도로망에서 연속적인 스카이라인 질의처리를 위한 효율적인 전처리기법

(An Efficient Pre-computing Method for Processing Continuous Skyline Queries in Road Networks)

장 수 민 † 유 재 수 ††
(Sumin Jang) (Jaesoo Yoo)

요 약 최근에 검색서비스에서 스카이라인 질의 처리는 상당한 주목을 받고 있다. 스카이라인은 모든 속성들 측면에서 다른 객체에 지배되지 않는 관심 있는 객체들을 포함한다. 이에 관련된 기존 많은 연구들은 정적 데이터에 대한 스카이라인이거나 유클리디언 공간상에서 움직이는 대상에 대한 스카이라인 처리이다. 그러나 본 논문은 스카이라인 질의를 요청한 객체가 도로망에서 연속적으로 이동하는 것을 가정한다. 본 논문은 전처리된 객체들의 최단영역을 통하여 도로망에서 효과적인 연속적 스카이라인 질의처리를 위한 새로운 처리 기법을 제안한다. 제안하는 기법은 검색 대상에 대한 미리 연산된 최단거리 범위 데이터를 통하여 연속 스카이라인을 처리한다. 성능평가는 질의처리속도 측면에서 제안하는 기법이 기존방법보다 약 100배 정도 빠르다는 것을 보여준다.

키워드 : 연속적 스카이라인 질의, 스카이라인 질의처리, 위치기반서비스, 이동 객체, 도로망

Abstract Skyline queries have recently received considerable attention in the searching services. The skyline contains interesting objects that are not dominated by any other objects on all dimensions. Many related works have processed a skyline on static data or on moving objects in Euclidean space. However, this paper assumes that the point of a skyline query continuously moves in road networks. We propose a new method that efficiently processes continuous skyline queries in road networks through pre-computed shortest range data of objects. Our experiments show that the proposed method is about 100 times faster than previous methods in terms of query processing time.

Key words : continuous skyline query, processing skyline query, location-based service, moving objects, road networks

1. 서 론

데이터베이스에서 스카이라인 질의처리는 Borzonyi에 의해서 소개되었다[1]. 스카이라인 질의(Skyline Query)는 전체 객체집합에서 대상 객체의 여러 속성을 다른 객체가 지배(Dominate)하지 않는 관심있는 객체집합을 검색한다. 여기서 '지배'라는 정의는 다음과 같다. 두 개의 객체 O_1과 O_2가 있을 경우, $O_1=(x_1,x_2,...,x_d)$, $O_2=(y_1,y_2,...,y_d)$와 같이 d개의 여러 속성을 가지고 있고, 만약 $\forall_i$, $x_i \leq y_i$, 이고 $\exists_j$, $x_j<y_j$ $(1\leq i, j\leq d)$을 만족한다면 O_1이 O_2를 지배한다. 다시 말하자면, 어떤 속성에서나 O_2가 O_1보다 우수한 값이 없을 경우 O_1이 O_2를 지배하는 것이다.

그림 1은 질의의 대상이 되는 객체가 호텔이고 그 호텔들과 해변의 거리와 호텔의 숙박료라는 두 가지 속성을 갖고 있다. 이러한 호텔에 대하여 스카이라인 질의는 "해변에서 가깝고 숙박료가 싼 호텔들을 검색하라." 이다. 질의의 결과에 해당하는 호텔은 호텔들 간에 서로 비교를 통하여 특정호텔에 비하여 해변에서 멀고 숙박료도 비싼 호텔들을 제거한 후 남아있는 호텔들이다. 호

· 이 논문은 2009년 지식경제부의 특구연구개발사업으로부터 지원과 2009년 교육과학기술부로부터 지원받아 수행된 연구임(지역거점연구단육성사업/충북BIT연구중심대학육성사업단)

† 정 회 원 : 충북대학교 정보통신공학과 박사 후 연구원
jsm@cbnu.ac.kr
†† 종신회원 : 충북대학교 정보통신공학과 교수
yjs@cbnu.ac.kr
(Corresponding author임)

논문접수 : 2008년 11월 28일
심사완료 : 2009년 6월 22일

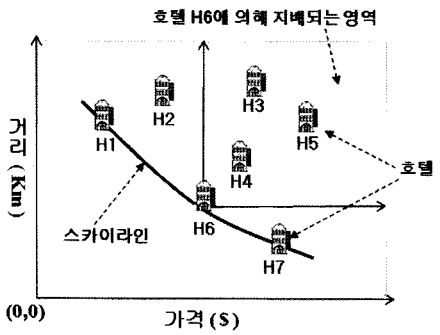

그림 1 스카이라인 질의의 예제

텔 H6은 거리나 가격측면에서 호텔 H3, H4, H5를 지배한다. 이처럼 지배되는 호텔들을 모두 제거하면 호텔 H1, H6, H7만이 남게 된다. 이를 스카이라인이라 한다.

이전의 스카이라인 질의를 처리하는 기법들로는 정적인 속성만을 고려한 기법들과 동적인 속성을 고려한 기법들로 나누어진다. 정적인 속성만을 고려한 기법들[1-5] 중에 대표적인 것으로 BNL(Block Nested Loop) 기법[4]와 D&C(Divide-and-Conquer)기법[2]가 제안되었고, 비교 횟수를 줄이기 위해 Bitmap과 B-tree 색인을 이용한 기법[6]이 제안되었다. 최근에는 R-tree기반의 NN(Nearest Neighbor) 검색하는 방법을 이용한 BBS (Branch-and-Bound Skyline)기법[5]가 제안되었다. 동적인 속성을 고려한 기법으로는 KDS(kinetic data structures)이라는 자료구조를 이용하여 특정시간에 변환의 가능성을 예측하고 순차적으로 처리하는 CSQ라는 기법이 제안되었다[7]. 또한 스카이라인 질의를 웹서비스[8]나 P2P환경[9]와 같은 다양한 분야로 확대하여 연구하고 있으며, 최근에는 스트림데이터에서 대한 동적 스카이라인 질의처리[10,11]도 연구되고 있다. 그러나 기존의 제안된 기법들은 동적인 속성에 대한 값을 측정하고 예측하는데 많은 처리시간을 필요로 하는 문제점을 갖고 있다.

본 논문에서 처리하고자 하는 서비스의 형태는 도로망에서 자동차와 같은 이동객체가 스카이라인 질의를 발생시키고 지속적으로 도로망을 따라 이동하면서 스카이라인 질의의 결과를 지속적으로 요구하는 환경이다. 제안하는 기법은 검색 대상에 대한 미리 연산된 최단거리 범위 데이터를 통하여 연속 스카이라인 질의를 빠르게 처리하는 새로운 처리 기법이다. 제안하는 기법은 주유소 추천서비스나 맛집 추천서비스 등과 같은 위치기반서비스에서 빠른 질의처리를 위해 활용될 수 있다. 그러나 본 논문에서 고려하는 동적인 속성은 도로망에서 고정된 객체와 이동하는 객체간의 거리로 제한한다. 즉, 도로망의 상태변화나 교통사고로 인한 주행속도변화에 따른 동적 가중치는 본 논문에서 고려하지 않는다.

본 논문의 구성은 다음과 같다. 2장에서 도로망에서 연속적인 스카이라인 질의처리의 문제점을 제시하고 3장에서는 본 논문에서 제안하는 최단영역(Shortest Range)을 이용한 전처리 기법을 설명한다. 마지막으로 4장에서 성능평가를 통하여 질의처리속도 측면에서 제안하는 기법이 기존기법보다 효율적임을 입증한다.

2. 도로망에서 연속적인 스카이라인 질의처리의 문제점

도로망에서 연속적 스카이라인 질의처리는 스카이라인 질의를 요청한 객체가 도로망을 따라 이동하면서 그 결과를 지속적으로 요구하는 것이다. 본 논문에서는 스카이라인 질의를 요청한 객체를 질의객체로 정의하고 스카이라인 질의의 대상이 되는 객체를 질의대상객체로 정의한다. 또한 질의대상객체의 데이터 속성들 중에 정적인 속성만을 고려하여 스카이라인을 구한 결과를 정적 스카이라인으로 정의한다.

본 논문에서 제안하는 스카이라인 질의처리는 동적인 속성과 정적인 속성을 동시에 고려한다. 그림 2는 도로망에서 연속적 스카이라인 질의의 형태를 보여준다. 여기서 질의객체는 "나의 위치로부터 가까우면서 가격이 싸고 등급이 좋은 호텔을 제시해 달라."라는 스카이라인 질의를 요청한다. 이때, 호텔의 가격과 등급은 변하지 않는 데이터로 정적인 속성에 해당된다. 그리고 호텔들과 질의객체의 거리는 계속해서 변하는 데이터로 동적인 속성에 해당된다. 유클리디언 환경에서 질의대상객체(호텔)와 질의객체의 거리는 각각 객체들의 위치정보 O1(x1,y1), O2(x2,y2)를 이용하여 아래와 같은 식 (1)과 같이 간단히 계산된다.

$$D = \sqrt{(x2-x1)^2+(y2-y1)^2} \quad (1)$$

그러나 도로망 환경에서 객체들 간의 거리는 객체들을 연결해주는 모든 길들 중에 가장 짧은 도로망에서

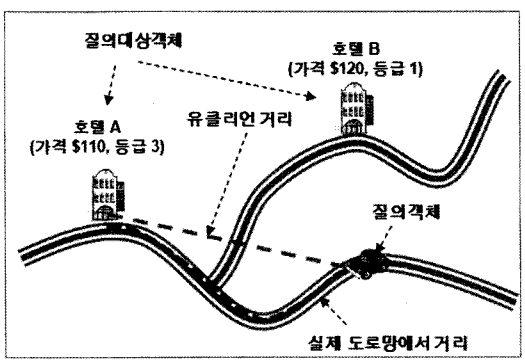

그림 2 도로망에서 연속적 스카이라인 질의의 형태

거리를 연산해야 함으로 매우 복잡하고 계산비용이 크다. 또한 질의객체가 이동할 때마다 동적인 속성 값은 재연산해야 한다. 이처럼 도로망에서 질의객체와 질의대상객체들 간의 거리를 구하는 연산비용이 매우 크기 때문에 그 비용을 줄이기 방법으로 필터링기법을 사용한다. 우선적으로 질의대상객체들 중에서 정적 스카이라인을 구한다.

그리고 그 정적 스카이라인에 해당하는 객체들과 질의객체간의 거리를 고려하여 특정거리영역을 설정하고 그 영역을 벗어나는 객체를 스카이라인 질의의 비교대상에서 제거하는 방법을 사용한다. 그림 3은 도로망에서 연속적 스카이라인 질의를 부분적으로 필터링하여 처리하는 과정을 보여준다. 첫 번째로, 그림 3(a)과 같이 우선적으로 정적인 속성인 호텔 가격과 등급만을 고려하여 정적 스카이라인을 구한다. 그 정적 스카이라인에 해당하는 호텔들(호텔 A, B, C)과 질의객체의 거리를 모두 계산하고 이중에서 가장 먼 거리 D1을 설정한다. 그런 다음, 그림 3(b)와 같이 질의객체의 위치로부터 설정된 거리 D1만큼 도로망을 확장하고 정적 스카이라인을 제외한 확장된 그 영역안에 포함되는 호텔(호텔 K)을 찾는다. 그 확장된 거리 D1보다 먼 호텔(호텔 T)은 일단 정적인 속성 측면에서 정적 스카이라인에 지배되고 거리 속성 측면에서도 지배되기 때문에 최종 스카이라인에 해당되지 않는다. 그래서 최종 스카이라인은 정적 스카이라인에 해당하는 호텔들과 거리 D1의 확장된 범위에 포함되는 호텔들로 한정될 수 있다. 그리고 그 한정된 객체들 간의 모든 속성에 대한 지배여부를 비교하여 최종 스카이라인을 결정한다. 그러나 이러한 필터링기법도 객체들 간의 거리를 계산하는 비용과 그 범위에 해당하는 호텔들 간에 지배여부를 검사하는 비용이 필요하다.

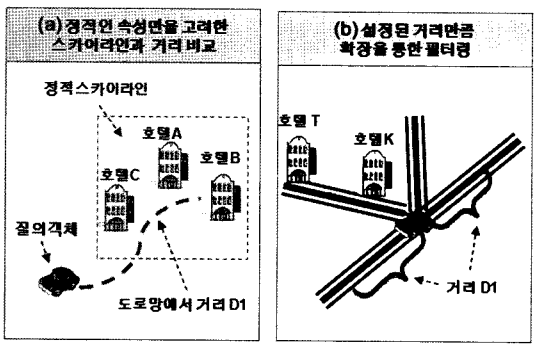

그림 3 도로망에서 스카이라인 질의처리를 위한 필터링 과정

3. 최단영역(Shortest Range)을 이용한 전처리 기법

도로망에서 동적인 속성을 고려한 연속적 스카이라인 질의 처리는 동적인 데이터에 대한 재계산이 필요하기 때문에 질의 처리 속도가 매우 느린 문제점을 가지고 있다. 그래서 본 논문에서는 질의대상객체의 최단영역이라는 것을 참조하여 효율적으로 스카이라인 질의를 처리하는 새로운 기법을 제안한다. 질의대상객체 Ti의 최단영역은 객체 Ti가 질의대상객체의 속성들 중에 정적인 속성이 아닌 동적인 거리 정보만 고려하여 스카이라인에 포함되는 최대영역을 의미한다. 이는 질의대상객체의 정적인 속성들의 값에 상관없이 거리속성만으로 다른 질의대상객체들에게 지배당하지 않는 영역을 의미한다. 본 논문에서는 질의대상객체들의 모든 최단영역을 미리 계산하고 정보화한다. 전처리된 질의대상객체들의 최단영역을 이용한 스카이라인의 결과는 정적 스카이라인에 해당하는 객체들과 질의객체의 현 위치와 관련된 최단영역 객체들의 합집합이다. 이처럼 전처리된 최단영역 정보를 통하여 연속적 스카이라인 질의처리는 매우 간단하다. 본 논문에서 도로망을 구성하는 요소들을 다음과 같이 정의한다. 도로망에서 도로에 해당하는 선과 선의 교차되는 지점은 노드(Node)로 정의하고, 노드와 노드를 연결하는 선을 에지(Edge)로 정의한다. 질의대상객체들의 최단영역 정보는 도로망을 구성하는 에지들에 저장된다. 그 생성절차는 다음과 같다.

① 스카이라인 질의대상객체들(T1~Tn)에서 객체 Ti ($1 \le i \le n$, i는 정수)을 선택한다.

② ①단계에서 선택된 객체 Ti가 갖는 정적 속성 값(들)에 지배되지 않는 객체집합 $S=\{Tj \mid j \neq i, 1 \le j \le n$, j는 정수$\}$를 선정한다.

③ 선택된 객체 Ti는 도로망에서 ②단계에서 선정된 집합 S의 객체들과 거리측면에서 상대적으로 가까운 영역, 즉 객체 Ti의 최단영역을 결정한다.

④ ③단계에서 결정된 객체 Ti의 최단영역과 관련된 에지(Edge)들에 그 최단영역 정보를 저장한다.

⑤ 질의대상객체들(T1~Tn) 중에 ①단계에서 선택되지 않은 Ti가 있다면 ①단계부터 다시 시작한다.

그림 4에서 그림 6까지는 위에서 설명한 질의대상객체들(호텔 A, B, C)의 최단영역을 생성하여 에지들에 저장하는 과정을 순서적으로 보여준다. 최단영역을 생성하는 순서는 호텔 A, 호텔 B 순이다. 그러나 호텔 C는 생략하였다. 첫 번째로, 그림 4는 도로망에서 호텔 A의 최단영역을 생성하는 예제이다. 앞에서 설명한 최단영역의 생성하는 절차와 같이, 호텔 A를 선택하고, 선택된 호텔 A의 정적인 속성 값들에 의해서 지배되지 않는 호텔들을 선정한다. 이때, 그림 4의 예제는 정적인 속성 값 측면에서 호텔 A가 호텔 C를 지배하고 호텔 B를 지배하지 못하는 조건이다.

그래서 지배되는 호텔 C는 호텔 A의 최단영역을 생

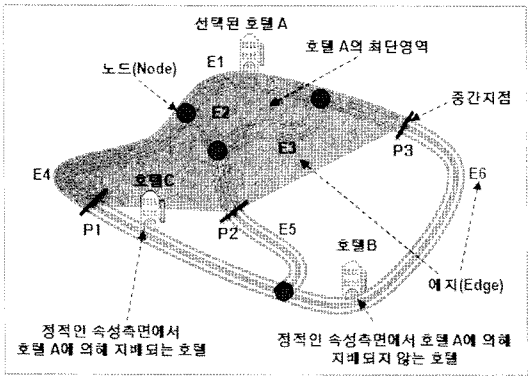

그림 4 호텔 A의 최단영역

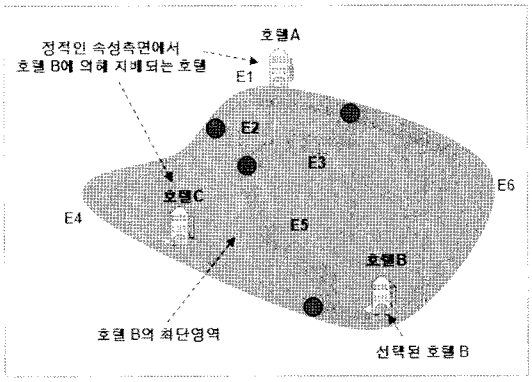

그림 5 호텔 B의 최단영역

성하는 과정의 비교대상에서 제거된다. 그리고 선택된 호텔 A는 지배되지 않는 호텔 B와 상대적인 도로망에서 거리를 비교하여 자신의 최단영역을 결정한다. 호텔 A의 최단영역은 중간지점들(P1, P2, P3)의 위부분에 해당하는 영역이다. 그 최단영역은 에지의 전체적 범위 해당하는 에지들(E1, E2, E3)과 에지의 일부분에 해당하는 에지들(E4, E5, E6)로 구성된 것을 볼 수 있다. 이러한 최단영역의 정보는 질의대상객체들을 구분하기 위한 호텔의 ID와 범위정보를 관련된 에지들(Edge)에 저장된다. 다음 단계로, 그림 5는 다음으로 선택된 호텔 B의 최단영역을 생성과정을 보여준다. 우선, 선택된 호텔 B의 정적인 속성 값들에 지배되지 않는 호텔들을 구해야 한다. 그런데 그림 5의 예제는 호텔 A와 호텔 C가 호텔 B의 정적인 속성 값들에 모두 지배되는 조건이다. 그래서 호텔 A와 호텔 C는 호텔 B의 최단영역 비교대상에서 모두 제거되어 호텔 B는 비교대상이 없게 된다. 즉, 호텔 B의 최단영역은 전체범위에 해당한다. 그리고 이러한 호텔 B의 최단영역 정보는 호텔 A의 최단영역 정보와 함께 추가하여 저장된다. 이러한 과정은 마지막 남은 호텔 C의 최단영역 정보까지 추가한다. 제안하는 기법은 이처럼 모든 질의대상객체들의 최단영역 정보를 갖는 데이터를 참조하여 질의를 처리한다.

그림 6은 그림 4와 그림 5에서 생성한 객체들의 최단영역 정보들을 에지단위로 저장된 것을 보여준다. 그림 6(a)는 호텔 A의 최단영역 정보만을 에지들에 반영한 것이고, 그림 6(b)는 호텔 A의 최단영역 정보에 호텔 B의 최단영역 정보를 추가적으로 반영한 것이다. 호텔 A의 최단영역만을 에지들에 반영한 그림 6(a)의 범위 R1에 해당하는 호텔은 호텔 A로 하나이다. 그러나 호텔 B의 최단영역이 전체영역이기 때문에, 그림 6(b)의 범위 R1에 해당하는 호텔은 호텔 A와 호텔 B로 두 개이고, 그림 6(b)의 범위 R2에는 호텔 B가 포함된다. 이처럼 에지들에 저장되는 호텔의 ID가 겹치게 된다. 그 이유는 다음과 같다. 최단영역을 구하기 위해서 선택된 객체

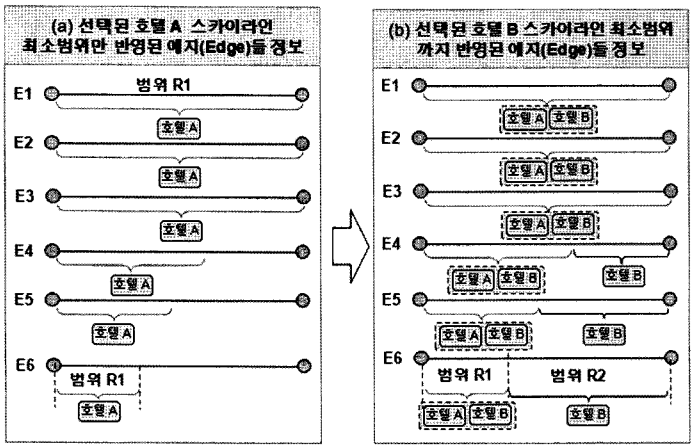

그림 6 에지단위로 저장된 객체들의 최단영역 정보

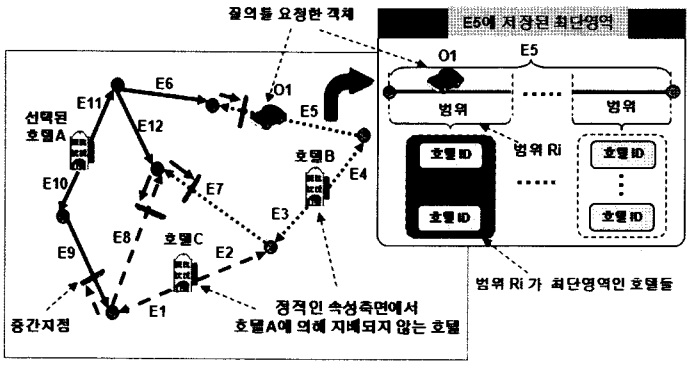

그림 7 최단영역을 생성을 위한 중간지점의 선정과정

와 정적인 속성 값 측면에서 그 선택된 객체에게 지배되지 않는 객체들이 서로 다르게 선정되기 때문에 각각의 최단영역들이 겹칠 수 있다. 에지단위로 이러한 정보를 저장하기 때문에 특정 에지위에서 이동하는 객체가 스카이라인 질의를 요청하면 그 객체의 위치에 해당하는 에지를 찾고 그 에지에 저장된 최단영역 정보를 참조하여 처리한다. 각 객체 O1의 최단영역을 결정하기 위해서는 도로망의 거리측면에서 비교대상인 객체들과 비교하여 객체 O1이 가장 가까운 대상이 되는 영역을 완성해야 한다.

그림 7은 최단영역을 생성하기 위한 중간지점의 선정과정을 보여준다. 호텔 A의 최단영역을 생성하는 과정이다. 그리고 호텔 B와 호텔 C는 정적인 속성측면에서 선택된 호텔 A에 지배되지 않는 조건이다. 그래서 호텔 B와 호텔 C는 선택된 호텔 A의 비교대상이 되고 객체의 최단영역을 생성하기 위한 선택된 호텔 A를 포함하여 에지확장의 시작지점이 된다. 첫 번째로 그 시작지점(호텔 A, B, C)들과 연결된 에지들(E1, E2, E3, E4, E10, E11)을 확장하기 위한 후보 집합에 포함시키고, 이들 에지들 중에 시작지점과의 거리가 가장 짧은 에지 순으로 확장한다. 그렇게 확장된 에지에 (시작지점의 호텔 ID, 시작지점과의 거리)와 같은 형태로 임시적으로 저장한다. 이러한 에지확장 순서는 최단거리 알고리즘으로 잘 알려진 다익스트라 알고리즘(Dijkstra algorithm)과 매우 유사하다. 그러나 본 논문에서 제안하는 방법과 다익스트라 알고리즘의 에지를 확장하는 순서의 차이점은 시작지점이 한 개가 아닌 다수 개이고 최종목표지점이 없이 모든 에지를 확장하는 것이다. 한 에지의 확장이 결정되면 그 에지의 노드에 연결된 다른 에지들을 다시 후보 집합에 포함하여 가장 거리가 짧은 에지를 다시 선택하여 확장한다. 이때, 이전 시작지점으로부터 미리 확장된 에지를 다른 시작점에서 확장할 경우, 그 에지에 이전에 유지된 (시작지점의 호텔 ID, 시작지점과의 거리)의 정보를 참조하여, 각각의 시작지점까지의 거리 값들을 비교하여 중간이 되는 지점을 선정한다. 그리고 관련된 에지들에 그 중간지점을 정보화하여 저장한다. 그러한 중간지점들을 이용하여 그림 6과 같은 처리를 한다면 각 에지들은 그림 7의 에지 E5의 최단영역과 같이 다수개의 범위영역들과 그 범위영역에 포함된 호텔들의 정보를 유지한다. 이러한 정보를 이용하여 이동객체에 대한 연속적 스카이라인 질의를 처리한다. 만약 객체 O1이 연속적 스카이라인 질의를 요청한다면, 자신의 위치정보를 통하여 그 위치에 해당하는 에지 E5의 범위 Ri를 결정하는 것만으로 질의처리가 끝난다. 즉, 객체 O1의 현 위치에 해당하는 범위 Ri에 저장된 호텔 ID들과 정적 스카이라인으로 설정된 호텔 ID들의 합집합이 요청한 스카이라인이 된다.

그림 8은 질의대상객체들의 최단영역들을 저장하기 위한 자료구조를 보여준다. 도로망에서 노드들과 에지들의 연결정보는 매트릭스형태로 유지된다. 그리고 그 매트릭스를 구성하는 셀은 에지 ID와 그 에지의 분할된 범위들을 저장하기 위한 연결리스트의 링크정보를 포함한다. 그리고 객체의 최단영역을 생성하는 과정에서 중

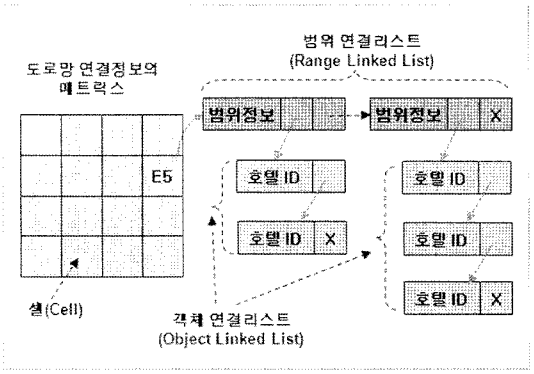

그림 8 최단영역을 저장하기 위한 자료구조

간지점이 추가되면서 생성되는 범위들을 저장하는 범위 연결리스트(Range Linked List)와 그러한 범위를 최단 영역으로 갖는 객체들을 저장하는 객체 연결리스트 (Object Linked List)로 구성된다. 이처럼 제안하는 기법은 전처리된 최단영역의 정보를 참조하기 때문에 동적인 속성의 데이터에 대하여 지속적인 재연산하는 기존기법보다 질의처리속도가 빠르다.

4. 성능평가

본 논문의 성능평가는 LINUX 운영체계, 펜티엄IV 2.0GHz, 메인메모리 1G를 갖는 시스템에서 실시하였다. 본 논문의 성능평가는 저장비용과 질의처리속도 측면으로 크게 두 개로 나누어 실시하였다. 제안하는 기법의 성능평가를 위하여 정적인 속성의 스카이라인 질의를 가장 효율적으로 처리하는 BBS기법을 도로망에 적합하도록 확장한 Extended-BBS을 비교대상으로 선정하였다. Extended-BBS기법에서 질의대상객체와 질의객체까지의 거리측정은 도로망의 에지확장을 통하여 계산하였다. 그러나 Extended-BBS기법은 모든 질의대상객체간의 거리측정은 재연산하지 않고 그림 3에서 설명한 것과 같은 부분적 필터링을 이용하였다.

Extended-BBS기법의 저장비용은 BBS기법에서 사용하는 R-Tree 파일과 도로망 연결정보를 유지하는 파일, 이 두 개의 파일용량을 측정하였다. 그리고 제안하는 기법의 경우에는 Extended-BBS기법에서 사용한 R-Tree 파일과 도로망 연결정보를 유지하는 파일뿐만 아니라 질의대상객체들의 최단영역 정보파일까지 포함한 파일용량을 측정하였다. 그림 9는 질의대상객체의 개수 증가에 따른 각각의 저장비용을 측정한 결과이다. 도로망을 구성하는 노드의 개수는 총 1,000개로 설정하고, 질의대상객체의 개수를 500개에서 5,000개 증가시키면서 저장비용을 측정한 것이다. 질의대상객체의 개수가 증가함에 따라 Extended-BBS기법과 제안하는 기법의 저장비용이 모두 증가하였다. 그러나 제안하는 기법은 미리 계산된 질의대상객체들의 최단영역을 저장한 파일이 객체정보를 에지단위로 중복적으로 저장하기 때문에 각각의 객체 정보만을 유지하는 Extended-BBS기법보다 상대적으로 저장비용이 더 크다. 또한 제안하는 기법은 객체의 개수가 증가함에 따라 저장비용의 증가율이 높다. 그러나 제안하는 기법의 저장비용이 실 응용서비스에 큰 영향을 미치는 정도는 아니다.

그림 10은 질의의 개수 증가에 따른 질의처리속도에 대한 결과를 보여준다. 총 노드의 개수는 1,000개로, 총 질의대상객체의 개수는 500개로 설정되었다. 그리고 총 질의의 개수를 5,000개에서 50,000개로 증가시키면서 질의처리속도를 측정하였다. 그 결과는 제안하는 기법이

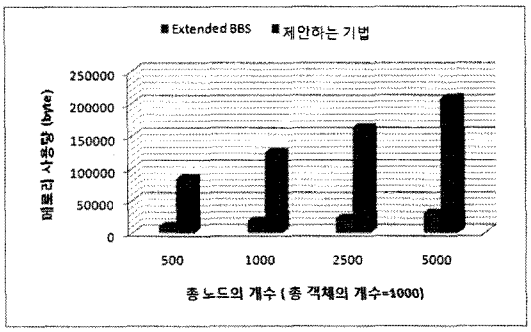

그림 9 객체의 개수 증가에 따른 저장비용

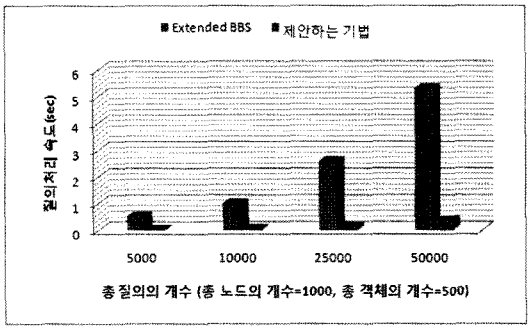

그림 10 질의의 개수 증가에 따른 질의처리속도

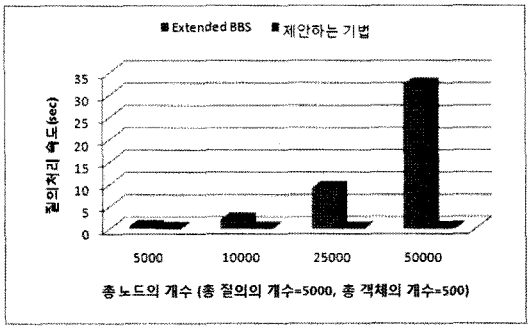

그림 11 노드의 개수에 증가에 따른 질의처리속도

Extended-BBS기법보다 약 20배에서 50배 정도 빠르게 처리한 것을 보여준다. 제안하는 기법은 최단영역정보를 참조하는 것으로 질의를 처리하기 때문에 매우 빠르다. 그러나 Extended-BBS기법은 늦은 질의처리속도뿐만 아니라 질의의 개수 증가에 따른 질의처리속도가 증가율이 매우 높다.

그림 11은 노드의 개수 증가에 따른 질의처리속도에 대한 결과를 보여준다. 총 질의의 개수는 5,000개로 설정하였고, 총 객체의 개수는 500개로 설정하였다. 총 노드의 개수는 5,000개에서 50,000개로 증가시키면서 처리속도를 측정하였다. Extended-BBS기법의 질의처리는

질의객체에 위치로부터 에지의 확장을 통하여 질의대상 객체와의 거리 값을 계산하여 처리한다. 그래서 제안하는 기법에 비해 매우 늦은 처리속도를 보여준다. 특히 노드의 개수 증가는 에지의 확장을 통한 질의처리에 큰 영향을 주기 때문에 Extended-BBS기법과 제안하는 기법의 질의처리속도는 약 50배에서 100배 이상의 차이가 난다.

5. 결 론

도로망에서 연속적 스카이라인 질의처리는 스카이라인을 요청한 질의객체가 도로망을 따라 이동하기 때문에 동적 데이터를 발생시킨다. 그래서 기존의 기법은 재연산을 통하여 지속적으로 변하는 동적 데이터를 처리한다. 그러나 이러한 재연산은 질의처리속도를 저하시킨다. 그래서 본 논문은 도로망에서 연속적 스카이라인 질의처리에 효율적인 기법을 제안하였다. 제안하는 기법은 질의대상객체의 속성들 중에 정적인 속성이 아닌 동적인 거리속성만으로 최종 스카이라인에 포함되는 객체의 최단영역을 질의처리에 사용하였다. 또한, 그러한 최단영역은 선처리되어 연속적인 스카이라인 질의처리에 사용하기 때문에 빠른 질의처리속도를 제공하였다. 성능평가를 통하여 제안하는 기법이 질의처리속도 측면에서 기존기법보다 약 100배 정도 빠른 것을 보여주었다.

참 고 문 헌

[1] S. Borzsonyi, D. Kossmann, and K. Stocker, "The skyline operator," ICDE, 2001, pp.421-430.
[2] D. Kossmann, et al. "Shooting stars in the sky: an online algorithm for skyline queries," VLDB, 2002, pp.275-286.
[3] D. Papadias, et al., "Progressive skyline computation in database systems," *TODS*, 30(1), 2005, pp.41-82.
[4] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting," ICDE, 2003.
[5] D. Papadis, Y. Tao, G. Fu, B. Seeger, "An Optimal and Progressive Algorithm for Skyline Queries," SIGMOD, 2003.
[6] K. L. Tan, P. K. Eng, and B. C. Ooi, "Efficient progressive skyline computation," VLDB, 2001.
[7] Huang, Z., Lu, H., Ooi, B. C., and Tung, A. K. H. "Continuous skyline queries for moving objects," TKDE 18, 2006, 1645-1658.
[8] W. Balke, et al., "Efficient distributed skylining for web information systems," EDBT, 2004, pp.256-273.
[9] K. Hose, "Processing Skyline Queries in P2P Systems," VLDB 2005 PhD Workshop, 2005, pp.36-40.
[10] Y. Tao and D. Papadias, "Maintaining sliding window skylines on data streams," TKDE, 18(3), 2006, pp.377-391.
[11] X. Lin, et al. "Stabbing the sky: efficient skyline computation over sliding windows," ICDE, 2005, pp.502-513.

장 수 민
정보과학회논문지 : 데이타베이스
제 36 권 제 1 호 참조

유 재 수
정보과학회논문지 : 데이타베이스
제 36 권 제 1 호 참조