 이러한 사실을 증명한다.

보조정리 1. 이원성 모델에서 데이타 집합 $\mathbb{D}$와 질의 집합 $\mathbb{Q}$가 주어질 때, 데이타 엘리먼트 $d \in \mathbb{D}$와 질의 $q \in \mathbb{Q}$인 사이에 겹침이 있으면, d는 q의 조건을 참이 되게 한다.



(a) 공간상에 표현된 데이타 엘리먼트 (b)공간상에 표현된 질의

그림 4 이원성 모델에서의 데이타 엘리먼트와 질의

증명: 정의 1과 2에 의해서 데이타 엘리먼트 d와 질의 q는 n차원 공간에서 $(v'_1, v'_2, ..., v'_n)$인 점과 $\{(v_1, v_2, ..., v_n) \mid \phi_q(v_1, v_2, ..., v_n) = \text{true}\}$인 영역으로 표현된다. 이때, 데이타를 표현하는 점과 질의를 표현하는 영역이 겹치는 관계에 있다는 것은 정의 2에 의해서 $\phi_q(v'_1, v'_2, ..., v'_n)$이 참이 됨을 의미한다. 따라서, 데이타 엘리먼트 d와 질의 q간에 겹침이 있으면, 데이타 엘리먼트 d는 질의 q의 조건을 만족한다. □

예 2: 그림 4에서 데이타 엘리먼트 d_2와 질의 q_2는 겹치며, 이때 데이타 엘리먼트 d_2의 값 (1,2)는 질의 q_2의 조건 $(0 \le R.x \le 3)$ and $(0 \le R.y \le 3)$을 만족한다. 데이타 엘리먼트 d_3와 질의 q_1의 경우에도 마찬가지이다. 하지만, 데이타 엘리먼트 d_1의 경우는 겹치는 질의가 없기 때문에, 어떠한 질의의 조건도 만족하지 않는다. □

5. 공간조인 기반 연속질의 처리 알고리즘

본 장에서는 데이타와 질의의 이원성 모델을 이용하여 *Spatial Join CQ*라는 이름의 공간조인 기반 연속질의 처리 알고리즘을 제안한다. 알고리즘을 점진적으로 설명하기 위하여 본 장에서는 단순 선택 연속질의 처리를 설명하고, 제6장에서 이를 슬라이딩 윈도우 조인 연속질의 처리로 확장한다. 제5.1절에서는 제안하는 알고리즘의 특징을 간단히 설명한다. 그리고, 제5.2절에서는 제안하는 알고리즘에서 사용하는 다차원 공간조인 방법인 변환 기반 공간조인 방법에 대해서 설명한다. 제5.3절과 제5.4절에서는 공간조인 수행에서 사용되는 데이타 색인과 질의 색인을 구성하는 방법에 대해서 설명한다. 마지막으로 제5.5절에서는 다차원 공간조인을 사용하여 연속질의를 처리하는 알고리즘을 제시한다.

5.1 특징

제안하는 알고리즘에서는 서로 겹치는 데이타와 질의의 쌍을 찾기 위하여 다차원 공간조인을 사용한다. 다차

2) 질의 조건이 $C_1 \vee C_2 \vee \cdots \vee C_n$으로 주어졌을 때($C_i$는 프레디킷들의 논리곱), 각각의 C_i는 하나의 영역으로 표현된다. 따라서, 하나의 질의 조건은 둘 이상의 영역의 집합으로 표현될 수 있다. 본 논문에서는 설명의 편의를 위해 하나의 C_i만을 조건으로 갖는 질의를 고려한다. 여러 개의 C_i들을 가지는 경우는, 각각의 C_i들의 결과를 ORing함으로써 이 방법을 쉽게 확장할 수 있다.

원 공간조인은 두 공간에 존재하는 공간 객체들간에 특정 공간 관계(본 논문에서는 서로 겹치는 관계)를 만족하는 모든 쌍을 찾는 공간 데이타베이스 연산이다[3]. 다차원 공간조인을 위하여 현재까지 많은 알고리즘이 연구되었으며, 본 논문에서는 구석점 변환 기반 공간조인 알고리즘[20]을 사용한다.

제안하는 알고리즘은 서로 이원적인 데이타-이니셔티브와 질의-이니셔티브 방법을 하나의 알고리즘으로 통합한다. 따라서, 이 알고리즘은 **여러 개**의 데이타 엘리먼트들과 각각 겹치는 **여러 개**의 질의들을 동시에 찾을 수 있다. 이러한 효과를 위하여 제안하는 알고리즘은 데이타스트림으로부터 입력된 데이타 엘리먼트들을 일정 개수 모아서 데이타 집합을 구성하여 배치 방법으로 연속질의를 처리한다. 그리고, 즉각적인 연속질의 처리는 배치 처리에서 데이타 집합에 엘리먼트가 하나만 존재하는 특수한 경우로 하여 수행한다. 제안하는 알고리즘은 배치 처리를 위하여 데이타 집합에서 인접한 여러 개의 데이타 엘리먼트들을 최소 포함 사각형(Minimum Bounding Rectangle: MBR)으로 묶어서 질의들과 공간조인을 수행한다. 앞으로 이러한 데이타 엘리먼트들의 MBR을 **데이타 클러스터**(*data cluster*)라 부른다. (즉각 처리의 경우는 데이타 엘리먼트 그 자체가 데이타 클러스터가 된다.) 인접한 데이타 엘리먼트들은 동일한 질의들과 겹치게 될 확율이 높으므로, 이것들을 MBR로 묶어서 공간 조인하면 여러 개의 데이타 엘리먼트와 겹치게 되는 질의를 각 데이타 엘리먼트마다 액세스 하는 것을 피할 수 있다는 장점이 있다.

공간조인 기반 연속질의 처리 알고리즘은 그림 5와 같이 (1) 데이타와 질의에 대해서 색인을 구성하는 과정, (2) 생성된 색인을 사용하여 공간조인을 수행하여 질의 결과 후보 집합을 찾는 과정, 그리고 (3) 후보 집합을 정제(refine)하여 최종 결과를 얻는 과정의 3단계로 이루어진다. 그림 5에서 데이타 공간과 질의 공간은 각각 데이타 엘리먼트들을 위한 도메인 공간과 질의들을 위한 도메인 공간을 의미한다.

첫 번째 단계는 **색인 생성 단계**이다. 이 단계에서는

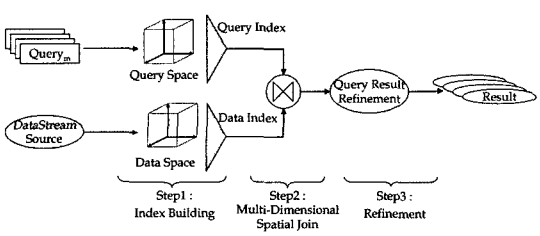


그림 5 공간조인 기반 연속질의 처리 알고리즘의 3단계 수행 과정

데이타 엘리먼트들의 집합과 질의들의 집합에 대해서 각각 색인을 생성한다. 데이타 색인은 입력된 데이타 엘리먼트를 표현하는 점들을 간단한 일차원 색인 구조에 저장하고, 질의 색인은 연속질의를 나타내는 영역을 다차원 색인 구조에 저장한다. 이때, 질의 색인에서는 질의가 초월 사각형(hyper-rectangle) 형태가 아닌 복잡한 형태로 나타날 때는 이 영역을 포함하는 MBR로 표현하여 다차원 색인 구조에 저장한다. 데이타 색인과 질의 색인에 대한 자세한 방법은 제 5.3절과 제 5.4절에서 설명한다.

두 번째 단계는 **다차원 공간조인 단계**이다. 이 단계에서는 데이타 색인과 질의 색인을 사용하여 다차원 공간조인을 수행함으로써 **후보 결과 쌍**(*candidate result pair*)들을 구한다. 질의를 표현한 영역의 형태가 초월 사각형이 아닌 경우는 질의 색인 과정에서 MBR을 사용하여 색인하기 때문에 공간조인의 결과로 착오 해답(false alarm)이 나올 수 있다. 다차원 공간조인에 대한 자세한 방법은 제5.5절에서 설명한다.

마지막 단계는 질의 **결과 정제 단계**이다. 이 단계에서는 공간조인 단계의 결과로 생성된 후보 결과 쌍들에 대해서 실제로 데이타 엘리먼트가 질의 조건을 만족하는지의 여부를 검사하여 최종 결과 쌍을 생성한다.

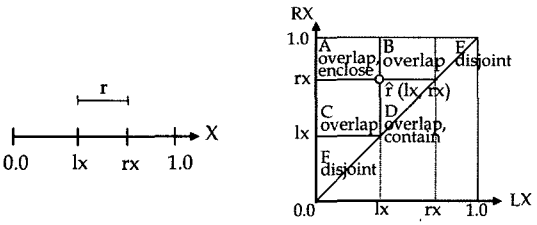
5.2 구석점 변환 기반 공간조인 알고리즘

본 절에서는 Song등[20]이 제안한 구석점 변환 기반 공간조인 알고리즘을 소개한다. 구석점 변환 기반 공간조인 알고리즘은 공간 객체를 구석점 변환 기법[18]을 사용하여 점으로 변환한 후 조인을 수행한다. 이 알고리즘은 크기가 없는 점만을 다루므로 적은 비용으로 전역 최적화가 가능하다. 이로 인해, 이 알고리즘은 R-tree기반의 원공간 공간조인 알고리즘[3,13]에 비해 우수한 성능을 보인다.

구석점 변환 기법은 n차원 원공간의 객체의 MBR을 $2n$차원의 변환공간의 점 객체로 변환하는 방법이다. 구석점 변환 기법은 각 차원 축에 대한 MBR의 최소값과 최대값을 사용하여 변환공간의 점 객체의 좌표값을 구성한다[12,18]. 예를 들어, 일차원 원공간에서 공간 객체의 최소값과 최대값이 각각 lx와 rx인 공간 객체는 이차원 변환공간상의 점 객체 (lx,rx)로 변환된다.

원공간에서 특정 공간 객체와 겹치는 공간 객체들을 찾는 연산은 변환공간에서는 특정 영역 안의 모든 점 객체들을 찾는 연산으로 변환하여 처리된다[18]. 그림 6은 변환공간에서 하나의 점과 영역들 사이의 관계를 나타낸 것이다. 영역 A내의 모든 객체들은 최소값 LX가 lx보다 크고, 최대값 RX는 rx보다 작기 때문에 r을 포함한다. 반대로, 영역 D내의 모든 객체들은 LX값이 lx보다 크고, RX값은 rx보다 작기 때문에 r에 포함된다.

영역 B와 C에 속한 객체들은 각각 LX값이 lx보다 크고 RX값이 rx보다 작기 때문에 좌측점과 우측점을 포함한다. 그러나, 영역 E와 F에 속한 객체들은 각각 RX가 lx보다 작거나 LX가 rx보다 크기 때문에 r과 겹치는 부분이 없다. 따라서, 원공간 상에서 영역 r과 겹치는 영역들은 변환공간상의 영역 A, B, C, D내의 점으로 변환된다[18]. 이러한 특성을 사용하여 영역 r과 겹치는 공간 객체들을 찾는 공간조인 연산은 변환공간에서 A, B, C, D를 합한 색칠된 영역 안에 들어 있는 점 객체들을 찾는 연산으로 수행할 수 있다.



(a) 원공간 상의 영역 r (b) 영역 r에 대한 변환공간에서의 관계

그림 6 원공간상의 주어진 영역과 공간 관계에 대응되는 변환공간상의 영역들

다차원 공간 조인을 수행하는데 있어서는 **공간조인 윈도우**(*Spatial Join Window*)라는 개념을 사용하고 이를 정의 3에서 정형적으로 정의한다.

정의 3. (**공간조인 윈도우**) [20] 서로 공간조인되는 색인 R과 S의 변환공간을 TS(R)과 TS(S)라 하자. 이때, TS(R)에 있는 사각형 영역 P에 대한 공간조인 윈도우 SJW(P)는 영역 P에 포함될 수 있는 모든 객체들과 겹치는 객체들이 존재하는 TS(S)내의 최소 영역이다. □

그림 7의 짙게 색칠된 영역 P에 대한 SJW(P)는 다음의 과정으로 구해진다. 설명의 편의를 위하여 본 논문에서는 이차원 변환공간에서의 SJW만을 설명한다. $2n$차원 변환공간에서의 SJW는 Song등의 연구[20]에 자세히 기술되어 있다. P의 임의의 객체와 겹치기 위해서는, P의 좌상점과 대응되는 객체 $\hat{s} = (lx,rx)$와 반드시 겹쳐야 한다. 이는 $\hat{s}$이 P의 모든 객체를 포함하는 가장 큰 객체이기 때문이다. $\hat{s}$과 교차하는 모든 객체들이 존재하는 TS(S)내의 최소 영역, 즉 SJW(P)는 보조정리 2에 따라 $[0,rx] \times [lx,1]$에서 대각선 위 부분이다. 즉, SJW(P)는 그림 7에서 옅게 색칠된 영역이 된다. 이때 대각선 아래 부분에는 객체가 존재할 수 없으므로, 지금부터는 빗금친 영역을 포함하여 $[0,rx] \times [lx,1]$를 SJW로 사용한다.

보조정리 2. [20] TS(R)내의 원공간 객체 $\hat{s} = (lx,rx)$와 원공간에서 겹치는 모든 객체들이 존재하는 TS(S)내의

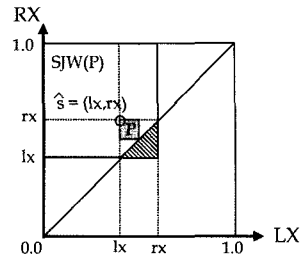
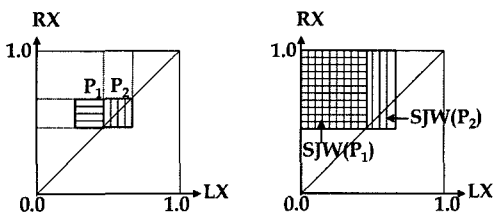


그림 7 공간조인 윈도우(SJW)

최소 영역은 $[0,rx] \times [lx,1]$에서 대각선 위 부분이다.

증명: Song등의 연구[20] 참고. □

보조정리 2로 부터, TS(R)내의 서로 인접한 두 영역에 대한 TS(S)에서의 두 공간조인 윈도우 사이에는 큰 겹침이 존재함을 알 수 있다. 그림 8은 인접한 두 영역 P_1과 P_2의 SJW간에 많은 겹침이 있음을 보여주는 예이다. 우리는 이러한 성질을 **SJW들간의 겹침 성질**(줄여서 **겹침 성질**)이라고 부른다. 따라서, TS(R)에서 서로 인접한 영역 P_i와 P_j 순서로 다차원 공간 조인을 수행하면 SJW(P_j)의 대부분이 SJW(P_i)를 읽어들이면서 이미 버퍼에 읽어들였기 때문에 다시 읽을 필요가 없어서 적은 수의 I/O로 조인을 처리할 수 있다. 여기서 버퍼 교체 정책(replacement strategy)은 LRU를 사용한다.



(a) 인접한 두 영역 P1과 P2 (b) SJW(P1)과 SJW(P2)

그림 8 인접한 영역들간의 공간조인 윈도우들

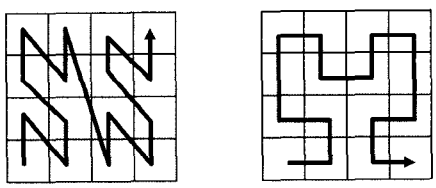
5.3 데이타 색인 과정

데이타 색인은 배치 연속질의 처리에서 입력된 데이타 엘리먼트들을 모으고 이것들을 인접한 순서로 읽어들이기 위해 사용된다. 데이타 엘리먼트들을 인접한 순서로 읽어오게되면 데이타 클러스터 P를 인접한 순서로 생성함으로써 SJW(P)간의 **겹침 성질**을 이용할 수 있다. 생성된 데이타 색인은 배치 연속질의 처리가 끝나면 폐기되고, 다음 배치 처리때 까지 입력되는 데이타 엘리먼트들로 다시 구성된다. 즉각적인 연속질의 처리에서는 데이타 색인은 사용되지 않는다.

데이타의 색인을 위해서는 다차원 색인 구조를 사용하지 않고 일차원 색인 구조를 사용한다. 데이타 색인은

도메인 공간에서 데이타 엘리먼트들을 근접한(proximity) 순서로 액세스 하는 기능만을 제공하면 되기 때문이다.

일차원 색인 구조에 다차원 공간 상에서 근접해 있는 순으로 데이타 엘리먼트를 저장하기 위해서 공간 채움 곡선[6,17]을 사용한다. 공간 채움 곡선은 다차원 공간 상에서 인접한 부분 영역들을 일차원 순서에서도 높은 확률로 인접하도록 순서화하는 방법이다[6,17]. 대표적인 공간 채움 곡선 방법으로는 Z-순서화(z-ordering)[17]와 힐버트 순서화(Hilbert Ordering)[6] 등이 있다. 그림 9(a)와 9(b)는 2차원 공간에서의 각각의 예를 표현한 것이다.[3] 본 논문에서는 파티션된 부분 영역들을 읽을때 한번에 한 차원의 값만을 변화시킴으로써 높은 근접성을 보장하는 힐버트 순서화를 사용한다. 힐버트 순서화에 의하여 정렬된 순서로 데이타 엘리먼트들을 저장하기 위해서는 이진 탐색 트리 구조(binary search tree)를 사용한다.



(a) Z-순서화 (b) 힐버트 순서화

그림 9 대표적인 공간 채움 곡선

5.4 질의 색인 과정

질의 색인은 다차원 공간 조인 수행시 데이타 공간의 영역 P가 주어졌을 때, 질의 공간에서 SJW(P)에 속한 질의들을 빠르게 읽어오기 위해서 사용된다. 질의의 색인을 위해서는 R-Tree[10], Quad Tree[7], Buddy Tree[19], MLGF[23,24] 등의 다차원 색인 구조를 사용할 수 있다.

질의 색인에 질의를 하나 추가하는 과정은 다음과 같다. 우선 입력된 질의를 정의 2에 따라 영역으로 변환하고 이 영역이 초월 사각형 형태가 아니면 MBR로 근사하여 표현한다. 다음으로 이 MBR은 구석점 변환 기법을 통하여 점으로 변환하고 다차원 공간 색인에 입력한다. 각주 2에서 기술한 바와 같이 논리합 조건을 가지는 질의는 여러 개의 영역들의 집합으로 나타난다. 이때, 위에서 기술한 질의 색인 방법은 동일한 질의에 속한 영역들 각각에 적용된다.

예 3: 그림 10은 질의를 표현한 영역을 다차원 색인 구조에 저장하는 예이다. 질의 q의 조건은 데이타의 두 개의 속성 x와 y가 [0.2,0.5]의 범위이고 x가 y보다 크다는 조건을 가진다. 질의 q는 x축과 y축에서 [0.2,0.5]의 범위를 가지는 사각형 영역에서 대각선 아래 부분에 해당하는 삼각형 영역 R_q로 표현된다. 이러한 영역은 그대로 색인될 수 없기 때문에 최소포함 사각형인 MBR_q로 근사된다. 그리고, MBR_q는 구석점 변환 기법을 사용하여 4차원 공간상의 점 (0.2, 0.5, 0.2, 0.5)으로 변환되어 다차원 색인 구조에 입력된다. □

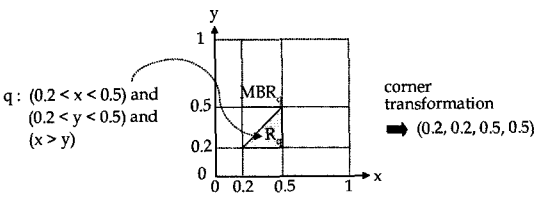


그림 10 질의 색인 예

질의 색인 알고리즘은 질의를 표현한 영역을 점으로 변환하여 색인함으로써 크기가 큰 영역을 효율적으로 색인할 수 있다. 질의는 조건으로 열린 간격(open interval)을 가질 수 있고, 속성에 조건이 주어지지 않을 경우 해당 차원의 전체 구간을 영역으로 가지기 때문에 원공간 상에서는 크기가 큰 영역으로 표현될 수 있다. 크기가 큰 공간 객체들은 서로 겹침이 발생할 가능성이 높다. 이런 경우 크기가 있는 객체들을 R-Tree와 같은 다차원 색인 구조에 그대로 색인하면 색인의 비단말 노드들 사이의 겹침이 크게 증가하는 현상이 발생하기 때문에 색인의 크기가 커지고 검색 성능이 저하된다[11]. 그러나, 본 방법에서는 점으로 변환하여 색인하므로 이러한 영역들간의 겹침 문제는 원천적으로 발생하지 않는다.

다차원 색인 구조는 차원의 수가 증가할수록 성능이 저하되는 문제(dimensionality curse)[22]를 가지고 있으나, 이는 다음의 두 가지 이유 때문에 우리의 알고리즘에서는 큰 문제가 되지 않는다. 먼저, 질의 색인은 조건이 주어진 속성들 만을 다루는데 실제 데이타베이스 응용에서 대부분의 질의는 하나 혹은 두개의 속성에 대해서만 조건이 주어지고 5개 이상의 속성에 조건이 주어지는 경우는 매우 드물다[11]. 그리고, 예외적인 경우(예를들어 질의에 나타나는 속성의 수가 10개 이상인 경우) Pyramid-technique[2]과 같은 고차원에서의 성능 저하 문제(dimensionality curse)를 최소화하는 기존의 연구들을 적용할 수 있다.

5.5 공간조인을 사용한 연속질의 처리 방법

다차원 공간조인 단계에서는 데이타 엘리먼트들을 도

3) 공간 채움 곡선은 동일한 패턴을 각 차원에 대해서 재귀적으로 적용함으로써 3차원 이상의 공간에서도 사용될 수 있다[6,17].

메인 공간에서 인접한 순서로 읽어와서 영역 P를 구성하고, 질의 공간의 SJW(P)에 속한 질의들을 읽어온다. 이때, 전자의 과정에서는 데이타 색인을 사용하고, 후자에서는 질의 색인을 사용한다. 그리고, 영역 P에 속한 데이타 엘리먼트들과 SJW(P)에 속한 질의들로 부터 서로 겹치는 쌍을 찾는다.

영역 P는 앞에서 설명한 데이타 클러스터와 대응된다. 데이타 클러스터는 공간적으로 인접한 순서로 읽어온 일정 개수의 데이타 엘리먼트들의 MBR이다. 데이타 클러스터를 구성할 때, 데이타 클러스터의 크기가 과도하게 커지거나 작아지는 경우를 막기 위해 크기를 제한한다. 데이타 클러스터의 크기가 너무 큰 경우는 SJW 역시 커지게 되므로 질의 색인의 많은 부분을 읽어야 하는 문제가 발생한다. 반대로, 데이타 클러스터의 크기가 너무 작은 경우는 배치 처리로 인한 성능상의 이점이 사라지게 된다. 따라서, 데이타 클러스터의 크기는 연속질의 처리 성능을 튜닝하는데 중요한 파라미터이다. 최적의 데이타 클러스터 크기는 데이타와 질의의 개수와 분포에 따라 달라지기 때문에 분석적으로 구하기가 어려우므로 제7.2절에서는 실험을 통하여 이를 찾기로 한다.

그림 11은 이와같이 다차원 공간조인으로 연속질의를 처리하는 Spatial Join CQ 알고리즘을 보여준다. 알고리즘의 첫번째 과정(lines 1~3)과 두번째 과정(lines 4~5)은 제5.4절과 제5.3절에서 설명한 바와 같이 질의 색인과 데이타 색인을 구성하는 과정이다. 세번째 과정(lines 6~14)에서는 연속질의를 처리하기 위하여 다차원 공간조인을 수행한다. 배치 처리의 경우, 다차원 공간 조인은 N_{batch_data}개의 데이타 엘리먼트들이 모일 때마다 수행되고 즉각 처리의 경우, 하나의 데이타 엘리먼트가 입력되었을 때마다 수행된다. 다차원 공간 조인은 다음과 같은 과정을 $\lceil N_{batch\_data} / N_{dc} \rceil$ 번 수행한다. (즉각 처리의 경우는 N_{batch_data}=1 이고, N_{dc}=1이다.) 먼저, 데이타 색인을 사용하여 N_{dc}개의 데이타 엘리먼트들을 인접한 순서로 읽어와서 데이타 클러스터를 구성한다(lines 6~8). 다음으로, 데이타 클러스터의 SJW에 속한 질의들을 질의 색인으로부터 읽어와서(line 9) 겹치는 데이타와 질의 원소의 쌍을 찾는다(line 10). 공간 조인 후에는 결과로 얻은 원소쌍들을 정제하여 최종 결과를 얻는다(line 11). 데이타 색인은 다차원 공간 조인 수행이 끝나면 폐기되고(line 13), 다음 데이타 엘리먼트가 새로 입력될 때 부터 다시 재구성된다.

예 4: 그림 12는 배치 처리 방법으로 연속질의를 수행하는 방법의 예이다. 그림 12의 데이타 공간은 하나의 속성 x를 가지는 데이타 엘리먼트 여섯 개가 변환공간상에 존재하는 경우에 이것을 각각 세 개씩 묶어서 데이타 클러스터를 구성한 것이다. 그림 12의 질의 공간은 q_1부터 q_6까지의 여섯 개의 질의가 존재하는 경우에 각각의 데이타 클러스터와 공간조인되는 SJW를 나타낸다. 데이타 클러스터1은 질의 q_1, q_2, q_3와 공간조인되고, 데이타 클러스터2는 질의 q_3, q_4, q_5와 공간조인된다. □

Algorithm Spatial Join CQ

Input: A newly arriving data element or a newly registered query
Output: Data element-query pairs that satisfy conditions
Algorithm:
/* N_{batch_data} is the number of data elements in a batch */
/* N_{dc} is the number of data elements in a data cluster */
/* for immediate processing, N_{batch_data} = 1; N_{dc} = 1 */
01: **if** a query is newly registered **then**
02: Insert the query into the query index (as described in Section 5.4);
03: **end if**
04: **if** a data element newly arrives **then**
05: Insert the data element into the data index (as described in Section 5.3);
06: **if** the number of data elements in the data index is N_{batch_data} **then**
07: ***repeat by*** $\lceil N_{batch\_data} / N_{dc} \rceil$ ***times***
08: Construct a data cluster *DC* containing N_{dc} data elements read in proximity(Hilbert) order using the data index; (For immediate processing, *DC* consists of only one newly arriving data element)
09: Retrieve the queries contained in *SJW(DC)* using the query index;
10: Find data element-query pairs, from *DC* and *SJW(DC)*, that satisfy the overlapping condition (Lemma 1);
11: Refine the data element-query pairs by checking whether the data element makes the conditions of the query true;
12: **end repeat**
13: Delete all data elements from the data index;
14: **end if**
15: **end if**

그림 11 연속질의 처리 알고리즘 Spatial Join CQ

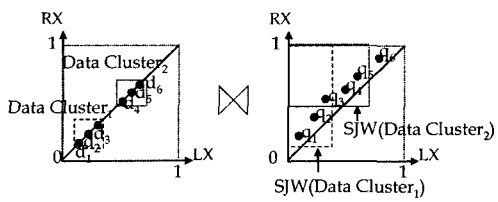


그림 12 복수의 데이타 엘리먼트를 묶어서 처리하는 데이타-질의 다차원 공간조인의 예

6. 슬라이딩 윈도우 조인을 위한 Spatial Join CQ 알고리즘의 확장

본 장에서는 제5장에서 설명한 연속질의 처리 알고리즘을 슬라이딩 윈도우 조인 연속질의 처리를 위해서 확장한다. 제6.1절에서는 슬라이딩 윈도우 조인 연속질의

의 정의를 설명하고, 제6.2절에서는 알고리즘의 확장에 대해서 설명한다.

6.1 슬라이딩 윈도우 조인 연속질의

데이타스트림 소스가 여러 개 있을 경우, 연속질의의 조건으로 데이타스트림 소스들 간의 조인이 주어질 수 있다. 이때, 새로 입력된 데이타 엘리먼트와 조인 관계에 있는 상대 데이타스트림의 슬라이딩 윈도우에 속한 데이타 엘리먼트들간의 조인을 수행한다. 이를 **슬라이딩 윈도우 조인**(*sliding window join*)[1]이라 부른다. 슬라이딩 윈도우 조인의 예로는, 서로 다른 종류의 두 가지 데이타스트림 소스인 온도 센서와 습도 센서가 존재하고 이 센서들이 각각의 지점에 함께 배치되어 있을 때, 최근 5분 동안 온도와 습도가 특정 범위 안에 있었던 지점들을 찾아내는 질의등을 들 수 있다.

그림 13은 데이타스트림 A와 데이타스트림 B에 대한 슬라이딩 윈도우 조인을 수행하는 방법을 나타낸다[14]. 그림 13(a)는 데이타스트림 A로 부터 새로운 데이타 엘리먼트 d_1이 입력되었을 때의 조인 수행 과정이다. 이때, 슬라이딩 윈도우 조인은 (1) 데이타 엘리먼트 d_1과 데이타스트림 B의 슬라이딩 윈도우의 데이타 엘리먼트들 간에 조인 조건을 만족하는지 검사하는 과정(*probe*), (2) 데이타 엘리먼트 d_1을 데이타스트림 A의 슬라이딩 윈도우에 삽입하는 과정(*insert*), 그리고 (3) 슬라이딩 윈도우 범위를 벗어난 데이타를 데이타스트림 A의 슬라이딩 윈도우로부터 삭제하는 과정(*invalidate*)으로 이루어진다. 그림 13(b)는 데이타스트림 B로 부터 새로운 데이타 엘리먼트 d_2가 입력되었을 때의 조인 처리 방법으로, 그림 13(a)와 대칭적인 방법으로 수행됨을 알 수 있다.

6.2 슬라이딩 윈도우 조인을 위한 연속질의 처리 알고리즘

슬라이딩 윈도우 조인을 수행하기 위해서는 (1) 데이타 색인 방법, (2) 질의 색인 방법, 그리고, (3) 다차원 공간조인 방법이 확장되어야 한다. 데이타 색인 방법은 슬라이딩 윈도우 범위에 포함된 데이타만을 유지하도록 확장되어야 하고 질의 색인 방법은 조인 조건 뿐만 아니라 슬라이딩 윈도우 조건도 함께 색인하도록 확장되어야 한다. 그리고, 다차원 공간 조인 방법은 이러한 조건들을 검사하도록 확장되어야 한다.

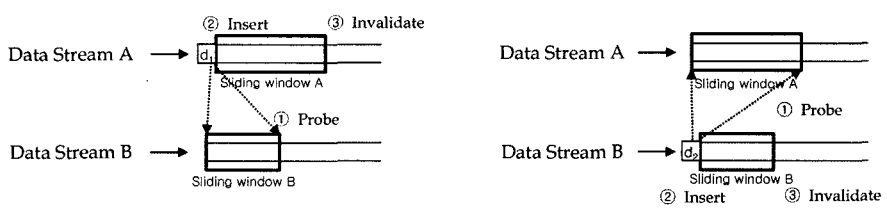
데이타 색인 방법의 확장

데이타 색인은 슬라이딩 윈도우 범위를 만족하는 데이타 엘리먼트들만을 유지하기 위해서 기존의 이진 탐색 트리 구조 이외에 이중 링크드 리스트 구조(doubly linked list)를 추가로 사용한다. 그리고, 데이타 색인은 속성 값 뿐만 아니라 데이타 엘리먼트가 도착한 시간을 나타내는 타임 스탬프값도 함께 저장한다. 이중 링크드 리스트 구조는 트리의 노드들을 입력된 순서로 연결함으로써, 입력된 데이타 엘리먼트들을 시간 순서대로 읽어들일 수 있도록 한다. 데이타 색인에서 이중 링크드 리스트는 슬라이딩 윈도우를 관리하기 위하여 사용된다. 데이타 엘리먼트가 도착하면 우선 이진 탐색 트리 구조에 삽입하고, 이중 링크드 리스트의 헤드(head)에 연결한다. 데이타 엘리먼트의 타임 스탬프가 슬라이딩 윈도우의 범위를 넘어가면, 이중 링크드 리스트의 테일(tail)과 이진 탐색 트리에서 그 데이타 엘리먼트를 제거한다.

질의 색인 방법의 확장

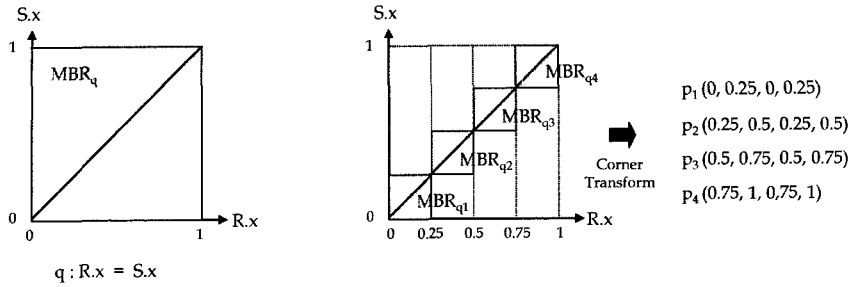
등가 조인(equi-join)인 경우, 조인 조건은 도메인 공간에서의 특정 평면(plane)의 대각선으로 표현된다. 예를 들어, 그림 14(a)와 같이 두 개의 데이타스트림 소스 R, S간에 R.x = S.x 조인 조건이 명시된 질의 q는 R.x와 S.x를 축으로 하는 평면에서 (0, 0)과 (1, 1)을 잇는 직선으로 표현된다. 등가 조인이 아니라 쎄타 조인(theta-join)인 경우는 이러한 대각선을 기준으로 아래의 영역 혹은 위의 영역으로 표현된다. 이러한 영역을 MBR로 근사하면 도메인 공간에서의 특정 평면 전체가 되어 질의 색인에 저장하는 것이 의미 없어진다. 예를 들어, 그림 14(a)에서 조인 조건을 가지는 질의 q의 MBR은 R.x와 S.x를 축으로 하는 평면 전체가 된다.

이러한 문제를 해결하기 위해, 슬라이딩 윈도우 조인 조건은 객체 분할 방법(object decomposition)[15]을 사용하여 색인한다. 객체 분할 방법은 다차원 공간의 특정 차원축을 기준으로 공간 객체를 임의의 개수로 분할한 뒤 각각의 조각에 대해 MBR을 구하는 방법이다. 예를



(a) 데이타스트림 A로 부터 데이타가 입력된 경우 (b) 데이타스트림 B로 부터 데이타가 입력된 경우

그림 13 슬라이딩 윈도우 조인



(a) 조인 조건의 MBR (b) 객체 분할 방법을 사용한 조인 조건의 MBR

그림 14 조인 조건의 색인

들어, 그림 14(b)와 같이 R.x축을 기준으로 질의 q의 영역을 네 개로 분할한 뒤 각각의 분할된 조각에 대해서 MBR을 구성한다. 이와 같이, 객체 분할 방법을 사용하여 여러 개의 MBR들을 색인함으로써 실제 질의를 표현한 영역을 보다 정확하게 색인할 수 있다.

또한, 질의 색인은 질의의 슬라이딩 윈도우 범위 조건도 속성에 대한 조건과 마찬가지 방법으로 다차원 공간상의 한 축의 범위로 저장한다. 다차원 공간조인 과정에서 질의 색인의 슬라이딩 윈도우 조건을 나타내는 축은 데이타 색인의 타임스탬프를 나타내는 축과 공간 조인된다. 이러한 방법을 통하여 서로 다른 슬라이딩 윈도우 크기를 가지는 연속질의들을 하나의 질의 색인에 저장하고, 한번의 공간 조인으로 이 연속질의들을 함께 처리할 수 있다.

이와 같이 제안하는 방법에서는 선택 조건 뿐만 아니라 조인 조건도 질의 색인에 저장할 수 있다. 따라서, 선택 조건과 조인 조건이 모두 명시된 연속질의도 공간조인으로 한번에 처리할 수 있다. 반면, 기존의 질의 색인 방법인 IBS 트리에서는 각각의 속성별로 색인을 구성하기 때문에[11] 속성간의 관계로 표현되는 조인 조건은 색인할 수 없다.

다차원 공간조인 방법의 확장

슬라이딩 윈도우 조인은 제 6.1절에서 설명한 바와 같이 새로 입력된 데이타 엘리먼트와 상대 슬라이딩 윈도우 내의 모든 데이타 엘리먼트들 간에 조인 조건을 만족하는 쌍을 찾아내는 연산이다. 이와 같은 데이타 엘리먼트의 쌍을 **조인 가능한 데이타 쌍**(*joinable data pair*)이라 부르고 정의 4에서 정의한다.

정의 4. (조인 가능한 데이타 쌍들의 집합) 두 개의 데이타스트림 소스 DS_i와 DS_j가 있을 때, DS_i에 데이타 엘리먼트 d_i가 새로 입력되었을 경우의 조인 가능한 데이타 쌍들의 집합 $JD_{DSi,\ DSj}$는 다음과 같이 정의된다. 여기에서 $SW(DS_j)$는 DS_j의 슬라이딩 윈도우 안에 있는 데이타 엘리먼트들의 집합을 의미한다.

$$JD_{DSi,\ DSj} = \{d_i\} \times SW\ (DS_j) \quad \square$$

슬라이딩 윈도우 조인 연속질의를 처리하기 위한 다차원 공간 조인은 다음의 과정으로 수행된다. 우선 데이타스트림 소스 DS_i로 부터 데이타 엘리먼트 d_i가 새로 입력되면 $JD_{DSi,DSj}$를 생성한다. 이때, SW(DS)에 있는 데이타 엘리먼트들을 데이타 색인을 사용하여 도메인 공간에서 인접한 순서(즉, 공간 채움 곡선 순서)로 읽어온다. 이렇게 생성된 조인 가능한 데이타 쌍들을 일정 개수씩 모아서 데이타 클러스터를 구성한다. 그리고, 제 5.5절의 설명한 것과 동일한 방법으로 데이타 클러스터들의 집합과 질의 집합 간의 다차원 공간 조인을 수행하여 질의 결과를 얻는다. 조인 질의를 위한 이 알고리즘은 그림 11의 알고리즘과 비교하여 데이타 엘리먼트 대신 조인 가능한 데이타 쌍을 사용했다는 점을 제외하고는 동일하다.

슬라이딩 윈도우 조인 알고리즘은 다중 조인(multi-way join) 질의에서도 또한 사용될 수 있다. 다중 조인 질의의 경우는 앞에서 설명한 방법과 마찬가지로 $JD_{DSi,\ \cdots,\ DSn}$를 생성하고 질의 색인과 공간 조인한다. 하지만, DS_1, ..., DS_n 간의 카티션 곱으로 인해서 $|JD_{DSi,\ \cdots,\ DSn}|$가 커질 수 있다. 다중 조인을 효율적으로 수행하는 방법은 향후연구에서 다루도록 한다.

7. 성능평가

본 장에서는 제안한 공간조인 기반 연속질의 처리 알고리즘의 성능 평가 결과를 설명한다. 제7.1절에서는 실험 데이타와 연속질의를 설명하고, 제7.2, 7.3, 7.4절에서는 실험 결과를 설명한다.

7.1 실험 데이타 및 실험 환경

실험에서는 제안하는 알고리즘의 성능을 기존의 방법들(IBS 트리를 사용하여 질의 프레디킷을 색인하는 방법과 질의 수행계획을 공유하는 방법)과 비교한다. 표 1은 실험에 사용된 알고리즘들을 정리한 표이다. 실험은 단순 선택 질의를 처리하는 경우와 슬라이딩 윈도우 조

표 1 실험에 사용된 알고리즘들

단순 선택 질의 처리	
SJ-Batch	Spatial Join CQ (배치 처리)
SJ-Immediate	Spatial Join CQ (즉각 처리)
IBS	IBS 트리를 사용한 연속질의 처리
슬라이딩 윈도우 조인 질의 처리	
SJ-Join	Spatial Join CQ (조인 확장)
IBS-JS	질의 수행계획 공유를 사용하여 조인 조건을 처리하고 IBS 트리를 사용하여 단순 선택 조건을 처리

인 질의를 처리하는 경우에 대해서 각각 수행하였다.

성능평가의 척도로는 연속질의 처리에 걸리는 수행 시간(elapsed time)을 사용한다. 수행 시간은 연속질의 처리의 성능을 평가하는 중요한 척도로서 데이타스트림 시스템이 단위 시간 당 얼마나 많은 입력 데이타를 처리할 수 있는가를 나타낸다. 실험의 공정성을 위하여 Spatial Join CQ 알고리즘에서는 데이타 색인을 구성하는 시간도 수행 시간에 포함하였다.

실험은 데이타스트림 소스가 한 개인 경우의 단순 선택 질의와 두 개인 경우의 슬라이딩 윈도우 조인 질의에 대해서 각각 수행한다. 실험에서 사용한 연속질의는 특별한 언급이 없으면 임의의 조건을 가지는 합성(synthetic)질의이다. 단순 선택 질의는 임의 개수의 속성들에 대한 선택(selection) 조건의 논리곱으로 이루어진다. 선택 조건은 $constant_1$ op_1 *attribute* op_2 $constant_2$ 형태를 가진다. 여기서 *attribute*는 속성의 이름이고, $constant_1$과 $constant_2$는 상수로서 $constant_1 \leq constant_2$인 0과 1사이의 임의의 값이 주어진다. op_1과 op_2는 비교 연산자로서 $\leq$ 와 $<$ 중 임의의 것이 주어진다. 또한, 조건에서 $constant_1$과 op_1만 주어질 확률, $constant_2$ 와 op_2만 주어질 확률, 그리고 둘 다 모두 주어질 확률은 동일하게 한다. 슬라이딩 윈도우 조인의 질의는 이러한 선택 조건과 더불어서 $attribute_1$ op $attribute_2$ 형태의 조인 조건이 추가로 주어진다. 이때, $attribute_1$과 $attribute_2$는 각각의 데이타스트림 소스에서의 조인 속성을 의미하고 op는 비교 연산자로서 $>$, $\geq$, $=$, $<$, $\leq$ 중에서 임의로 선택한다. 그리고, 모든 연속질의에서 슬라이딩 윈도우의 크기를 동일하게 하였다. 실험 데이타로는 각 속성 값이 0과 1사이에서 균일(uniform) 분포를 가지는 합성 데이타를 사용한다.

본 논문에서는 질의를 색인하기 위하여 MLGF[23,24]를 사용한다. MLGF는 점 객체를 저장하는 균형(balanced) 다차원 색인 구조로, 지역 분할 정책(local splitting strategy)[24]를 사용하여 객체의 분포가 불균일 할때도 이를 효율적으로 다룰 수 있는 특징을 가지고 있다. 실험에서는 메인 메모리 기반인 기존의 알고리즘들과의 공정한 비교를 위하여 MLGF 색인의 내용을 미리 메모리로 읽어들인 후 질의 처리를 수행하였다. 그리고, 데이타 색인에서는 힐버트 순서화를 사용하여 입력된 데이타 엘리먼트들을 인접한 순서로 저장한다.

실험은 1G 바이트 메모리를 가진 팬티엄4 2.5GHz 컴퓨터에서 Windows/XP를 기반으로 수행하였다. 실험 프로그램은 C++와 C 언어를 사용하여 작성하였다.

표 2는 제 7장에서 사용하는 표기법을 정리한 것이다.

표 2 주요 표기법

기 호	정 의 / 의 미
N_{query}	등록된 연속질의의 개수
N_{attr}	데이타스트림의 속성 개수[4]
R_{pred}	하나의 질의의 프레디킷들에 나오는 속성의 개수 / N_{attr}
N_{batch_data}	SJ-Batch에서 한번에 모아서 처리하는 데이타 엘리먼트의 개수
N_{dc}	데이타 클러스터를 구성하는 데이타 엘리먼트의 개수
$Size_{sw}$	슬라이딩 윈도우의 크기(데이타 엘리먼트의 개수)

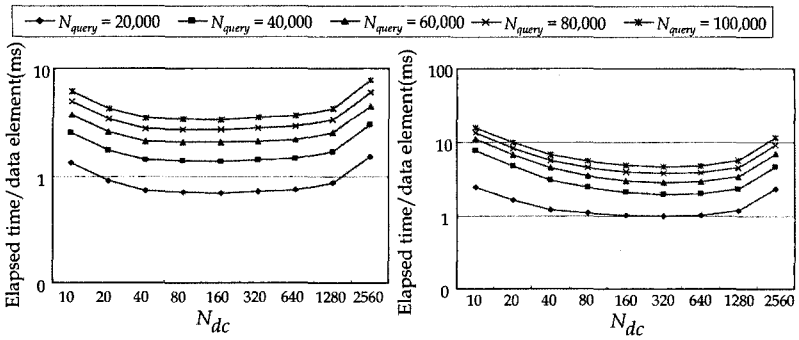
7.2 N_{dc}의 최적값

먼저, 실험을 통하여 최적의 N_{dc}를 구한다. N_{dc}는 SJ-Batch의 성능을 튜닝하는데 중요한 파라미터이다. 최적의 N_{dc}는 N_{batch_data}, N_{query}, N_{attr}, 그리고 데이타와 질의의 분포에 따라서 달라진다. 제 5.5절에서 설명하였듯이, 본 논문에서는 최적의 N_{dc}를 실험적으로 구한다.

그림 15는 N_{dc}와 N_{query}를 변화 시켰을 때, 연속질의 처리 수행 시간을 나타낸다. 여기서 N_{batch_data} = 10,000, 즉 10,000개의 데이타 엘리먼트들을 모아서 SJ-Batch로 처리하는 것으로 설정한다. 그림 15(a)와 (b)로 부터 N_{query}에 상관없이 N_{attr} = 4일 때는 N_{dc} = 200, N_{attr} = 8 일때는 N_{dc} = 400에서 수행 시간이 최소가 됨을 알 수 있다. 그리고, 최적값 부근에서 수행 시간의 변화가 크지 않음을 알 수 있다. 예를들어 그림 15(b)에서, N_{dc}의 값이 200부터 1200까지의 넓은 구간에서 최적값일 때에 비해 수행 시간의 증가폭이 10% 미만이었다. 따라서, N_{dc}의 정확한 최적값은 큰 오차없이 실험을 통하여 구할 수 있음을 알 수 있다.

표 3은 N_{query} = 10,000으로 설정하고 N_{batch_data}를 2,000부터 10,000까지 변화시켰을 때, 그림 15에서와 마찬가지 방법으로 최적의 N_{dc}값을 구한 결과를 정리한 것이다. 그리고, 이러한 최적의 N_{dc}를 사용하여 수행 시간을 측정하였다. 본 논문의 나머지 실험에서도 이러한

4) 본 실험에서는 데이타스트림의 모든 속성이 질의 프레디킷에 나올 수 있다고 가정한다. 따라서, 이 수치는 질의 색인에 저장되는 속성의 개수를 의미한다.



(a) N_{attr} = 4. (b) N_{attr} = 8.

그림 15 N_{dc} 와 N_{query} 의 변화에 따른 수행 시간(N_{batch_data} = 10,000)

표 3 최적의 N_{dc} 와 질의 처리 수행 시간 정리(N_{query} = 10,000)

N_{batch_data}		2,000	4,000	6,000	8,000	10,000
N_{attr} = 4	Optimal N_{dc}	100	120	120	160	200
	Elapsed time /data elements (ms)	0.406	0.391	0.391	0.381	0.375
N_{attr} = 8	Optimal N_{dc}	200	240	360	400	400
	Elapsed time /data elements (ms)	0.570	0.526	0.518	0.510	0.506

결과에 따라 N_{dc}를 설정하여 사용한다.

표 3의 결과를 좀 더 자세히 살펴보면, N_{batch_data}가 증가할수록, 즉 더 많은 데이타 엘리먼트들을 모아서 배치 처리할수록 최적의 N_{dc}는 커지고, 질의 처리 수행 시간은 감소하는 것을 알 수 있다. 최적의 N_{dc}가 커지는 이유는 N_{batch_data}가 증가할수록 다차원 공간에서 데이타 엘리먼트를 나타내는 점들의 밀도가 높아져서 더 많은 근접한 데이타 엘리먼트들을 하나의 데이타 클러스터로 묶을 수 있기 때문이다. 그리고, 질의 처리 수행 시간이 감소하는 이유는 최적의 N_{dc}가 커짐에 따라 그만큼 더 많은 데이타 엘리먼트들에 대한 질의 처리를 한 번의 질의 색인 검색으로 수행할 수 있어서 배치 처리의 이득을 더 많이 얻을 수 있기 때문이다. 따라서, 더 많은 데이타 엘리먼트들을 모아서 배치 방법으로 연속질의를 처리하는 것이 성능상 더 유리하다.

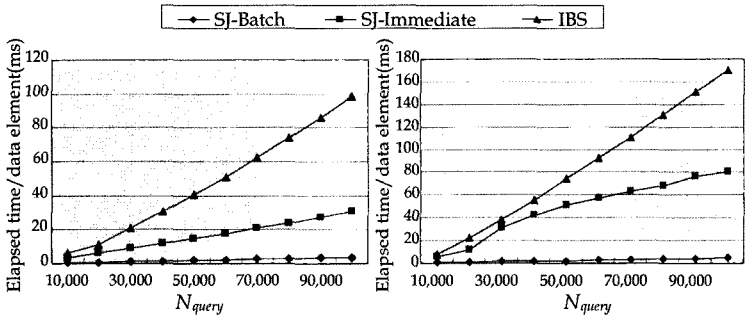
7.3 단순 선택 연속질의 처리

N_{query}와 N_{attr}의 영향

그림 16은 N_{query}의 변화에 따른 단순 선택 연속질의 처리 시간을 나타낸다. SJ-Batch 실험에서는 대표적으로 N_{batch_data}=10,000로 설정하였다. 그리고, 앞의 실험 결과에 따라 N_{attr}가 4인 경우는 N_{dc}=200으로, N_{attr}가 8인 경우는 N_{dc}=400으로 설정하였다. 그림 16에서 SJ-Batch가 가장 성능이 좋고, 다음으로 SJ-Immediate, IBS순으로 좋은 성능을 보인다. SJ-Immediate가 IBS에 비해서 성능이 좋은 이유는 색인을 사용하는 방법의 차이에서 기인한다. IBS에서는 각각의 속성에 대해서 질의가 따로 색인되기 때문에 여러 개의 색인을 탐색해야 하고 최종적으로 이 결과를 병합하는 과정이 필요하다. 반면 SJ-Immediate는 다차원 색인을 사용하여 하나의 색인에 질의를 저장하기 때문에 병합 과정이 필요없고 한번의 색인 탐색으로 질의 결과를 얻을 수 있어서 더 좋은 성능을 나타낸다. SJ-Batch가 가장 성능이 좋은 이유는 여러 개의 데이타 엘리먼트들을 데이타 클러스터로 모아서 함께 처리함으로써 질의 색인의 검색 횟수를 줄이고, SJW간의 겹침 성질을 사용하여 질의 색인 검색에 필요한 I/O를 최적화하였기 때문이다.

그림 16(a)에서 IBS와 SJ-Immediate간의 성능 차이는 N_{query}=10,000일 때 1.7배, N_{query}=100,000일 때 3.2배, IBS와 SJ-Batch간의 성능 차이는 각각 15.0배와 28.5배로 나타난다. 그림 16(b)에서 IBS와 SJ-Immediate간의 성능 차이는 N_{query}=10,000일 때 1.4배, N_{query}=100,000일 때 2.1배, IBS와 SJ-Batch간의 성능 차이는 각각 15.9배와 36.2배로 나타난다. 이러한 결과로부터 등록된 질의의 개수 N_{query}가 커짐에 따라 우리의 알고리즘의 성능상의 이점이 점점 더 커짐을 알 수 있다. 또한 데이타의 속성 개수가 우리 알고리즘의 성능상의 이점에 큰 영향을 주지 않음을 알 수 있다.

R_{pred}의 영향



(a) $N_{attr} = 4$. (b) $N_{attr} = 8$.

그림 16 N_{query}에 따른 단순 선택 연속질의 처리 수행 시간(N_{batch_data} = 10,000)

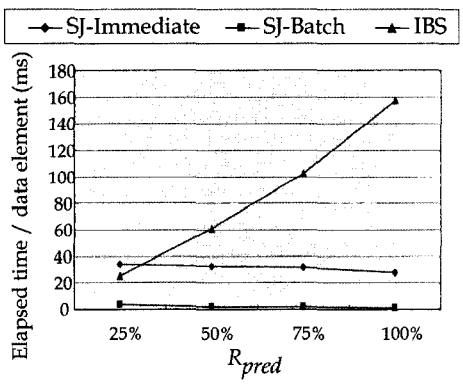


그림 17 R_{pred}에 따른 단순 선택 연속질의 처리 수행 시간(N_{attr} = 8, N_{query} = 50,000)

그림 17은 R_{pred}에 따른 단순 선택 연속질의 처리 수행 시간을 나타낸다. 실험 결과에서 IBS는 R_{pred}가 증가할 수록 수행 시간도 함께 증가한다. 이는 R_{pred}가 증가할 수록 각각의 속성에 더 많은 조건이 주어짐으로써 IBS트리의 깊이가 log R_{pred}에 비례해서 깊어져서 색인 탐색 비용이 증가하기 때문이다[11]. 반면 SJ-Batch와 SJ-Immediate는 오히려 조건이 많이 주어질수록 성능이 향상되는 것을 알 수 있다. 이유는 SJ-Batch와 SJ-Immediate에서는 질의를 조건 개수에 상관없이 하나의 영역으로 표현하여 색인하므로 질의 색인(즉, MLGF)의 깊이가 R_{pred}가 아니라 N_{query}에만 영향을 받기 때문이다. 그리고, 오히려 조건이 많이 주어질수록 질의를 표현하는 영역의 크기가 작아져서 색인 탐색 성능이 좋아지기 때문이다.

데이타 엘리먼트들의 분포에 따른 영향

지금까지의 실험에서는 입력된 데이타 엘리먼트들의 값의 분포가 균일하다고 가정하였다. 이제부터는 대부분의 입력 데이타 엘리먼트들의 값이 특정 좁은 범위에 집중되어 있는 편향된 분포를 가지는 경우의 실험 결과들을 보인다. 이러한 편향된 분포를 가지는 데이타의 예로는 실내 온도 센서의 데이타와 같이 시간에 따라 값이 크게 변하지 않는 데이타스트림을 들 수 있다. 입력 데이타의 편향도(*skewness*)를 전체 입력 데이타 엘리먼트들 중 특정 범위에 속하는 데이타 엘리먼트의 비율인 R_{skew}로 정의한다. 실험의 단순화를 위하여, 전체 도메인 크기의 10% 크기를 가지는 영역을 특정 영역으로 고려한다. 그림 18은 R_{skew}를 변화시켰을 때의 성능 변화를 나타낸다. SJ-Immediate와 IBS의 수행 시간은 균일 분포일때와 비교해서 큰 변화가 없는 반면 SJ-Batch는 R_{skew}가 커질수록 더 빨라져서 최대 약 1.7배 성능 향상이 있는 것을 알 수 있다. 이는 특정 범위에서 데이타 엘리먼트들의 밀도가 높아져서 더 많은 데이타 엘리먼트들로 하나의 데이타 클러스터를 구성할 수 있게 되었기 때문이다. 최적의 N_{dc}값은 R_{skew}가 20%일 때 200에서 100%일 때 400으로 증가하였다. 따라서, 이러한 특성으로 인하여 SJ-Batch는 데이타의 값이 편향된 경우에 더 유용함을 알 수 있다.

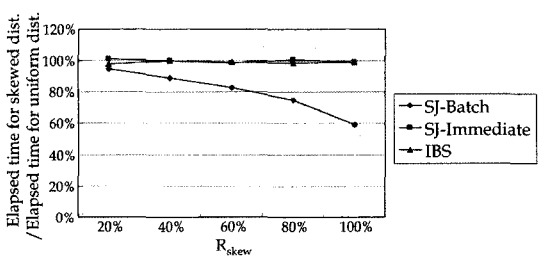
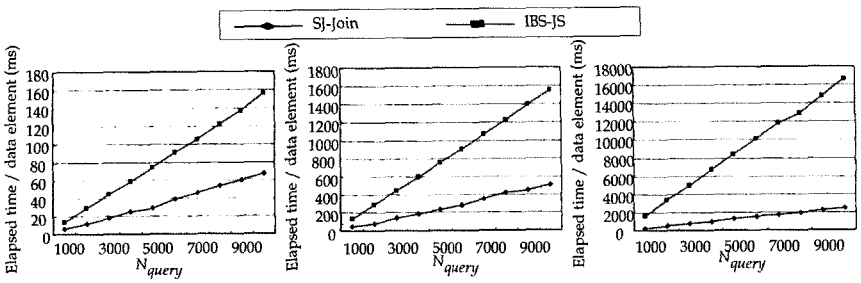


그림 18 데이타 분포에 따른 수행 시간 비교(N_{atrt} = 4, N_{query} = 100,000, N_{batch_data} = 10,000)

7.4 슬라이딩 윈도우 조인 연속질의 처리 N_{query}와 $Size_{sw}$의 영향



(a) $Size_{sw}$ = 100. (b) $Size_{sw}$ = 1,000. (b) $Size_{sw}$ = 10,000.

그림 19 $Size_{sw}$ 에 따른 슬라이딩 윈도우 조인 연속질의 처리 시간

그림 19는 두 개의 데이타스트림이 존재하는 경우에 N_{query}의 변화에 따른 슬라이딩 윈도우 조인 연속질의 처리 시간을 나타낸다. SJ-Join에서는 조인 조건을 색인하기 위하여 객체 분할 방법을 사용하여 조인 속성의 차원축으로 4개로 영역으로 분할하였다. 실험 결과에서 SJ-Join이 IBS-JS 보다 좋은 성능을 보인다. 이는 그림 16의 실험에서 SJ-Batch가 IBS에 비해서 성능이 좋았던 것과 마찬가지 이유에서이다. 그리고, $Size_{sw}$가 커질수록 IBS-JS와 SJ-Join간의 성능차가 커짐을 알 수 있다. 이는 IBS-JS는 $Size_{sw}$에 비례해서 비용이 증가하는 반면, SJ-Join은 $Size_{sw}$가 증가할수록 표 3에서 보인 것과 같이 더 큰 데이타 클러스터를 구성함으로써 질의 색인 검색을 줄이는 효과를 얻었기 때문이다($Size_{sw}$는 표 3의 N_{batch_data}에 대응된다.). 실험에서 $Size_{sw}$=100인 경우는 SJ-Join이 IBS-JS에 비해서 최대 2.6배의 성능 향상이 있었으며, $Size_{sw}$=1,000인 경우는 3.9배, $Size_{sw}$= 10,000인 경우는 6.9배 성능 향상이 있었다.

지금까지의 실험 결과를 종합해 보면, 제안하는 알고리즘은 단순 선택 연속질의와 슬라이딩 윈도우 조인 연속질의 모두에서 기존의 연속질의 최적화 기법에 비해서 좋은 성능을 나타낸다. 그리고, 제안하는 알고리즘들은 (1) N_{query}, (2) $Size_{sw}$, (3) R_{pred}, (4) R_{skew}가 증가할수록 성능 향상이 더 증가하는 좋은 특성을 가지고 있다.

8. 결 론

본 논문은 데이타스트림 환경에서 연속질의를 효율적으로 처리하는 방법을 제안하였다. 제안하는 방법은 데이타와 질의의 이원성을 사용하여 데이타와 질의를 동등하게 볼 수 있다는 점에 착안한다. 본 논문의 공헌은 다음과 같다.

첫째, 데이타와 질의를 동등하게 취급하는 데이타와 질의의 이원성 모델(Duality Model of Data and Queries)을 제안하였다. 이원성 모델은 질의와 데이타를 다차원 공간상의 영역으로 표현하고, 서로 겹치는 질의와 데이타의 영역들을 찾는 연산으로 질의 처리를 정의한다. 이 모델은 데이타와 질의를 동등하게 다룸으로써 서로 이원적인 데이타-이니셔티브와 질의-이니셔티브 방법을 하나의 질의 처리 모델 안에서 통합한다.

둘째, 이원성 모델을 사용하여 데이타스트림에서의 연속질의 처리 문제를 다차원 공간에서의 공간조인 문제로 변환하는 새로운 관점을 제시하였다. 이러한 변환을 통하여 다차원 공간조인 기법을 사용하여 연속질의를 효율적으로 처리할 수 있음을 보였다.

세째, 공간조인 기반 연속질의 처리 알고리즘 *Spatial Join CQ*를 제안하였다. 이 알고리즘은 대칭적인 성격을 가지는 공간조인 연산을 데이타 엘리먼트들의 집합과 질의들의 집합에 대해서 수행하여 서로 이원적인 두 가지 질의 처리 방법의 효과를 동시에 얻음으로써 연속질의를 -특히 배치 처리에서- 보다 효율적으로 처리할 수 있다.

네째, 실험을 통하여 제시한 알고리즘이 기존의 데이타스트림에서의 연속질의 처리 방법에 비해서 더 우수한 성능을 가짐을 보였다. 제시한 알고리즘은 기존의 방법에 비해서 단순 선택 질의 처리시에 즉각 처리 방법을 사용할 때는 최대 3.2배의 빠른 성능을 보였으며, 배치 처리 방법을 사용할 때는 최대 36.2배의 빠른 성능을 보였다. 슬라이딩 조인 질의 처리시에는 최대 6.9배의 빠른 성능을 보였다. 그리고, 등록된 연속질의의 개수, 슬라이딩 윈도우의 크기, 속성에 조건이 주어질 확률, 입력 데이타의 분포가 편향(skew)된 정도가 클수록 성능 향상이 더 증가하는 좋은 특성을 가지고 있음을 보였다.

이러한 결과로 볼 때 본 논문에서 제시하는 이원성 모델과 연속질의 처리 알고리즘은 데이타스트림 환경에서 연속질의의 처리 성능을 크게 향상시킬 수 있는 새로운 방법으로 믿어진다. 향후 연구로는 제안한 알고리즘을 확장하여 다중 슬라이딩 윈도우 조인 연속질의을 효율적으로 처리하는 방법을 연구한다. 그리고, 최적의

데이타 클러스터 크기를 자동적으로 구하는 방법에 대해서 연구한다.

참 고 문 헌

[1] Babcock, B. et al., "Models and Issues in Data Stream Systems," In *Proc. the 21st ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems(PODS)*, Madison, Wisconsin, pp. 1-16, June 2002.

[2] Berchtold, S., Bohm, C., and Kriegel, H.-P., "The Pyramid-Technique: Towards Breaking the Curse of Dimensionality," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, Seattle, Washington, pp. 142-153, June 1998.

[3] Brinkhoff, T., Kriegel, H.-P., and Seeger, B., "Efficient Processing of Spatial Join Using R-trees," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, Washington, DC., pp. 237-246, May 1993.

[4] Chandrasekaran, S. et al., "TelegraphCQ: Continuous Dataflow Processing for an Uncertain World," In *Proc. the First Biennial Conf. on Innovative Data Systems Research*, Asiloma, Califonia, pp. 269-280, Jan. 2003.

[5] Chen, J. et al., "NiagaraCQ: A Scalable Continuous Query System for Internet Databases," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, Dallas, Texas, pp. 379-390, June 2000.

[6] Faloutsos, C. and Roseman, S., "Fractals for Secondary Key Retrieval," In *Proc. the Eighth ACM SIGACT-SIGMOD Symp. on Principles of Database Systems(PODS)*, Philadelphia, Pennsylvania, pp. 247-252, Mar. 1989.

[7] Finkel, R. A. and Bentley, J. L., "Quad-trees: A Data Structure for Retrieval on Composite Keys," *ACTA Informatica*, Vol. 4, No. 1, pp. 1-9, 1974.

[8] Fox, E. A. et al., "Order-preserving minimal perfect hash functions and information retrieval," *ACM Trans. on Information Systems*, Vol.9, No.3, pp. 281-308, July 1991.

[9] Golab, L. and Ozsu, M. T., "Issues in Data Stream Management," *ACM SIGMOD Record*, Vol. 32, No. 2, pp. 5-14, June 2003.

[10] Guttman, A., "R-trees: a Dynamic Index Structure for Spatial Searching," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, Boston, Massachusetts, pp. 47-57, June 1984.

[11] Hanson, E. N. et al., "A Predicate Matching Algorithm for Database Rule Systems," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, Atlantic City, New Jersey, pp. 271-280, June 1990.

[12] Hinrichs, K. and Nievergelt, J., "The Grid File: A Data Structure Designed to Support Proximity Queries on Spatial Objects," In *Proc. Int'l Workshop on Graphtheoretic Concepts in Computer Science*, Linz, Austria, pp.100-113, Aug. 1983.

[13] Huang, Y.-W., Jing, N., and Rundensteiner, E. A., "Spatial Joins Using R-trees: Breadth-First Traversal with Global Optimizations," In *Proc. the 23rd Int'l Conf. on Very Large Data Bases*, Athens, Greece, pp.396-405, Aug. 1997.

[14] Kang, J., Naughton, J. F., and Viglas, S. D., "Evaluating Window Joins over Unbounded Streams," In *Proc. the 19th IEEE Int'l Conf. on Data Engineering(ICDE)*, Bangalore, India, pp. 341-352, Mar. 2003.

[15] Kriegel, H.-P. et al., "Spatial Query Processing for High Resolutions," In *Proc. the Eighth Int'l Conf. on Database Systems for Advanced Applications*, Tokyo, Japan, pp. 17-26, Mar. 2003.

[16] Motwani, R. et al., "Query Processing, Approximation, and Resource Management in a Data Stream Management System," In *Proc. the First Biennial Conf. on Innovative Data Systems Research*, Asiloma, California, pp. 245-256, Jan. 2003.

[17] Orenstein, J. A. and Merrett, T. H., "A Class of Data Structures for Associative Searching," In *Proc. the Third ACM SIGACT-SIGMOD Symp. on Principles of Database Systems(PODS)*, Waterloo, Canada, pp. 181-190, Apr. 1984.

[18] Seeger, B. and Kriegel, H.-P., "Techniques for Design and Implementation of Efficient Spatial Access Methods," In *Proc. the 14th Int'l Conf. on Very Large Data Bases*, Los Angeles, California, pp.360-371, Aug. 1988.

[19] Seeger, B. and Kriegel, H.-P., "The Buddy-tree: An Efficient and Robust Access Method for Spatial Database Systems," In *Proc. the 16th Int'l Conf. on Very Large Data Bases*, Queensland, Australia, pp.590-601, Aug. 1990.

[20] Song, J.-W., Whang, K.-Y., Lee, Y.-K., and Kim, S.-W, "Spatial Join Processing Using Corner Transformation," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 11, No. 4, July 1999.

[21] Terry, D. et al., "Continuous Queries over Append-Only Databases," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, San Diego, California, pp. 321-330, June 1992.

[22] Weber, R., Schek, H.-J., and Blott, S., "A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces," In *Proc. the 24th Int'l Conf. on Very Large Data Bases*, New York City, New York, pp.194-205, Aug. 1998.

[23] Whang, K.-Y. and Krishnamurthy, R., Multilevel Grid Files, IBM Research Report RC11516, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, Nov. 1985.

[24] Whang, K.-Y. and Krishnamurthy, R., "The Multi-

level Grid File - a Dynamic Hierarchical Multidimensional File Structure," In *Proc. Int'l Conf. on Database Systems for Advanced Applications*, Tokyo, Japan, pp. 449-459, Apr. 1991.

[25] Zdonik, S. et al., "The Aurora and Medusa Projects," *IEEE Data Engineering Bulletin*, Vol. 26, No. 1, pp. 3-10, Mar. 2003.

임 효 상

1994년 3월~1998년 2월 연세대학교 컴퓨터과학과 학사. 1998년 3월~1999년 8월 한국과학기술원 전자전산학과 전산학전공(석사). 1999년 9월~현재 한국과학기술원 전자전산학과 전산학전공(박사) 관심분야는 데이타스트림, 센서네트웍, 시계열 데이타베이스, 공간 데이타베이스

이 재 길

정보과학회논문지 : 데이타베이스
제 33 권 제 1 호 참조

이 민 재

정보과학회논문지 : 데이타베이스
제 33 권 제 1 호 참조

황 규 영

정보과학회논문지 : 데이타베이스
제 33 권 제 1 호 참조