이동 객체의 부분차원 스카이라인 질의를 위한 효율적인 가지치기 기법

(An Efficient Pruning Method for Subspace Skyline Queries of Moving Objects)

김 진 호 † 박 영 배 ††
(Jin-Ho Kim) (Young-Bae Park)

요 약 대부분의 스카이라인 질의에 대한 이전 연구들은 대상 객체의 정적 속성만을 고려하였다. 최근에는 모바일 응용 환경의 발전에 따라 이동 객체에 대한 연속적인 스카이라인 질의에 대한 필요성이 증대되고 있다. 연속적인 스카이라인 질의를 처리하기 위해 최근에 몇 가지 기법들이 제안되었지만, 이 기법들은 사용자가 관심을 가지는 일부 속성을 임의로 선택하는 부분차원 스카이라인 질의에 대해서는 고려하지 않았다. 이로 인하여 이동 객체와 부분차원을 동시에 고려해야 하는 모바일 응용에 있어서는 이전 연구들을 적용할 수 없다.

이 논문에서는 질의 시점에 이동 객체의 부분차원 스카이라인을 효율적으로 계산하기 위한 지배 객체 기반 가지치기 기법을 제안한다. 그리고 제안한 기법의 효율성을 증명하기 위해 모의실험을 통한 성능 평가를 수행한다.

키워드 : 부분차원 스카이라인 질의, 이동 객체, 지배 객체 기반 가지치기

Abstract Most of previous works for skyline queries have focused only on static attributes of target objects. With the advance in mobile applications, however, the need of continuous skyline queries for moving objects has been increasing. Even though several techniques to process continuous skyline queries have been proposed recently, they cannot process subspace queries, which use only the subset of attribute dimensions. Therefore it is not feasible to utilize those methods for mobile applications which must consider moving objects and subspaces simultaneously.

In this paper, we propose a dominant object-based pruning method to compute subspace skyline of moving objects efficiently at query time and present the experimental results to show the effectiveness of the proposed method.

Key words : subspace skyline queries, moving objects, dominant object-based pruning

1. 서 론

최근 이동 컴퓨팅 기술의 발전과 더불어 위치 측위 기술의 발전으로 휴대폰, PDA들과 같은 이동 장치의 시공간 데이타의 수집 및 전달이 용이해짐에 따라 다양한 위치 기반 서비스(location-based service)에 대한 요구는 점점 더 증가하고 있다.

스카이라인 질의(skyline queries)는 전체 객체집합으로부터 다른 객체들이 지배하지 않는 사용자가 선호하는 객체들로 이루어진 부분집합을 검색한다[1]. 이러한 질의는 다중 속성들을 동시에 고려해야 하는 다목적 의사결정(multi-objective or multi-criteria decision making)의 응용들에서 매우 중요한 연산이다.

대부분의 스카이라인 질의에 대한 이전 연구들은 대상 객체의 정적 속성들만을 대상으로 연구되어 왔다. 초기에 제안된 기법[1-4]들은 대부분 전체 대상 객체들을 한 번 이상 접근해야 했지만, 이후 적절한 색인을 이용

† 학생회원 : 명지대학교 컴퓨터공학과
solbong@mju.ac.kr
†† 종신회원 : 명지대학교 컴퓨터공학과 교수
parkyb@mju.ac.kr
논문접수 : 2007년 8월 2일
심사완료 : 2008년 1월 2일

정보과학회논문지: 데이타베이스 제35권 제2호(2008.4)

하여 페이지 접근을 최소화하는 기법[5-9]들이 제안되었다. 이러한 연구들은 모두 질의 시점 이전에 이미 값이 정해져 있는 정적 속성들만을 고려한 기법들이며, 동적 속성을 포함한 이동 객체에는 적용할 수 없다.

이동 객체는 공간 위치를 시간에 따라 변경하므로 대상 객체와의 거리와 같은 동적 속성 값들은 질의 시점 이전에 정해질 수 없으며, 질의가 유지되는 동안 계속해서 변화한다[10-13]. 이동 객체에 대한 대부분의 이전 연구들은 단일 동적 속성만을 고려하는 시공간 질의에 대한 처리 기법을 제안하고 있기 때문에 다중 속성을 고려해야 하는 스카이라인 질의에는 적용할 수 없다.

최근에는 모바일 응용 환경의 발전에 따라 이동 객체에 대한 연속적인 스카이라인 질의에 대한 필요성이 증대되고 있다. 모바일 환경에서 스카이라인 질의를 처리하기 위해서는 대상 객체의 정적 속성과 동적 속성을 동시에 고려해야 한다. 예를 들어 이동 객체는 "현재 위치에서 가깝고, 가격이 싸고, 해변까지의 거리가 가까운 호텔"의 검색과 같은 스카이라인 질의를 요청할 수 있다. 이 질의를 처리하기 위해서는 정적 속성 <가격, 해변까지의 거리>에 대한 스카이라인과 동적 속성 <이동 객체와 호텔까지의 거리>에 대한 스카이라인을 계산하여야 한다. 이와 동시에 이동 객체의 연속 질의에 대한 처리도 가능해야 한다.

최근 제안된 이동 객체의 연속적인 스카이라인 질의 처리 기법[14-17]들은 대부분 효율적인 처리를 위해 사용자가 관심을 가지는 부분차원을 고정하여 R-트리 등으로 색인하고, 단지 이동 객체의 동적 속성만을 고려하였기 때문에 사용자가 임의로 부분차원을 선택하는 질의에 대한 처리는 불가능하다. 여기서 부분차원이란 대상 객체의 여러 속성들 중에 사용자가 임의로 선택하는 일부 속성들을 말한다. 예를 들어 호텔의 정적 속성들이 <가격, 해변까지의 거리, 공항까지의 거리>의 3차원으로 구성되어 있는 경우, 사용자는 "현재 위치에서 가깝고, 가격이 싸고, 해변까지의 거리가 가까운 호텔"을 검색하고, 다른 사용자는 "현재 위치에서 가깝고, 가격이 싸고, 공항까지의 거리가 가까운 호텔"을 검색할 수 있다. 부분차원을 고정하여 색인하는 이전 기법을 이용하여 이러한 질의를 처리하기 위해서는 가능한 모든 경우의 수만큼 색인을 구성해야 하기 때문에 고차원에서는 적용이 불가능하다.

부분차원을 고려한 이동 객체의 연속적인 질의는 크게 두 가지 기법이 요구된다. 첫째, 이전 기법들처럼 전처리 과정에서 정적 속성에 대해 대상 객체들을 미리 색인하는 것이 불가능하기 때문에 질의 시점에 실시간으로 이동 객체의 현재 위치에 대한 부분차원 스카이라인을 효율적으로 계산할 수 있는 기법이 요구된다. 둘째 이동 객체의 위치 변화에 따르는 연속 질의에 대한 처리 기법이 요구된다.

이 논문에서는 질의 시점에 이동 객체의 현재 위치에 대한 부분차원 스카이라인을 효율적으로 계산하기 위한 지배 객체 기반 가지치기(dominant object-based pruning) 기법을 제안한다. 그리고 제안한 기법의 효율성을 증명하기 위해 모의실험을 통한 성능 평가를 수행한다.

2. 관련 연구

2.1 정적 속성에 대한 부분차원 스카이라인 질의 처리 기법

부분차원이란 대상 객체의 여러 속성들 중에서 사용자가 관심을 가지는 일부 속성들을 의미한다. 대상 객체가 고차원의 속성을 가지는 경우, 질의 시점에 사용자는 자신의 관심에 따라 임의로 부분차원을 선택하며, 일반적으로 저차원의 부분차원을 선택한다. 그림 1은 고차원의 속성을 가지고 있는 대상 객체 집합의 예를 보여준다. 그림의 대상 객체의 정적 속성인 가격과 사각형의 내부에 나열된 여러 속성들(해변까지의 거리, 지하철역까지의 거리, 공항까지의 거리 등)에 대해서, 휴가 여행 중인 사용자는 2차원인 <가격, 해변까지의 거리>를 부분차원을 선택하여 질의하고, 해외 출장 중인 사용자는 3차원인 <가격, 공항까지의 거리, 지하철역까지의 거리>를 부분차원으로 선택하여 질의할 수 있다.

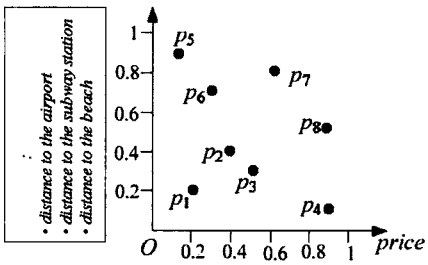


그림 1 고차원 속성을 가지는 대상 객체의 부분차원

BBS(Branch and Bound Skyline) 기법[8]은 대상 객체의 특정 부분차원에 대해 대상 객체들을 R-트리로 미리 색인함으로써 이전 연구들에 비해 우수한 성능을 나타낸다. 그러나 이 기법은 대상 객체의 속성이 고차원이고 전처리 과정에서 색인된 부분차원(일반적으로 전체 속성들)이 아닌 다른 부분차원(전체 속성이 아닌 일부 속성들)에 대해 질의할 경우에는 처리가 불가능하다. 이러한 문제점을 해결하기 위해 사용자가 선택할 수 있는 모든 경우의 부분차원에 대해 미리 색인을 구성한다면 d차원의 전체 차원에 대하여 2^d-1개의 색인이 필요하기 때문에 현실적으로 적용이 불가능하다.

SUBSKY 기법[18]에서는 고차원으로 구성된 대상 객체들의 정적 속성 값들을 식 (1)과 같은 1차원의 값으로 표현하고 식 (2)의 조건을 이용하여 색인을 구성하지 않고 가지치기를 수행하여 부분차원 스카이라인을 결정하는 효율적인 기법을 제안한다.

$$f(p) = \max(1 - p[i]), i = 1, ..., d \quad (1)$$

$$f(p) < \min(1 - p_{sky}[i]),\ i \in SUB \quad (2)$$

식 (1)의 $f(p)$ 값은 대상 객체 p의 전체 차원 중에서 가장 좋은 속성 값을 의미하며, 이 값은 각 객체에 대해 고정이므로 질의 발생 이전에 미리 계산하여 색인이 가능하다. 그리고 식 (2)의 좌변은 질의 시점에 사용자가 관심을 가지는 부분차원 SUB에 대해 스카이라인 객체 p_{sky}가 가지는 가장 나쁜 속성 값을 의미한다. 그러므로 전체 차원에 대해 p의 가장 좋은 속성 값이 p_{sky}의 가장 나쁜 속성 값보다 좋지 않다면 p_{sky}는 p를 지배하기 때문에 $f(p)$만을 비교하여 p를 부분차원 스카이라인 연산에서 제외시킬 수 있다. $f(p)$ 값에 대해 B$^+$-트리 등의 색인을 구성하고, 부분차원 스카이라인 질의를 대상 객체들을 $f(p)$의 내림차순으로 접근하여 처리한다면, 전체 대상 객체를 접근하지 않고 효율적으로 부분차원 스카이라인을 계산할 수 있다.

그림 2는 *SUBSKY*에서 제안한 기법을 이용한 지배 관계를 판별 방법을 도식한 것이다. 그림에서 p_{sky}에 대하여 식 (2)의 좌변은 p_{sky}가 지배하는 음영 처리된 영역을 의미한다. 즉, 이 영역이 포함하는 객체들의 $f(p)$ 값은 식 (2)의 조건을 만족하게 된다. 그러므로 전체 차원에 대해 불변적인 1차원 $f(p)$ 값만으로도 효율적으로 지배 관계를 판별할 수 있다. 그림에서 음영 처리된 영역의 모든 객체들은 $f(p)$ 값만으로 연산에서 제거할 수 있지만 p'와 같은 객체의 경우에는 $f(p)$ 값만으로는 지배 관계를 판별할 수 없기 때문에 연산에 포함되게 된다.

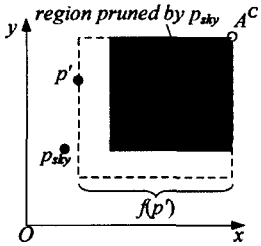


그림 2 $f(p)$와 지배 관계

2.2 이동 객체의 연속적인 스카이라인 질의 처리 기법

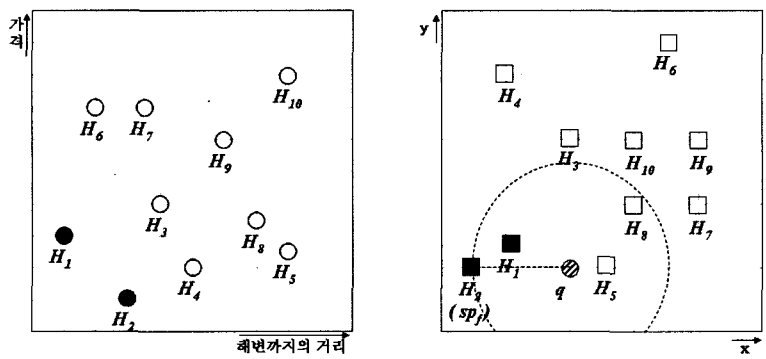
이동 객체에 대한 스카이라인 SK_{all}은 대상 객체의 정적 속성에 대한 정적 스카이라인 SK_{ns}와 정적 속성에 대해 자신의 모든 지배 객체보다 이동 객체와 가까운 객체들로 구성된 동적 스카이라인 SK_{chg}로 구성된다.

$$SK_{ns} = \{p_i \mid \text{정적속성에 대해지배객체가없는객체}\} \quad (3)$$
$$SK_{chg} = \{p_i \mid \exists p_j \prec p_i,\ dist(q,p_i) < dist(q,p_j)\}$$
$$SK_{all} = SK_{ns} + SK_{chg}$$

SK_{ns}는 질의가 종료할 때까지 항상 일정한 집합을 유지하며, SK_{chg}는 이동 객체의 위치에 따라 변경되는 집합이다. 각 스카이라인은 식 (3)과 같이 정의할 수 있다. 여기서 기호 $p_j \prec p_i$는 p_j는 p_i를 지배한다는 의미이다.

대상 객체들이 그림 3의 (a)와 같은 정적 속성 관계를 가지면 SK_{ns}는 $\{H_1, H_2\}$이다. 그리고 공간 속성 관계와 이동 객체 q의 위치가 그림 3의 (b)와 같으면 H_4는 스카이라인 객체인 자신의 지배 객체 H_2보다 이동 객체 q에 가깝기 때문에 SK_{chg}에 포함된다.

대상 객체가 SK_{chg}에 포함되기 위해서는 자신을 지배하는 현재 스카이라인 객체들보다 이동 객체에 가까워야 한다. 그러나 질의 시점에 스카이라인 객체가 아닌 모든 대상 객체에 대하여 이러한 비교 연산을 수행하는 것은 상당히 많은 시간을 소비하는 문제점이 발생한다. 이러한 문제점을 해결하기 위해서는 적적한 가지치기 기법이 요구된다.



(a) 정적 속성 (b) 공간 속성

그림 3 대상 객체들의 속성 관계

2.2.1 CSQMO 기법

CSQMO 기법[14]에서는 부분차원이 고정되어 있다고 가정하고 대상 객체의 정적 속성들은 R-트리, 공간 좌표는 2차원 그리드(Grid) 파일로 색인한다. 이 기법에서는 SK_{ns}는 *BBS* 기법[8]을 이용하여 계산하고, SK_{chg}를 계산하기 위한 가지치기 기법을 제안한다. 제안한 가지치기 기법은 그림 3의 (b)와 같이 SK_{ns}의 스카이라인 객체들 중에서 이동 객체 q에 가장 먼 스카이라인 객체 sp_f와의 거리 $dist(q, sp_f)$를 가지치기 기준으로 사용하여 기준 거리 보다 먼 대상 객체들은 연산에서 제외한다. 그러나 이 기법은 그림 3의 (b)와 같이 SK_{ns}의 스카이라인 객체들의 공간 좌표가 서로 가까이 분포하고, 이동 객체 q의 현재 위치가 스카이라인 객체들에 가까운 특별한 경우에만 효율적인 성능을 보인다. 이 기법의 문제점은 4장에서 예제 데이타를 통해 좀 더 자세하게 기술한다.

2.2.2 스카이라인 영역 기법

스카이라인 영역 기법[15]에서는 대상 객체가 SK_{chg}에 포함될 *Voronoi cell* 기반의 공간 영역을 정의하고, 전처리 과정을 통해 모든 대상 객체에 대해 해당 영역을 미리 계산하여 효율적으로 계산할 수 있는 기법을 제안한다.

이 기법에서 제안한 스카이라인 영역 SR_i이란 객체 p_i가 정적 속성에 대한 자신의 지배 객체 p_j들보다 이동 객체 q에 가까운 영역으로, 이동 객체가 q가 이 영역 내에 위치하면 p_i가 SK_{chg}에 포함될 수 있는 영역이다. SR_i는 식 (4)와 같이 정의할 수 있다.

$$SR_i = \{q(x,y) | dist(q, p_i) < dist(q, p_j), \forall p_j < p_i\} \quad (4)$$

SR_i는 *Voronoi cell*과 유사하게 지배 객체와의 수직 이등분선들을 이용하여 결정할 수 있다. 모든 대상 객체의 스카이라인 영역을 전처리 과정에서 미리 결정하고 각 스카이라인 영역 간의 겹침 관계를 색인한다면 질의가 발생한 시점에 SK_{chg}를 효율적으로 결정할 수 있다. 그림 4의 (b)는 그림 4의 (a)와 같은 정적 속성 관계를 가지는 대상 객체들의 스카이라인 영역을 나타내며 이를 이용하면 Q_1은 SR_3, SR_6내에 위치하므로 SK_{chg}={H_3, H_6}이며, Q_2인 경우 SK_{chg}={H_5, H_7}임을 간단하게 계산할 수 있다.

3. 이동 객체의 부분차원 스카이라인 질의

이동 객체의 부분차원 스카이라인도 이동 객체의 스카이라인과 같이 이동 객체의 부분차원 스카이라인 SK_{all}은 정적 부분차원 스카이라인 SK_{ns}와 동적 부분차원 스카이라인 SK_{chg}로 구성된다. 이 장에서는 질의 예제를 통해 각 부분차원 스카이라인의 정의에 대해 설명한다.

3.1 이동 객체의 부분차원 스카이라인 질의 예제

이전 연구에서 자주 이용했던 호텔의 예를 들어 SK_{all}을 계산하는 방법을 설명한다. 먼저 호텔의 정적 속성들이 <가격, 해변까지의 거리, 공항까지의 거리>의 3차원으로 구성된다고 가정하고, 이동 객체의 부분차원 스카이라인 질의는 q_a와 q_b와 같이 주어질 수 있다.

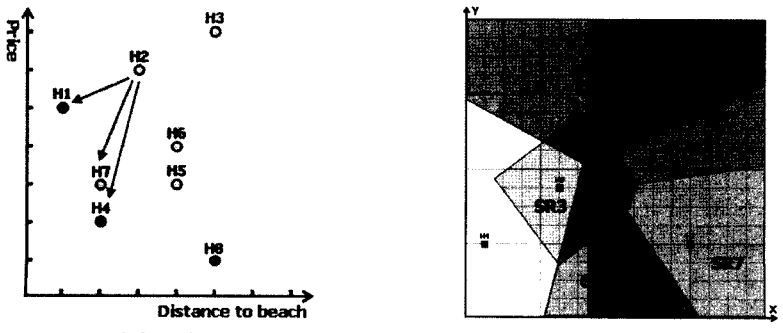
q_a : "*가격이 싸고, 해변까지의 거리가 가깝고, 현재 위치에서 가까운 호텔*"

q_b : "*가격이 싸고, 공항까지의 거리가 가깝고, 현재 위치에서 가까운 호텔*"

각 질의의 SK_{all}은 2차원의 정적 부분차원 스카이라인 SK_{ns}와 이동 객체와 대상 객체 간의 거리를 기준으로 하는 동적 부분차원 스카이라인 SK_{chg}를 계산하여야 한다. 표 1은 예제에 사용하는 데이타이며 호텔들의 부분차원 정적 속성 관계는 그림 5와 같다.

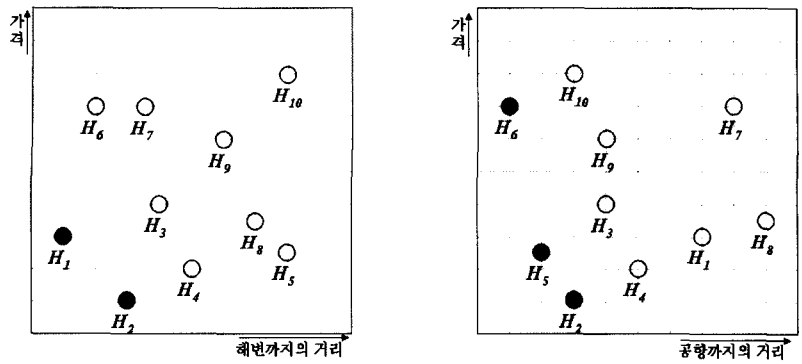
3.2 정적 부분차원 스카이라인

SK_{ns}는 *SUBSKY* 기법[18]에서 제안한 것과 같이 모든 차원의 속성 값의 범위를 [0, 1]로 하고 전체 차원의 속성 값들을 1차원으로 대표할 수 있는 식 (1)과 같은 $f(p)$ 값과 식 (2)의 조건을 이용한 가지치기 기법으로



(a) 정적 속성 관계 (b)전체 스카이라인 영역

그림 4 스카이라인 영역 기법



(a) 가격-해변까지의 거리 (b) 가격-공항까지의 거리

그림 5 호텔들의 부분차원 정적 속성 관계

표 1 부분차원 스카이라인 예제 데이타

호텔	가격	거리 (해변)	거리 (공항)	f(p)	x	y
H_1	0.3	0.1	0.7	0.9	0.1	0.8
H_2	0.1	0.3	0.3	0.9	0.2	0.2
H_3	0.4	0.4	0.4	0.6	0.4	0.6
H_4	0.2	0.4	0.5	0.8	0.55	0.2
H_5	0.25	0.8	0.2	0.8	0.5	0.4
H_6	0.7	0.2	0.1	0.9	0.7	0.9
H_7	0.7	0.35	0.8	0.65	0.8	0.4
H_8	0.35	0.7	0.9	0.65	0.6	0.4
H_9	0.6	0.6	0.4	0.6	0.8	0.6
H_{10}	0.8	0.8	0.3	0.7	0.6	0.6

효율적으로 계산할 수 있다. 이 기법을 적용하면 질의 예제 q_a의 SK_{ns}={H_1, H_2}이고 q_b의 SK_{ns}={H_2, H_5, H_6}이다. 계산된 SK_{ns}는 이동 객체의 위치와는 무관하기 때문에 질의가 종료하기 전까지 항상 동일한 집합을 유지하게 된다.

3.3 동적 부분차원 스카이라인

SK_{chg}는 자신을 부분차원 정적 속성에 대해 지배하는 객체보다 이동 객체에 가까운 객체들을 포함한다. 다시 말해서, 현재 스카이라인 SK_{curr}이 포함한 객체들이 부분차원 정적 속성에 대해 지배하는 대상 객체들의 집합 P_{tee}의 객체들과 이동 객체 간의 거리를 계산하여 판별하게 된다.

부분차원 스카이라인 질의의 경우, 스카이라인 기법[15]과 같이 모든 대상 객체들의 스카이라인 영역을 전처리 과정을 미리 계산할 수 없다. SK_{chg}는 부분차원과 이동 객체의 위치를 알 수 있는 질의 시점에 비로소 계산될 수 있기 때문이다. 그러므로 앞서 설명한 바와 같이 효율적인 연산을 위해서는 P_{tee}의 전체 객체들에서 SK_{chg}가 포함할 수 있는 후보 객체 집합 P_{cand}를 추출할 수 있는 가지치기 기법이 필요하게 된다.

4. 이동 객체의 부분차원 스카이라인 질의를 위한 효율적인 가지치기 기법

이동 객체의 부분차원 스카이라인 질의는 부분차원을 질의 시점에 결정하게 되므로 전처리 과정이 필요한 이전 기법들[14,15]로는 질의 처리가 불가능하다. 또한 *SUBSKY* 기법[18]을 이용하여 SK_{ns}를 계산하고, *CSQMO* 기법[14]에서 제안한 가지치기 기법을 이용해 SK_{chg}를 계산한다 하더라도 성능 향상을 기대하기 어렵다.

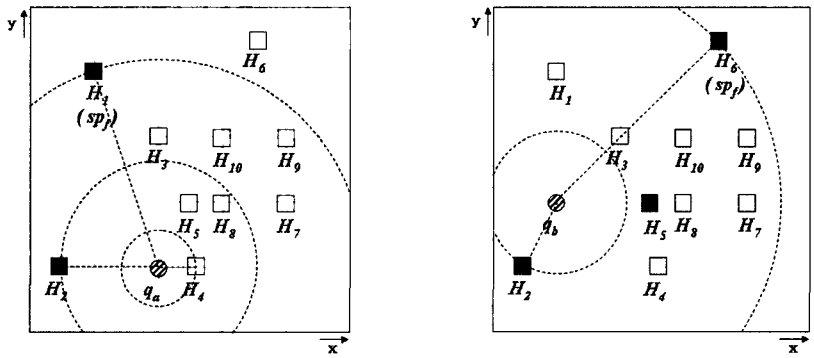
이 장에서는 먼저 *CSQMO* 기법과 이상적인 기법의 문제점을 분석하고, 제안하는 지배 객체 가지치기(*DOBP: Dominant Object-Based Pruning*) 기법을 이용하여 질의 시점에 효율적으로 SK_{chg}를 결정하는 방법을 설명한다.

4.1 *CSQMC*의 가지처기 기법

CSQMO 기법에서는 SK_{chg}를 계산하기 위해서 대상 객체들의 공간적인 속성을 그리드 파일로 색인한다. 이 기법에서는 SK_{ns}가 포함한 스카이라인 객체들 중 이동 객체로부터 가장 멀리 있는 객체를 sp_f라 하고, sp_f와의 거리 $dist(q, sp_f)$를 가지치기 기준으로 사용한다. 그리드 파일을 질의 점을 중심으로 나선형 탐색하며 P_{tee}의 객체와 이동 객체 사이의 거리 $dist(q, p_{tee})$가 $dist(q, sp_f)$보다 가까우면 후보 객체가 되고 SK_{chg} 포함 여부를 계산한다.

CSQMO 기법의 문제점은 그림 6과 같이 가지치기 기준으로 사용하는 $dist(q, sp_f)$가 대부분 너무 큰 값으로 설정되어 P_{tee}의 거의 모든 객체들을 P_{cand}가 포함하기 때문에 발생한다. P_{tee}의 거의 모든 객체들이 P_{cand}가 되면 실제 객체 외에 인덱스를 접근하는 추가적인 시간이 필요하게 되므로 순차적으로 접근하는 것보다 더 많은 시간을 소비하게 된다.

이러한 문제점의 원인은 sp_f가 P_{cand}의 모든 객체들을 지배하는 것은 아니며, sp_f보다 더 가까운 지배 객체가



(a) q_a's SK_{chg} = {H_4} (b) q_b's SK_{chg} = {∅}

그림 6 호텔들의 공간 속성 관계

있을 수도 있다는 점을 고려하지 않았기 때문에 발생한다. 예를 들어 그림 6의 (a)와 같이 SK_{ns}={H_1, H_2}일 때, 가지치기 기준을 H_1(=sp_f)로 선택하면 q_a의 P_{cand}={H_3, H_4, H_5, H_7, H_8, H_9, H_{10}}이다. 이 경우 H_4와 H_5는 sp_f가 지배하는 객체가 아니고 이들 객체들을 지배하는 이동 객체에 더 가까운 H_2를 기준으로 P_{cand}를 판별하는 것이 더 바람직하다. 최종 SK_{chg}는 H_4만 포함하지만 이 기법에서는 이런 문제점으로 인하여 나머지 6개의 후보 객체들에 대해서도 SK_{chg} 포함 여부를 판별하는 불필요한 연산을 수행하게 된다. 그림 6의 (b)와 같이 q_b의 경우에 이 기법을 적용하면 P_{cand}={H_1, H_3, H_4, H_7, H_8, H_9, H_{10}}이고, SK_{chg}는 이 중 어떤 객체도 포함하지 않는다.

4.2 이상적인 동적 부분차원 스카이라인 결정 기법

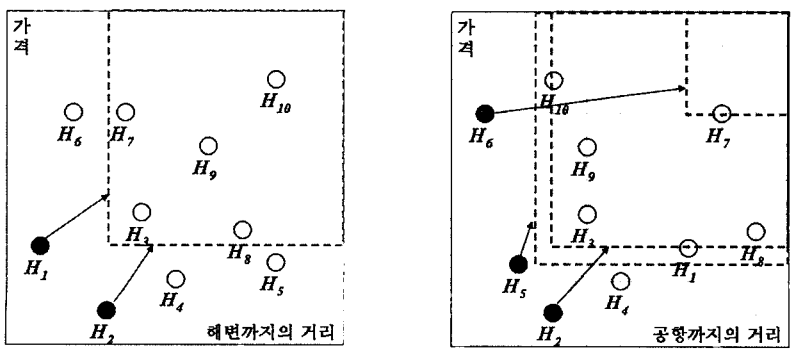
SK_{chg}를 계산하는 가장 이상적인 기법은 SK_{ns}가 아닌 현재 스카이라인 SK_{curr}의 SK_{ns}가 포함한 스카이라인 객체들 중 대상 객체를 지배하는 이동 객체로부터 가장 가까운 객체를 sp_n라 정의하고, sp_n까지의 거리 $dist(q, sp_n)$를 가지치기 기준으로 선택하는 것이다.

그림 7의 (a)에 이 기법을 적용하면 H_2가 sp_n이기 때문에 P_{cand}={H_4, H_5, H_8}이다. 게다가 SK_{curr}의 스카이라인 객체를 기준으로 하고, 거리가 가까운 순으로 대상 객체를 처리한다면 첫 번째 연산에서 H_4는 SK_{curr}에 포함되고 다음 연산에서는 H_4가 sp_n으로 선택하게 된다. 그러므로 두 번째 연산에서는 H_5와 H_8은 P_{cand}에서 제외되어 실제 1개의 객체(H_4)에 대한 연산만으로 SK_{chg}를 결정할 수 있다.

그러나 이러한 기법은 SK_{curr}의 객체들과 모든 대상 객체들 간의 지배 관계가 미리 계산되어 있다는 전제하에서만 가능한 방법이다. 부분차원을 질의 시점에 결정하는 경우, 모든 가능한 부분차원의 경우의 수만큼 전처리 과정에 통해 지배 관계를 미리 저장하는 것은 불가능하기 때문에 이 기법을 그대로 적용할 수 없다.

4.3 지배 객체 기반 가지치기 기법

제안하는 지배 객체 기반 가지치기(*DOBP : Dominant Object-Based Pruning*) 기법에서는 먼저 각 속성 값들을 대표하는 식 (1)의 1차원 $f(p)$ 값은 불변이므로 전처리 과정에서 미리 계산하고, 공간 속성을 색인하



(a) q_a의 지배 객체 (b) q_b의 지배 객체

그림 7 $f(p)$ 값을 이용한 부분차원 지배 객체 판별

는 R*-트리의 *MBR*에 저장한다. 그리고 질의 시점에 이동 객체 q를 중심으로 거리 순으로 R*-트리를 탐색하면서 계산된 $f(p)$ 값과 식 (2)의 조건을 이용하여 SK_{curr}에서 대상 객체의 부분차원 지배 객체들을 판별하고, 판별된 지배 객체들 중 sp_n을 선택함으로써 효율적인 가지치기를 수행한다.

그림 7은 제안한 기법의 부분차원 지배 객체를 결정하는 예이며 점선으로 처리된 영역은 식 (2)의 우변에 의해 정해진다. 그림에서 나타나는 바와 같이 대상 객체가 속한 영역에 따라 지배 객체를 결정할 수 있으며 그림 7의 (a)의 경우 영역 내에 위치하는 H_3, H_7, H_8, H_9, H_{10}은 각 $f(h)$ 값만으로도 지배 객체가 H_1과 H_2임을 판별할 수 있다. 나머지 영역 밖에 위치하는 대상 객체는 모든 부분차원 값을 알아낸 후 지배 객체를 판별해야 하지만 영역 밖의 객체는 상대적으로 적게 발생하는 것은 *SUBSKY* 기법[18]에서 이론적으로 증명하였다.

DOBP 기법은 대상 객체의 공간 속성을 R*-트리로 색인하고, 영역 내 객체들의 지배 객체를 판별하기 위해 각 인덱스 노드는 미리 계산된 자식 노드들의 $f(p)$의 최대값을 저장한다. *DOBP* 기법을 위한 R*-트리의 *MBR*과 인덱스 구조는 그림 8과 같다.

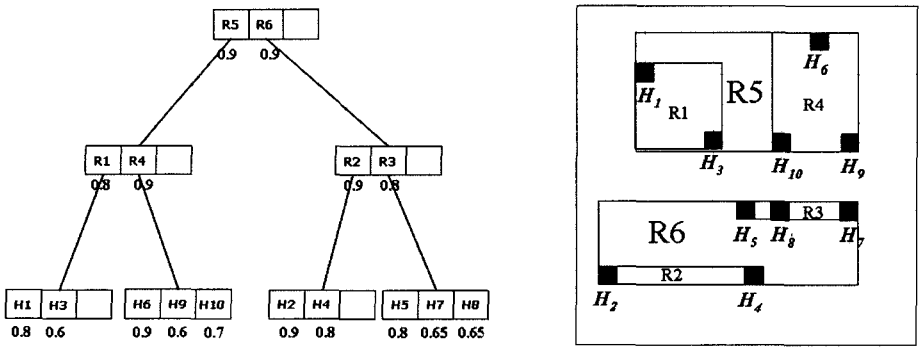
DOBP 기법에서는 효율적인 가지치기를 위해 우선순위 큐(priority queue)를 이용하여 R*-트리를 거리(*MinDist*) 순으로 탐색할 때 각 노드의 $f(p)$ 값을 비교하여 현재 노드의 하위 레벨로의 진행 여부를 결정한다. 방문한 노드의 $f(p)$ 값과 식 (2)의 조건을 이용하여 SK_{curr}의 스카이라인 객체들 중에서 부분차원 지배 객체를 판별하고, 판별된 지배 객체들 중에서 sp_n을 선택하여 $dist(q, sp_n)$가 $MinDist(q, MBR)$보다 가까우면 해당 노드는 더 이상 하위 레벨로 진행하지 않는다.

DOBP 기법을 이용하여 이동 객체의 부분차원 스카이라인 SK_{all}를 결정하는 알고리즘의 의사 코드는 그림 9와 같다. subspace_skyline 알고리즘의 10번째 줄에 있는 함수 *isCandidate(fh, dist)*는 SK_{curr}의 SK_{ns}뿐만 아니라 SK_{chg}의 스카이라인 객체들 중에서도 부분차원 지배 객체를 판별하여 sp_n을 결정하고, 이동 객체와의

```
Algorithm subspace_skyline(q, sub)
Input : q is the query point
        sub is subspace of interesting static dimension
Output : SK_all is the initial subspace skyline
         for q's staring position
1. Compute SK_ns using SUBSKY
2. d_bnd = max[dist(q,ns_i)], ns_i ∈ SK_ns
3. queue.add(rtree.readRoot(), -1)
4.  while queue is not empty
5.      e = queue.poll()
6.      if e instanceof INode
7.          for each child node nc
8.              fh = nc.getFH()
9.              dist = nc.getMinDist(q)
10.             if isCandidate(sub, fh, dist)&& d_bnd>dist
11.                 ce = rtree.getChildEntry(e,
                                    nc.childIndex)
12.                 queue.add(ce, dist)
13.             end if
14.         end for
15.     else
16.       if isSkyline(e)
17.         SK_chg.add(e)
18.       end if
19.   end if
20. end while
21. SK_all = SK_ns + SK_chg
22. return SK_all
```

```
Algorithm isCandidate(sub, fh, dist)
Input : sub is subspace of interesting static dimension
        fh is f(p) of the entry
        dist is the distance from query
Output : true/false: whether is candidate
1. Set SK_curr = SK_ns + SK_chg
2. sp_n = min[dist(q,sp_i)], sp_i ∈ SK_curr ∧
          fh < [min(1 − sp_i[j]),  j ∈ sub]
3. if dist > sp_n return false
4. return true
```

그림 9 이동 객체의 부분차원 스카이라인 알고리즘



(a) Index 구조 (b) MBR 구조

그림 8 *DOBP*를 위한 R*-트리

표 2 q_a의 SK_{chg} 결정 단계별 큐의 내용

Action	Queue content <*node, MinDist, f(p)*>	Description
Access Root	<R_6, 0, 0.9> <R_5, 0.40, 0.9>	
Expand R_6	<R_2, 0, 0.9> <R_3, 0.22, 0.8> <R_5, 0.40, 0.9>	
Expand R_2	<H_4, 0.10, 0.8> <R_3, 0.22, 0.8> <R_5, 0.40, 0.9>	$H_2 \in SK_{ns}$
Object H_4	<R_3, 0.22, 0.8> <R_5, 0.40, 0.9>	H_4's $sp_n = H_2$: → $H_4 \in SK_{chg}$
Expand R_3	<R_5, 0.40, 0.9>	(1) {H_5, H_8}'s $sp_n = H_4$ → *pruning* (2) H_7's $sp_n = H_2$ → *pruning*
Exapnd R_5	<R_1, 0.40, 0.8> <R_4, 0.45, 0.9>	
Expand R_1	<R_4, 0.45, 0.9>	(1) $H_1 \in SK_{ns}$ (2) H_3's $sp_n = H_2$ → *pruning*
Expand R_4		(1) H_6's $sp_n = H_2$ → *pruning* (2) {H_9, H_{10}}'s $sp_n = H_4$ → *pruning*

거리 $dist(q, sp_n)$보다 이동 객체에서 가까운 대상 객체를 후보 객체로 선택한다.

거리 순으로 대상 객체를 탐색하기 때문에 SK_{chg}의 스카이라인 객체들은 자신을 지배하는 SK_{ns}의 스카이라인 객체보다 항상 이동 객체에 가깝다. 그러므로 SK_{curr}의 스카이라인 객체 중 가장 가까운 지배 객체 sp_n를 선택한다면 가지치기의 기준이 되는 $dist(q, sp_n)$는 점점 가까워져 접근해야할 후보 객체의 개수는 점진적으로 감소한다. 표 2는 질의 예제 q_a의 동적 속성에 대한 부분차원 스카이라인 SK_{chg} 결정 과정의 단계별 우선순위 큐의 내용을 나타낸다.

5. 실험 및 성능 평가

5.1 실험 환경

이 논문에서 제안한 이동 객체의 부분차원 스카이라인질의를 처리하기 위한 지배 객체 기반 가지치기 기법의 성능을 평가하기 위해 대상 객체의 모의 데이타를 생성하여 실험하였다. 실험에서 정적 부분차원 스카이라인의 결정 성능은 모두 *SUBSKY*에서 제안한 기법으로 이용하는 경우 모두 동일하기 때문에 성능 평가에서는 제외하고 동적 부분차원 스카이라인 결정 성능의 비교 실험을 수행하였다. 그리고 가장 간단하게 생각할 수 있는 순차 탐색(*SEQ*), 정적 스카이라인 객체를 이용한 가지치기 기법(*CSQMO*), 그리고 제안한 지배 객체 기반 가지치기 기법(*DOBP*)을 대상으로 비교 실험을 통한 성능 평가를 수행하였다.

5.1.1 실험 환경

비교 실험을 통한 성능 평가를 위해 모든 기법은 동일하게 Pentium-IV 2.4GHz 프로세서와 1GB의 메인 메모리, 80GB의 하드 디스크를 가진 Windows 2000 운영체제의 PC에서 자바언어(JSDK 1.5)를 이용한 환경에서 구현하고 실험을 수행하였다.

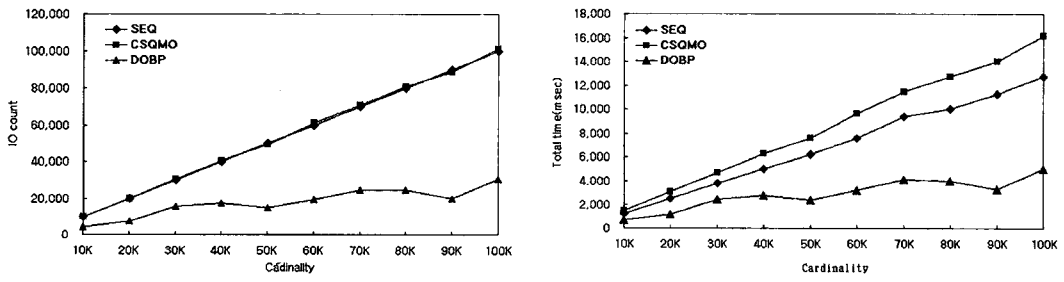
5.1.2 실험 데이타

비교 실험을 위해 사용하는 데이타는 다양한 실험을 위하여 대상 객체는 10차원의 정적 속성에 대해 균등(uniform), 반상관(anti-correlated) 분포하고, 공간 속성은 균등 분포하는 10K에서 100K 크기의 모의 데이타를 사용하여 성능 비교 실험을 수행하였다.

5.2 성능 평가

5.2.1 대상 객체의 개수 변화에 따른 성능 비교

정적 속성에 대해 균등 분포하고, 부분차원은 3차원으로 고정하고 대상 객체의 개수를 10K에서 100K로 변화하면서 수행한 각 기법의 성능 비교 그래프는 그림 10과 같다.



(a) IO count (b) Total time (msec)

그림 10 대상 객체의 개수 변화에 따른 성능 비교

실험 결과 *DOBP* 기법은 *CSQMO* 기법보다 IO는 최소 48%에서 최대 77% 감소하였고, 전체 수행 시간은 최소 47%에서 최대 76%까지 감소하였다. 또한 대상 객체 개수가 증가함에 따라 *CSQMO*는 연산 시간이 현저하게 증가하는 반면 *DOBP* 기법은 상대적으로 적은 연산 시간의 증가를 보였다.

5.2.2 부분차원 변화에 따른 성능 비교

정적 속성에 대해 균등 분포하고, 대상 객체의 개수를 100K로 고정하고 부분차원은 2차원에서 5차원까지 변화하면서 수행한 각 기법의 성능 비교 그래프는 그림 11과 같다.

이 실험에서 차원이 증가하면 SK_{ns}의 증가로 인해 $dist(q, sp_f)$는 점점 더 멀어지므로 *CSQMO* 기법은 거의 모든 대상 객체를 포함하게 되어 3차원 이상에서는 오히려 *SEQ*보다 더 많은 수행시간을 소비했다. *DOBP* 기법도 부분차원이 증가하면 연산 시간이 증가하지만 다른 기법들에 비해 향상된 성능을 나타냈다.

5.2.3 데이타 분포에 따른 성능 비교

부분차원은 3차원으로 고정하고, 대상 객체의 개수를 10K에서 100K로 변화하면서 정적 속성 값이 균등(uniform) 분포, 반상관(anti-correlated) 분포하는 경우, 각 기법의 실험 결과는 그림 12와 같다. 이 실험에서 순차 검색은 데이타 분포와 성능은 무관하기 때문에 제외하였고, *CSQMO*나 *DOBP* 기법의 성능은 두 기법 모두 데이타 분포보다는 객체의 개수에 의해 크게 변화함을 보였다.

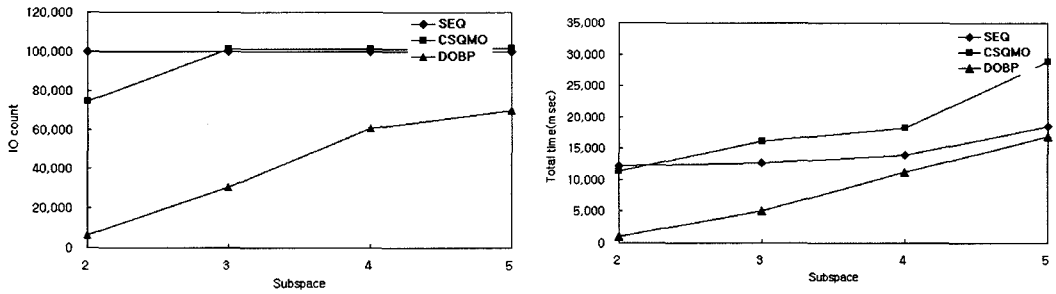
6. 결 론

이동 객체의 연속적인 스카이라인 질의에 대한 이전 처리 기법들에서는 부분차원 스카이라인 질의를 고려하지 않았기 때문에 사용자는 미리 정해진 조건에 대해서만 질의할 수 있었다.

이 논문에서는 이전 기법들의 문제점을 분석하고, 이를 해결할 수 있도록 질의 시점에 이동 객체의 부분차원 스카이라인을 효율적으로 계산할 수 있는 지배 객체 기반 가지치기(*DOBP*) 기법을 제안하였다.

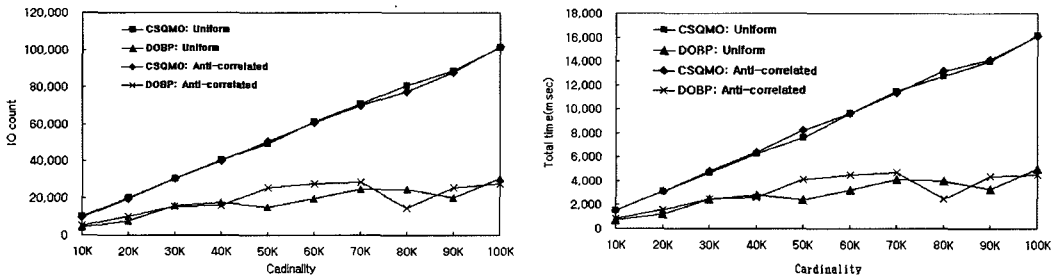
모의실험을 통하여 성능 평가를 수행한 결과, 제안한 기법은 *CSQMO* 기법에 비해 IO는 최대 77% 감소하였고, 전체 수행 시간은 76% 감소하는 성능 향상을 보였다. 또한 *CSQMO* 기법은 대상 객체의 개수와 부분 차원 변화에 따라 *SEQ* 기법보다도 저하된 성능을 보인 반면에 제안한 기법은 모든 실험에서 성능 향상을 보였다.

향후 이 논문에서는 고려하지 않은 이동 객체의 연속 질의 처리를 위한 유효 영역을 질의 시점에 결정할 수 있는 기법에 대한 연구를 진행하여 제안하는 기법과 통합하면 다양한 위치 기반 서비스에 응용될 수 있을 것으로 기대된다.



(a) IO count (b) Total time (msec)

그림 11 부분차원 변화에 따른 성능 비교



(a) IO count (b) Total time (msec)

그림 12 데이타 분포에 따른 성능 비교

참 고 문 헌

[1] S. Borzsonyi, D. Kossmann, K. Stocker, "The Skyline Operator," IEEE ICDE, pp. 421-430, 2001.

[2] K. Tan, P. Eng, B. Ooi, "Efficient Progressive Skyline Computation," VLDB, pp. 301-310, 2001.

[3] J. Chomicki, P. Godfrey, J. Gryz, D. Liang, "Skyline With Presorting," Technical Report, York University, 2002.

[4] D. Kossmann, F. Ramsak, S. Rost, "Shooting Stars in the Sky: an Online Algorithm for Skyline Queries," VLDB, pp. 275-286, 2002.

[5] D. Papadias, Y. Tao, G. Fu, B. Seeger, "An Optimal and Progressive Algorithm for Skyline Queries," SIGMOD, pp. 443-454, 2003.

[6] J. Pei, W. Jin, M. Ester, Y. Tao, "Catching the Best Views of Skyline: A Semantic Approach Based Decisive Subspaces," VLDB, 2005.

[7] P. Godfrey, R. Shipley, J. Gryz, "Maximal Vector Computation in Large Data Sets," VLDB, 2005.

[8] D. Papadias, Y. Tao, G. Fu, J. M. Chase, B. Seeger, "Progressive Skyline Computation in Database Systems," ACM TODS, Vol.30(1), pp. 41-82, 2005.

[9] Y. Yuan, X. Lin, Q. Liu, W. Wang, J. X. Yu, Q. Zhang, "Efficient Computation of the Skyline Cube," VLDB, 2005.

[10] R. H. Guting, M. H. Bohlen, M. Erwig, C. S. Jensen, N. A. Lorentzos, M. Schneider, M. A. Vazirgiannis, "A Foundation for Representing and Quering Moving Objects," ACM TODS, Vol.15 (12), pp. 905-910, 2000.

[11] L. Forlizzi, R. H. Guting, E. Nardelli, M. Schneider, "A Data Model and Data Structures for Moving Objects Databases," ACM SIGMOD, 2000.

[12] Y. Theodoridis, J. R. O. Silva, and Mario A. Nascimento, "On the Generation of Spatiotemporal Datasets," Proceedings of the 6th International Symposium on Large Spatial Database, 1999.

[13] E. Pitoura, G. Samaras, "Locating Objects in Mobile Computing," IEEE TKDE, Vol.13(4), pp. 571-592, 2001.

[14] Z. Huang, B. C. Ooi, "Continuous Skyline Queries for Moving Objects," IEEE TKDE, Vol.18, No.12, 2006.

[15] 나경석, 김진호, 안대혁, 박영배 "연속적인 스카이라인 질의를 위한 효율적인 영역 결정기법", 정보과학회 데이타베이스연구, 제22권, 제2호, 2006.

[16] 이종혁, 박영배, "연속적인 스카이라인 질의의 정적 유효 영역을 이용한 효율적인 처리", 정보과학회논문지: 데이타베이스, 제33권 제6호, 631-643, 2006.

[17] M. Sharifzadeh, C. Shahabi, "The Spatial Skyline Queries," VLDB, pp. 751-762, 2006.

[18] Y. Tao, X. Xiao, J. Pei, "SUBSKY: Efficient Computation of Skylines in Subspaces," IEEE ICDE, 2006.

김 진 호

1994년 군산대학교 전자공학과(공학사) 2000년 명지대학교 대학원 컴퓨터공학과(공학석사). 2007년 명지대학교 대학원 컴퓨터공학과(공학박사). 관심분야는 Mobile DB, Spatial DB, Data Stream Management, Sensor Network

박 영 배

1993년 서울대학교 대학원 컴퓨터공학과(공학석사). 1990년~1992년 명지대학교 전자계산소장. 1981년~현재 명지대학교 컴퓨터공학과 교수. 관심분야는 Mobile DB, Spatial DB, 한국어 정보처리, 대용량 지문 DB