DOI: 10.3745/KIPSTD.2010.17D.1.17

불완전 데이터를 위한 효율적 Top-k(g) 스카이라인 그룹 질의 처리 기법

박 미 라[†] · 민 준 기[††]

요 약

최근에 스카이라인 질의에 대한 관심이 점차 증가하고 있다. 대부분의 스카이라인 질의에 대한 연구는 데이터들이 널 값을 가지지 않는다는 가정에서 이루어진다. 그러나 우리가 웹이나 다른 도구로 데이터베이스에 자료를 입력할 때는 널 값을 가지는 불완전한 데이터가 존재한다. 따라서 불완전한 데이터를 위한 다양한 스카이라인 처리 기법들이 제안되었다. 그러나 기존의 불완전한 데이터를 위한 스카이라인 질의 처리 기법은 불완전한 데이터만을 고려함으로써 완전한 데이터와 불완전한 데이터가 공존하는 환경을 고려하지 않았다.

본 논문에서는 완전한 데이터를 위한 스카이라인 질의와 불완전한 데이터를 위한 스카이라인 질의를 모두 처리 하는 스카이라인 그룹 질의 처리 기법을 제안한다. 이를 위하여, 사용자 정의에 의한 차원의 선호도에 따라서 g개의 스카이라인 그룹을 검색하는 top-k(g) 스카이라인 그룹 질의를 도입하고, 이를 질의 처리하는 기법을 제안한다. 그리고 모의실험을 통하여 제안한 방식의 성능을 보인다.

키워드 : 스카이라인, 불완전한 데이터, 스카이라인 그룹 질의, Top-k(g)

An Efficient Processing Method of Top-k(g) Skyline Group Queries for Incomplete Data

Mi-Ra Park[†] · Jun-Ki Min[††]

ABSTRACT

Recently, there has been growing interest in skyline queries. Most of works for skyline queries assume that the data do not have null value. However, when we input data through the Web or with other different tools, there exist incomplete data with null values. As a result, several skyline processing techniques for incomplete data have been proposed. However, available skyline query techniques for incomplete data do not consider the environments that coexist complete data and incomplete data since these techniques deal with the incomplete data only.

In this paper, we propose a novel skyline group processing technique which evaluates skyline queries for the environments that coexist complete data and incomplete data. To do this, we introduce the top-k(g) skyline group query which searches g skyline groups with respect to the user's dimensional preference. In our experimental study, we show efficiency of our proposed technique.

Keywords : Skyline, Incomplete Data, Skyline Group Queries, Top-k(g)

1. 서 론

DBMS (DataBase Management System) 등과 같은 데이터 처리 시스템들이 처리하는 데이터가 방대해짐에 따라 사용자가 원하는 모든 조건에서 관심이 적은 데이터를 제외한 나머지 관심 있는 데이터를 검색하는 스카이라인 질의(skyline queries)의 필요성이 증대되고 있다. 스카이라인 질의는 다기준 의사결정론 (multi-criteria decision making), 데이터 마이닝 [2], 사용자 선호도 질의 [3]와 같은 많은 양의 데이터에서 원하는 자료를 찾는 응용들에서 매우 중요한 연산이다.

대부분 스카이라인 질의에 대한 연구는 데이터의 값이 완전하다는 가정에서 이루어진다 [4-6, 10-11]. 그러나 우리가 웹이나 기타 다른 도구로 데이터베이스에 자료를 입력할 때는 널 값을 가지는 부분이 존재한다. 스카이라인 질의에 대한 대부분의 이전 연구들은 완전한 데이터만을 고려하는 질의에 대한 처리 기법을 제안하고 있기 때문에 불완전한 데

※ 이 논문은 2009년 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2009-0072612).
† 준 회 원 : 한국기술교육대학교 정보미디어공학과 석사과정
†† 종신회원 : 한국기술교육대학교 인터넷미디어공학부 조교수
논문접수 : 2009년 8월 21일
수 정 일 : 1차 2009년 10월 14일, 2차 2009년 11월 11일
심사완료 : 2009년 11월 16일

이터를 고려해야 하는 데이터 공간에는 적용할 수 없다. 최근에는 이런 불완전한 데이터를 처리하기 위한 연구가 진행되었다 [7].

본 연구에서는 웹이나 멀티미디어 응용 같은 다양하고 방대한 양의 데이터를 다루는 시스템에서 검색 조건을 만족하는 모든 결과를 가져오기 보다는 상위 k개만 가져오고자 하는 top-k 질의 처리 [13]를 지원하고, 고려하는 조건이 많아지면 모든 데이터가 스카이라인이 되는 문제를 처리하는 기법 [5, 12]을 지원한다. 아래는 본 연구에서 연구하고자 하는 동기가 된 스카이라인 그룹 질의가 필요한 예를 소개한다.

[예 1] 온라인 이력서 자동 심사

어떤 큰 회사에서 신입사원을 채용할 때 인터넷 전형을 사용하는 예를 고려해보자. 수천명의 지원자의 이력서가 데이터베이스로 전송되고, 회사의 인사 담당자가 모든 사람의 이력서를 검토하는 것은 불가능하다. 이런 경우에 회사에서 원하는 사원을 적절하게 뽑기 위해 스카이라인 질의가 필요하다. 회사에 이력서를 쓰는 사람들은 서류전형을 통과하기에 불리한 이력은 기입하지 않는다. 지원자가 기입하지 않은 공란은 데이터베이스에서 빈 공간으로 기록하게 되고, 공란 없이 기입한 사람과는 달리 데이터베이스에 널 값으로 남게 된다.

회사마다 사원을 채용할 때 우선순위로 생각하는 기준은 다르다. 예를 들어 어떤 회사는 영어를 잘하는 사람을 원하고, 어떤 회사는 학점이 높고 수상 경력이 많은 사람을 원할지도 모른다. 학점이 높고 수상 경력이 많은 사람을 원하는 회사가 토익 점수도 추가로 고려할 때를 생각해보자. 토익점수를 기록하지 않은 사람이 0점이라고 볼 수는 없을 것이다. 점수를 기재하지 않았더라도 토익점수를 기재한 사람의 점수가 기준보다 낮다면 토익점수를 기록한 다른 지원자보다 영어를 잘할지도 모른다.

[예 2] 원하는 상품 검색

온라인으로 고객이 상품을 고를 때, 고객은 물건의 가격이 싸고, 품질이 좋고, 수리서비스가 좋고, 제조사의 인지도 또한 좋은 상품을 구매하기를 원한다. B라는 상품은 A라는 상품보다 가격이 비싸고, 품질이 나쁘고, 수리서비스도 안좋고, 제조사의 인지도 또한 떨어진다. 따라서, 고객에게 B라는 상품은 보여줄 필요조차 없다. 이런 경우에 스카이라인 질의가 필요하다.

그런데, C라는 상품은 A라는 상품보다 가격이 싸고 품질도 좋고, 제조사의 인지도도 좋지만 수리서비스가 잘되는지의 여부는 알 수 없다. C라는 상품의 수리서비스 정보를 모르기 때문에 C라는 상품은 A라는 상품보다 모든 면에서 우위에 있지는 않지만 수리서비스 정보에 상관없이 고객에게는 더 좋은 상품이 될지도 모른다.

위의 두 예를 해결하기 위해 본 논문에서는 고려하는 모든 조건에 값이 있는 완벽한 데이터의 스카이라인과 함께 널 값이 포함된 불완전한 데이터의 스카이라인 질의를 함께 처리하고자 한다.

본 논문에서는 완전한 데이터와 동등하거나 혹은 더 좋을지도 모르는 데이터를 함께 검색해주는 스카이라인 그룹 질의를 도입하고 스카이라인 그룹 질의를 처리하는 방법을 제안한다. 또한, 불완전한 데이터를 그룹 지을 때 우선순위를 고려하는 효율적인 방법을 제안한다. 제안한 기법은 사용자의 선호도 우선순위가 높은 상위 k(g)개의 검색 결과를 가져온다.

2. 관련 연구

2.1 스카이라인

이 절에서는 스카이라인에 대한 정의를 제시한다. 이를 위하여 우선, <표 1>에 스카이라인의 정의에 사용된 기호들 및 본 논문의 설명을 위해 필요한 기호들 정의하였다.

〈표 1〉 사용되는 표기법들

표기	정의
C	완전 데이터들의 집합
I	불완전 데이터들의 집합
D	모든 차원의 집합
c	완전 데이터 집합의 원소(즉, $c \in C$)
i	불완전 데이터 집합의 원소 (즉, $i \in I$)
d_j	하나의 차원 (즉, $d_j \in D$, where $1 \le j \le n$)
$t(d_j)$	t의 d_j 차원 값, where $t \in I$ 또는 $t \in C$
$R(t_j)$	t_j의 타당한 차원정보, 즉 $\{d_j \mid t(d_j) \neq \text{null}\}$

<표 1>에서 $R(t_j)$는 투플 t_j의 차원들 중에서 t_j의 값이 정의된 차원들만을 반환하는 함수이다. 예를 들어 (1, 5, 4)에 대하여서는 $\{d_1, d_2, d_3\}$가 반환되고 (1, -, 4)에 대하여서 $\{d_1, d_3\}$가 반환된다.

스카이라인은 기존 연구들[1, 4, 5, 8, 9]에서 다음과 같이 정의된다. 우선 스카이라인을 정의하기 위하여 지배(dominate)를 정의한다.

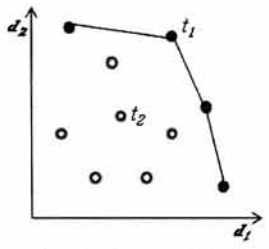
[정의 1] (지배). 만약 $\forall d_k \in D$, $t_a(d_k) \ge t_b(d_k)$ 그리고 $\exists d_j \in D$, $t_a(d_j) > t_b(d_j)$ 이면, 투플 t_a는 투플 t_b를 지배(dominate)한다[1].

예를 들어, t_1이 (1, 5, 4) 이고 t_2가 (2, 5, 4)일 경우, 정의 1에 따라서 t_2가 t_1을 지배한다고 한다. 정의 1의 지배를 이용하여 다음과 같이 스카이라인이 정의된다.

[정의 2] (스카이라인). 만약 D에서 $\forall t_b (\neq t_a)$ 가 t_a를 지배하지 않으면, 투플 t_a는 스카이라인 객체이다.

(그림 1)는 스카이라인 정의에 따라서 표현한 스카이라인

1) 여기서는 최대 스카이라인 연산자 [1]를 기반으로 정의한다.



(그림 1) 스카이라인

을 찾는 그림이다. d_1과 d_2의 차원을 고려하여 큰 값을 선호하는 스카이라인을 구할 때, t_1과 t_2를 비교하면 t_1은 d_1과 d_2의 차원에서 모두 t_2보다 우위에 있다. 그리고 t_1을 지배하는 다른 객체는 존재하지 않으므로 t_1은 스카이라인이 된다. 이런 방식으로 객체들의 스카이라인을 구하면 색이 칠해진 원들이 스카이라인 객체가 된다.

2.2 불완전한 데이터를 위한 스카이라인 질의 처리 기법

Khalefa [7] 등은 불완전한 데이터를 위한 스카이라인 질의 처리 알고리즘인 ISkyline을 제안하였다. 우선 [7]에서는 버킷 알고리즘을 제시하였다.

버킷 알고리즘은 모든 들어오는 데이터를 같은 비트 표현을 가지는 중복되지 않는 버킷들로 나눈다. 그리고 버킷별로 스카이라인을 구한다. 각 버킷을 위한 스카이라인들의 집합을 로컬 스카이라인이라고 부른다. 모든 로컬 스카이라인 집합들로부터 최종의 스카이라인이 될 수 있는 후보 객체들을 모으고 그것을 하나의 리스트로 만든다. 이를 후보 스카이라인이라고 부른다. 그러나 후보 스카이라인 리스트의 크기는 모든 버킷들에서 모든 지역 스카이라인들을 병합함으로 매우 클 수 있다.

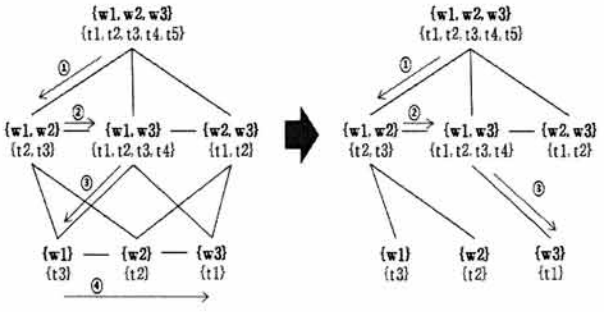
이를 해결하기 위하여, [7]에서는 Virtual Points와 Shadow Skylines라는 개념을 이용하여 버킷(Bucket) 알고리즘을 활용한 ISkyline 알고리즘을 제안하였다.

ISkyline 알고리즘에서는 로컬 스카이라인의 수를 줄임으로써 후보 스카이라인 리스트의 크기를 감소시키도록 하기 위하여 Virtual Points 기법을 제시하였다. Virtual Point란 현재까지 만들어진 후보 스카이라인 리스트를 이용하여 만들어지는 가상의 스카이라인으로 이를 검색 대상인 버킷에 삽입하여 해당 버킷의 로컬 스카이라인 수를 감소시킨다. 예를 들어 현재 후보 스카이라인에 (1, 3, -, 4) 값을 가지는 객체 P1이 존재하고 검색 대상 버킷에 Q1 (-, 1, 7, 2) 객체가 존재할 경우 P1를 이용하여 (-, 3, -, 4)를 Virtual Point로 만들고 이를 검색대상 버킷에 삽입한다. 이 Virtual Point로부터 2차원 및 4차원에서 3은 1보다 크고 4는 2보다 큼으로 Q1은 스카이라인이 될 수 없음을 알 수 있음으로 로컬 스카이라인이 되지 못한다.

추후, 후보 스카이라인 리스트 상에 존재하는 모든 객체들의 모든 가능한 쌍을 만들고 각 쌍들 간의 비교를 통하여 최종 스카이라인들을 추출하여야 한다. 이 경우 많은 처리 비용이 발생한다. 이러한 처리비용을 감소시키기 위하여 ISkyline에서는 shadow skyline 개념을 제안하였다. shadow skyline이란 한 버킷의 실제 스카이라인들 중에서 로컬 스카이라인이 아닌 것을 말한다. 이 shadow skyline들과 후보 스카이라인들과의 비교를 통하여 최종 스카이라인들을 추출한다 (자세한 사항은 [7] 참조).

2.3 우선순위를 고려한 top-k 질의 처리 기법

모든 차원에서 지배되지 않는 자료들을 스카이라인으로 선정하다 보면, 차원이 많아질수록 모든 자료가 스카이라인이 되는 '차원의 저주' 문제가 발생하게 된다. 이 문제점을 해결하기 위해 차원에 우선순위를 두고 우선순위가 높은 k개만을 검색하고자 제안된 알고리즘이 Telescope [5]이다.



(a) 일반적인 방법 (b)Telescope방법

(그림 2) lattice 그래프와 left-skewed 그래프.

(그림 2) (a)은 차원의 우선순위가 존재할 때의 top-k개의 스카이라인을 구하는 일반적인 처리 방법을 나타낸다. (그림 2)에서 w1, w2, w3는 각 차원을 나타내며, 차원의 선호도는 w1>w2>w3이다. 이 경우, (그림 2) (a)와 같이 lattice 형태의 그래프를 생성할 수 있으며 top-3의 스카이라인들을 구할 때, 총 4번의 노드 방문이 필요하다.

이에 반하여 Telescope알고리즘에서는 (그림 2) (b)와 같이 lattice 그래프를 left-skewed 그래프로 변환하고 left-skewed 그래프를 탐사하며 top-k 스카이라인을 추출한다. 우선, 최상위 노드부터 검색하여 스카이라인 객체와 결과 집합에 들어있는 스카이라인 객체의 합집합이 k개보다 작으면 현재 노드의 스카이라인을 결과 집합에 넣은 후 그래프가 이어진 형제 노드로 이동하고, k개보다 크면 그래프에 자식 노드로 이동한다. (그림 2(b))의 경우, 루트 노드에는 총 5개의 스카이라인이 있음으로 첫 번째 자식인 {w1, w2}로 이동하고, {w1, w2}에는 2개의 스카이라인이 있음으로 결과 스카이라인으로 포함시키고 이웃한 형제 노드인 {w1, w3}으로 이동한다. 이러한 방식으로 left-skewed 그래프를 탐색하면 총 3번의 노드 방문만으로 top-3 스카이라인들을 구할 수 있다.

3. 스카이라인 그룹 질의

3.1 스카이라인 그룹

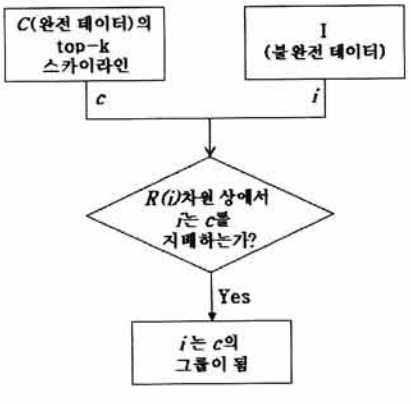
본 연구에서 제안하는 스카이라인 그룹 질의 처리 기법은 기존의 불완전 데이터의 스카이라인 질의 처리와는 달리 완

전한 데이터와 불완전한 데이터가 공존하는 상황을 고려하여 스카이라인 질의를 수행한다. 스카이라인 그룹 질의는 완전한 데이터의 스카이라인 객체에 대하여 추가적으로 불완전한 데이터들로부터 스카이라인 그룹을 검색한다. 여기서 스카이라인 그룹이란 불완전한 데이터들 중에서 하나의 스카이라인 객체 c를 지배하면서 값이 정의된 차원만을 고려 할 경우에는 스카이라인이 되는 객체들을 말한다. 완전한 데이터 집합의 한 스카이라인 c에 대한 스카이라인 그룹 $SG(c)$에 속한 원소들에 대한 정의는 다음과 같다.

[정의 3] (스카이라인 그룹). $i_a \in I$ 는 집합 $K=\{ i_j \mid R(i_j) = R(i_a)$ and $i_j \in I\}$ 에 대하여 $R(i_a)$ 차원 상에서 스카이라인이며 $R(i_a)$차원 상에서 완전한 데이터의 스카이라인 c를 지배한다면 i_a는 $SG(c)$의 한 원소이다.

예를 들어, i_a 가 (5, -, 7)이라고 할 때 $R(i_a) =\{d_1, d_3\}$이고 $R(i_a)$와 동일 차원 정보를 가지는 데이터가 (1, -, 3) 밖에 없다고 할 때, i_a는 $R(i_a)$차원 상에서 스카이라인이 된다. 또한 여기서, 완전한 데이터의 한 스카이라인 c가 (1, 4, 7)라고 할 때, $R(i_a)$차원에서 i_a는 c를 지배하므로 i_a는 $SG(c)$에 속한다.

스카이라인 그룹을 정의 3과 같이 정의한 이유는 $R(i_a)$차원 상에서 완전한 데이터 c를 지배하는 불완전한 데이터 i_a는 $R(c)$차원으로 확장하였을 때 완전한 데이터 c에 의해 지배받지 않는, 정의 2에 따라서 스카이라인이 될 가능성이 있는 객체이기 때문이다. 즉, 불완전한 데이터의 특정 차원 값이 없는 부분을 0으로 대체하여 완전한 객체로 만든 후, 완전한 데이터의 스카이라인과 비교하면 그 불완전한 데이터의 객체는 완전한 데이터의 스카이라인 객체에 지배되지 않는다. 예를 들어, 완전한 데이터의 한 스카이라인 c가 (1, 4, 7)이고, 불완전한 데이터 i_a가 (5, -, 7)이라고 할 때, i_a를 (5, 0, 7)로 만든 후 c(1, 4, 7)와 비교하면 i_a(5, 0, 7)는 c(1, 4, 7)에 지배되지 않는다. 그러므로 우리는 완전한 데이터의 스카이라인과 동등하거나 더 좋을지도 모르는 객체를 함께 보여주기 위해 스카이라인 그룹이라는 개념을 사용하여 스카이라인 질의를 한다.



(그림 3) 스카이라인 그룹을 구하는 순서

본 논문에서는 스카이라인 그룹에 속한 원소들의 개수가 무한히 커지는 것을 방지하기 위하여 한 개의 스카이라인 객체 당 g개로 스카이라인 그룹의 원소 수를 제한한다. 그리고 top-k 스카이라인의 한 스카이라인 객체 당 g개의 스카이라인 그룹을 구하는 것을 top-k(g)로 표기한다.

(그림 3)은 C의 top-k스카이라인 객체 한 개에 대한 그룹을 구하는 순서도를 보여준다. top-k(g)의 원소를 구하는 과정은 다음과 같다. 스카이라인 그룹을 구하기 위해서는 C의 top-k 스카이라인 집합과 I를 비교한다. 즉 완전한 데이터의 스카이라인 c와 I의 원소 i를 비교하여 $R(i)$차원 상에서 i가 c를 지배하면 i는 c의 그룹이 된다. I의 다음 원소를 가져와서 c와 비교한다. c의 그룹이 g개가 될 때까지 반복한다.

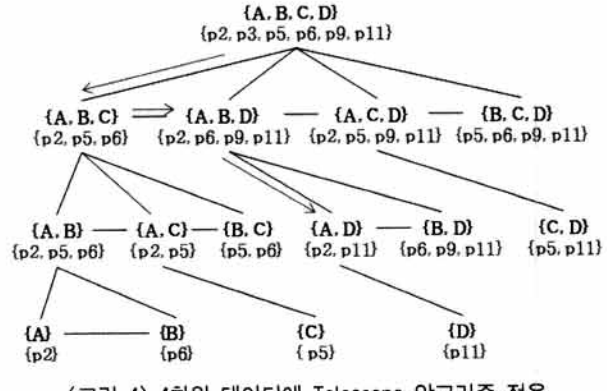
3.2 top-k(g) 스카이라인 그룹 질의

불완전한 데이터가 포함된 데이터베이스에서 완전한 데이터에 일반적인 스카이라인 질의를 하고, 추가로 3.1에서 정의한 스카이라인 그룹 개념을 사용하여 불완전 데이터의 스카이라인 그룹 질의를 수행한다. 본 연구에서는 스카이라인 질의를 수행할 때 생기는 '차원의 저주' 문제 등을 해결하기 위해 완전한 데이터의 스카이라인 질의를 수행할 때는 관련 연구 2.2의 Telescope 알고리즘 [5]을 활용한다.

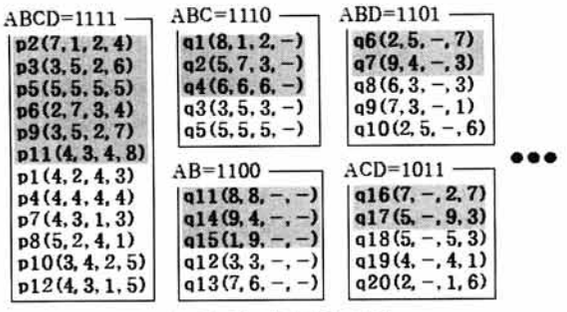
Telescope 알고리즘을 적용하면, 사용자가 선호하는 차원을 고려하여 top-k개의 스카이라인 객체를 검색해준다. 예를 들어, (그림 5)에 나타난 4차원의 데이터를 고려하여 (그림 4)와 같이 left-skewed 그래프로 나타낼 수 있다. 왼쪽 자식 노드에서 검색한 스카이라인은 자손들의 스카이라인 객체들을 모두 포함하므로 하나의 부모 노드에 연결된 자식 노드는 형제 노드의 자식 노드와는 연결하지 않는다.

완전한 데이터의 스카이라인은 p2, p3, p5, p6, p9, p11이다. 그래프에서 부모 노드의 자식 노드는 부모 노드의 부분집합 중 하나가 된다. 부모 노드에 있는 객체들 중 위쪽에 {A, B, C, D}로 표시한 차원을 고려하여 스카이라인을 구하여 그래프를 완성한다. (그림 4)는 A>B> C>D 차원의 순서로 사용자가 선호한다고 가정하고 만든 그래프이다.

화살표는 우선순위를 고려하여 top-4개를 검색하고자 할 때 탐색되는 과정을 나타낸다. (즉 top-k에서 k를 4로 설



(그림 4) 4차원 데이터에 Telescope 알고리즘 적용



(그림 5) 데이터의 버킷

정). 최상위 노드의 스카이라인 집합의 개수가 4개보다 크므로 그래프가 연결된 다음 노드로 이동한다. {A, B, C} 차원을 고려한 스카이라인 집합은 {p2, p5, p6} 이다. 이는 4개보다 작으므로 결과 집합에 넣은 후 다음 노드로 이동한다. {A, B, D} 차원의 스카이라인 집합인 {p2, p6, p9, p11}과 결과 집합을 합집합하면 {p2, p5, p6, p9, p11}이 되어 4보다 크므로 자식 노드로 이동한다. {A, D} 차원의 스카이라인 집합인 {p2, p11}을 결과 집합과 합집합하면 {p2, p5, p6, p11}이 되어 검색하고자 했던 개수(즉, k=4)와 같아진다.

이런 방식으로 완전한 데이터의 top-k 스카이라인을 구한 후, k개의 각 스카이라인 객체 당 스카이라인 그룹을 검색한다. 여기서 구하는 스카이라인 그룹은 완전한 데이터와 동등하거나 더 좋을지도 모르는 데이터를 그룹으로 만들게 된다. 즉, 3.1의 스카이라인 그룹 정의에 따라 스카이라인 객체를 지배하는 불완전한 데이터를 구한다.

우리는 불완전 데이터로부터 스카이라인 그룹을 추출할 때 사용자가 선호하는 차원의 우선순위를 고려하고, 많은 비교 횟수를 피하기 위해 [7]에서 제안한 버킷 개념을 적용한다.

데이터의 값이 있는 부분을 1, 없는 부분을 0으로 나타내는 것과 같이 데이터의 차원의 값을 1과 0의 비트로 표현할 때, 차원의 비트 표현이 같은 데이터를 같은 버킷으로 만든다. 예를 들어, q1(8, 1, 2, -), q2(5, 7, 3, -) 두 객체가 있다면 이것의 비트 표현은 1110이 되고, 두 객체는 비트 표현이 1110인 데이터들과 같은 버킷이 된다.

(그림 5)는 데이터의 버킷들을 표현한 것이고, 색칠된 부분은 각 버킷에서의 스카이라인 객체이다. 버킷의 개수는 최대 2^n-1개(n은 차원의 수)가 된다.

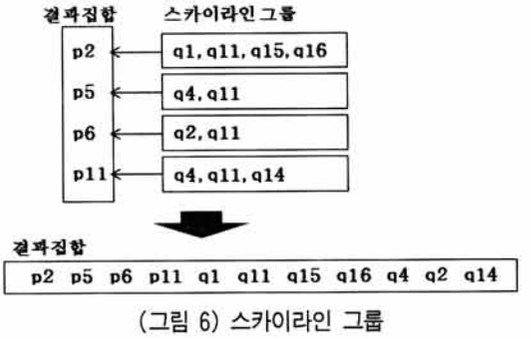
스카이라인 그룹을 만드는 과정은 하나의 완전한 데이터의 스카이라인 객체 당 g개의 불완전한 데이터를 검색한다. 이러한 불완전한 데이터의 스카이라인 그룹을 추출하는데 있어서 매번 하나의 데이터 버킷으로부터 불완전한 데이터 객체를 추출하는 것은 많은 비용을 발생시킨다.

이러한 문제점을 해결하기 위하여 본 연구에서는 차원의 비트 표현이 같은 데이터를 묶어 버킷으로 나누고, 버킷별로 스카이라인을 미리 구하여 스카이라인 그룹 질의에 사용할 수 있도록 유지한다. 이러한 이유는 하나의 버킷 내에서 스카이라인이 아닌 객체는 스카이라인 객체에 지배를 받으므로 정의 3에 의하여 절대로 스카이라인 그룹의 결과에 포함될 수 없기 때문이다. 다시 말하면, 어떤 불완전 객체 i_a가 스카이라인 그룹의 결과에 속할 경우, 이를 지배하는 불완전 객체 i_b가 있을 경우 스카이라인 그룹에서 i_a를 제거하고 i_b를 추가하는 것이 올바른 답이기 때문이다.

즉, 비교 연산을 수행할 때는 미리 구해놓은 각 버킷들의 스카이라인과 완전한 데이터의 스카이라인 객체와의 비교만 수행한다. 각 버킷에서의 스카이라인이 아닌 객체의 경우 어떤 한 객체의 그룹이 될 가능성이 있는 객체라 하더라도 이미 스카이라인 그룹이 된 버킷들의 스카이라인에 의해 지배되기 때문에 미리 연산을 수행하여 최종 연산 시간을 줄인다.

예를 들어, (그림 5)의 예에서 완전한 데이터의 스카이라인인 p5(5, 5, 5, 5)의 경우 q13(7, 6, -, -)과 비교했을 때 q13은 p5의 스카이라인 그룹이 될 수 있는 조건을 가졌지만 이미 스카이라인 그룹이 된 p11(8, 8, -, -)에 의해 지배되므로 결국 스카이라인 그룹에서 제외된다.

사용자의 선호하는 차원을 고려하기 위해, {A, B, C, D}라는 4개의 차원에서 스카이라인 그룹을 구한다면 'ABC->ABD->AB->ACD->AC->AD->A->BCD->BC->BD->B->CD ->C->D'와 같은 버킷 순서로 선호하는 차원순서와 더 많은 정보가 채워진 순서로 그룹을 찾는다.



(그림 6) 스카이라인 그룹

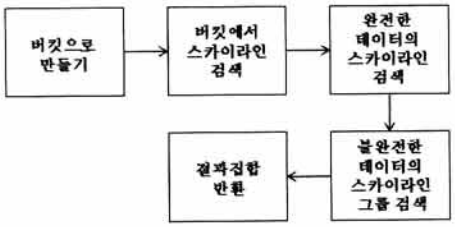
(그림 6)은 (그림 4)에서 구한 완전한 데이터의 스카이라인 객체에 대하여 그룹의 크기(즉, g)를 4로 설정한 경우의 스카이라인 그룹을 구하는 예를 소개한다. 결과 집합의 객체를 한 개씩 가져와서 (그림 5)의 각 버킷별로 구한 스카이라인과 비교한다.

p2의 경우, 비트연산자 1110을 가지는 버킷의 스카이라인과 처음으로 비교된다. p2의 ABC차원인 (7, 1, 2)와 ABC 버킷의 첫 번째 스카이라인 객체인 q1(8, 1, 2)을 비교해보면 q1이 p2를 지배한다. 그러면 q1은 p2의 스카이라인 그룹이 된다. 다음으로 p2(7, 1, 2)는 ABC 버킷의 두 번째 스카이라인 객체인 q2(5, 7, 3)와 비교된다. q2는 p2를 지배하지 못하므로 다음 스카이라인 객체인 q4로 넘어간다. q4와 비교한 후에는 1110다음으로 비트표현이 큰 1101표현을 가지는 ABD버킷의 스카이라인 객체와 비교를 수행한다. 이런 식으로 선호하는 차원 순서대로 스카이라인을 비교해보면 (그림 6)과 같이 스카이라인 그룹을 만들 수 있다.

검색이 끝나면 결과 집합에 포함된 객체들에 스카이라인 그룹으로 지정된 불완전한 데이터를 가져오고 중복이 되는

지 여부를 확인한 후, 결과 집합에 포함시킨다.

(그림 7)은 본 논문에서 제안하는 스카이라인 그룹 질의를 처리하는 방법을 흐름도로 나타낸다. 데이터들을 값이 채워진 비트 표현이 같은 것들끼리 버킷으로 만들고 각 버킷에서 스카이라인을 구한다. 그 다음 Telescope 알고리즘[5]을 활용하여 완전한 데이터들의 우선순위를 고려해 k개의 스카이라인 객체를 구하고, 그 스카이라인 객체에 대응하는 각각의 스카이라인 그룹을 구한다. 중복된 스카이라인 그룹을 제외하고 완전한 데이터와 스카이라인 그룹으로 지정된 객체를 반환한다.



(그림 7) 전체 흐름도

3.3 스카이라인 그룹 질의 처리 알고리즘

스카이라인 그룹 질의 처리 알고리즘은 완전한 데이터의 top-k 스카이라인 객체 당 스카이라인 그룹을 구하는 과정을 수행한다. 비교 연산을 수행하면서 스카이라인 객체의 스카이라인 그룹이 g개가 되거나 더 이상의 스카이라인이 없으면 탐색을 중단한다.

(그림 8)은 스카이라인 그룹을 구하는 함수를 보여준다.

SkyGroup() 알고리즘의 입력 데이터는: (1)완전한 데이터의 top-k 스카이라인 객체 집합 S, (2)사용자가 선호하는 순서로 된 고려하는 차원 정보 D, (3)스카이라인 그룹의 크기 g이다.

본 알고리즘에서 GS는 완전한 데이터의 스카이라인 객체 s에 대한 스카이라인 그룹 집합을 나타낸다 (라인 (1)). 이 알고리즘에서 사용하는 버킷의 우선순위는 집합에서 멱집합의 원소를 열거해놓은 것을 이용한다. 차원 정보 집합의 모든 부분집합을 원소로 하는 멱집합을 구하여 스택 U에 저장한 뒤, 마지막에 저장된 멱집합의 원소부터 가져오면 본 알고리즘에서 사용하는 버킷의 우선순위와 일치한다. 이 알고리즘에서는 멱집합을 구하는 함수를 powerSetFunc()로 표현하였다(라인 (2)(3)). top-k 스카이라인에 대응되는 스카이라인 그룹을 구하여 중복을 제거하며 Z에 담는다(라인 (4)-(7)).

Algorithm1 SkyGroup(S, D, g)
Input
S : top-k skyline points of complete data ($S \subseteq C$)
D : {d_1, d_2, ... d_n} // *user-specific qualitative preference*
g : group retrieval size
Output
Z : skyline group
Procedure\
1: $GS \leftarrow \{\}$ // *skyline group in bucket*
2: U is a stack
3: $U \leftarrow$ powerSetFunc(D) //*bucket priority of D*
4: **for** each point $s \in S$ **do**
5: GS = top_g (s, U, g) //*skyline points per bucket*
6: $Z = Z \cup GS$ //*remove duplication*
7: **end for**
8: **return** Z

(그림 8) 스카이라인 그룹 질의 알고리즘

하나의 완전한 데이터의 스카이라인 객체로부터 g 크기의 스카이라인 그룹을 구하는 알고리즘은 (그림 9)에 나타나 있다.

(그림 9)는 하나의 스카이라인 객체에 대한 스카이라인 그룹을 계산하는 의사코드(pseudo code)를 나타낸다. top-g() 알고리즘의 입력 데이터는: (1)완전한 데이터의 스카이라인 객체 s (2)사용자 선호도 우선순위를 고려한 버킷들의 우선순위 U, (3)스카이라인 그룹의 크기 g 이다.

일단, 더 이상 고려할 버킷이 없으면 알고리즘은 종료된다(라인 (2)(3)). (라인 (4))에서 F는 멱집합의 원소를 가져온 버킷의 정보를 저장한다. 이를 위하여 스택 U에 들어 있는 사용자 선호도 우선순위에 따른 버킷 정보를 pop()연산을 이용하여 순서대로 추출한다. 이후 해당 버킷의 스카이라인 집합을 BS에 저장하고(라인 (5)) 스카이라인 그룹 질의를 하는 탐색을 수행한다. 버킷에서 구해놓은 스카이라인 i와 완전한 데이터의 스카이라인 객체인 s를 비교한다. i의 값이 있는 차원들에 한해서 비교한 후, i가 s를 지배하면 i는 s의 그룹이 된다 (라인 (6)-(9)). 이후 GS의 원소 수가 g개가 되지 않으면 GS가 g개가 될 때까지 재귀 호출을 이용하여 반복한다(라인 (10)-(11)). 여기서, 다음번 재귀호출에서는 F가 스택 U에서 pop()됨으로 다음 순위의 버킷을 고려 대상으로 삼게 된다. 최종적으로 top_g() 함수를 호출한 SkyGroup() 함수로 현재 스카이라인 객체의 g개의 그룹을 담은 GS를 리턴 해준다.

Algorithm2 top_g (s, U, g)
Input
s : a skyline point of complete data ($s \in S$)
U : *bucket priority of D*
g : group retrieval size
Output
GS : skyline group with respect to g
Procedure
1: F, BS //*valuable, Object* ($BS \subseteq I$)
2: **if** U is empty **then**
3: return;
4: $F \leftarrow U$.pop() //*bucket bit operator*
5: $BS \leftarrow$ skyline set in bucket of F dimension
6: **for** each point $i \in BS$ **do**
7: **if** i dominates s **then** //*comparable in only F*
8: insert i into GS
9: **end for**
10: **if** $|GS| < g$ **then**
11: $GS = GS \cup$ top_g(s, U, g)
12: **else**
13: return GS

(그림 9) 불완전 스카이라인 그룹 알고리즘

4. 실험 및 성능평가

4.1 실험환경

본 논문에서 제안한 스카이라인 그룹 질의를 처리하기 위한 스카이라인 그룹 알고리즘의 성능을 평가하기 위해 대상 객체의 모의 데이터를 생성하여 실험하였다. 본 논문에서 제안하는 질의 처리는 완전한 데이터의 스카이라인을 구한 후, 추가적으로 불완전한 데이터의 스카이라인 그룹을 구하는 것이다. 완전한 데이터의 스카이라인을 구하는 방법은 기존에 연구된 Telescope알고리즘 적용하였고, 불완전한 데이터의 스카이라인을 구하는 연산을 평가한다.

비교 실험을 통한 성능 평가를 위해 모든 기법은 동일하게 Intel(R) Core(TM)2 Duo CPU E6550 2.33GHz 프로세서와 2GB의 메인 메모리, 300GB의 하드 디스크를 가진 Windows XP 운영체제의 PC에서 자바언어(JSDK 1.6)와 데이터베이스(PostgreSQL 3.1)를 이용한 환경에서 구현하고 실험을 수행하였다.

본 논문에서는 불완전한 데이터를 고려한 새로운 스카이라인 질의인 top-k(g) 스카이라인 그룹 질의 처리 기법을 제안하였으므로 비교 대상 기법이 존재하지 않는다. 따라서 본 실험에서 제안한 기법에서 불완전 데이터를 위한 버킷 당 스카이라인을 미리 구해놓는 방법(bucket)과 스카이라인을 미리 구해 놓지 않는 방법(non-bucket)을 비교 분석하였다.

본 실험에서 사용한 데이터는 균등 분포를 이용하여 생성하였다. 본 실험에서 사용한 변수들의 정보는 <표 2> 에 나타나 있다. 본 실험에서는 <표 2> 에 나타난 변수들의 값을 변화시켜 각 상황에 따른 성능을 측정하였다.

<표 2> 실험 환경 변수

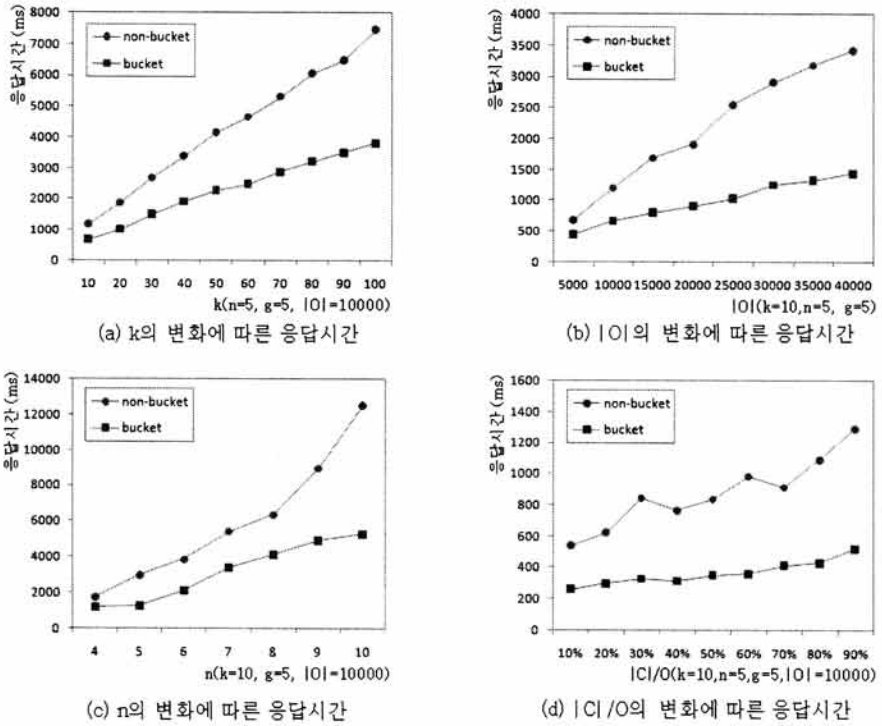
기호	기본값	설명
n	5	차원의 수
k	10	완전한 데이터의 스카이라인 수
g	5	스카이라인 그룹의 원소 수
$\|O\|$	10,000	객체 수 (= $\|C\|+\|I\|$)
$\|C\|/O$	30%	전체 객체 중 완전한 데이터의 비율

4.2 성능평가

완전한 데이터의 스카이라인을 구하는 성능은 기존에 연구된 Telescope알고리즘과 동일하다. (그림 10)은 Telescope 알고리즘의 처리시간과 본 논문에서 제안한 스카이라인 그룹 구하는데 필요한 시간을 합한 성능을 나타낸다.

(그림 10)에서 x축은 변화하는 고려조건 나타내고, y축은 최종 응답시간을 나타낸다. 그래프에서 사용한 기호는 non-bucket이라고 표기한 것은 버킷별로 스카이라인을 미리 구해 놓지 않은 방법이고, bucket이라고 표기된 것은 본 논문에서 제안하는 버킷별로 스카이라인을 미리 구해놓은 시스템에서 완전한 데이터의 스카이라인 객체 당 스카이라인 그룹을 구하는 스카이라인 질의 처리 기법이다.

(그림 10) (a)는 차원의 수를 5로, 스카이라인 그룹의 크기를 5로, 자료의 수를 10000개로 고정한 상태에서 완전한 데이터의 스카이라인 수 k를 변환시키며 실험한 결과이다. 버킷에서 스카이라인을 미리 구해 놓지 않은 자료에 비해 낮은 폭으로 응답속도가 증가하는 것을 볼 수 있다. 이는 k가 증가함에 따라서 스카이라인 그룹의 개수도 증가하게 됨으로 불완전한 데이터의 각 버킷별로 미리 스카이라인들을 구해 놓은 것이 매우 효과적임을 보여준다.



(a) k의 변화에 따른 응답시간 (b) |O|의 변화에 따른 응답시간

(c) n의 변화에 따른 응답시간 (d) |C| /O의 변화에 따른 응답시간

(그림 10) 환경의 변화에 따른 응답시간

(그림 10) (b)는 객체의 수 O를 변화시킨 실험 결과이다. 앞의 실험과 마찬가지로 버킷에서 스카이라인을 미리 구해 놓지 않은 non-bucket에 비해, 객체의 수가 증가함에 bucket방법은 처리 비용이 낮은 폭으로 증가하는 것을 볼 수 있다. non-bucket 방법의 기울기(즉, 객체 수 증가에 대한 처리 비용 증가율)은 0.078이고 bucket 방법의 기울기는 0.029이다. 따라서 bucket기법이 non-bucket기법 보다 2.69배 더 좋다.

(그림 10) (c)는 고려하는 차원의 수를 늘려가면서 실험한 결과이다. 버킷별로 스카이라인을 미리 구해 놓지 않은 시스템에 비해 낮은 폭으로 응답속도가 증가한다. 그림 10(c)에서 보는 바와 같이 non-bucket기법은 10차원 데이터의 경우 (n =10), 처리 비용이 급증하는데 비하여 bucket기법은 서서히 증가함을 볼 수 있다.

불완전한 데이터의 비율이 응답속도에 큰 영향을 미치는가를 실험한 결과인 (그림 10) (d)에서는 불완전한 데이터의 비율은 응답속도에 영향을 주지 않음을 알 수 있었다.

5. 결 론

본 연구에서는 완전한 데이터의 스카이라인에 추가로 불완전한 데이터를 그룹 질의하는 스카이라인 그룹 질의 및 해당 질의의 처리 기법을 제안하였다. 기존의 연구는 불완전한 데이터를 중점으로 처리하는 방법을 지원한다. 이와는 달리, 스카이라인 그룹 질의 처리 기법은 완전한 데이터의 스카이라인을 먼저 탐색한 후, 이와 동등하거나 더 좋을지도 모르는 데이터를 그룹 질의 처리한다.

스카이라인 그룹 질의 처리 기법에서는 스카이라인 그룹이라는 새롭게 정의한 개념을 사용한다. 각 스카이라인 객체별로 선호도가 높은 차원의 값을 가지고 있는 불완전한 데이터에서 스카이라인 그룹을 탐색하여 완전한 데이터와 함께 검색 결과에 포함시킨다. 사용자의 선호도를 고려하기 위해 값이 있는 차원을 비트 연산으로 표현하여 버킷으로 만든 후, 버킷의 우선순위에 따라 스카이라인 그룹을 탐색하였다. 그리고 버킷별로 미리 스카이라인 연산을 수행함으로써 비교 연산의 응답 속도를 향상시켰다.

향후 연구 내용으로는 스카이라인 그룹의 비교 연산을 감소시키고, 스트림 환경에의 적용을 고려한다.

참 고 문 헌

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

[2] W. Jin, J. Han, and M. Ester, "Mining thick skylines over large databases," PKDD, 2004.

[3] V. Hristidis, N. Koudas, and Y. Papakonstantinou, "PREFER: A system for the efficient execution of multi-parametric ranked queries," SIGMOD, 2001.

[4] Y. Yuan, X. Lin, Q. Liu, W. Wang, J. X. Yu, and Q. Zhang, "Efficient computation of the skyline cube," VLDB, 2005.

[5] J. W. Lee, G. W. You, and S. W. Hwang, "Telescope: Zooming to Interesting Skylines," DASFAA, 2007.

[6] T. Xia, and D. Zhang, "Refreshing the Sky: The Compressed Skycube with Efficient Support for Frequent Updates," ACM SIGMOD, 2006.

[7] M. E. Khalefa, M. F. Mokbel, and J. J. Levandoski, "Skyline Query Processing for incomplete Data," ICDE, 2008.

[8] J. Pei, W. Jin, M. Ester, and Y. Tao, "Catching the best views of skyline: A semantic approach based on decisive subspaces," VLDB, 2005.

[9] C. Y. Chan, H. V. Jagadish, K. L. Tan, A. K. H. Tung, and Z. Zhang, "Finding k-Dominant Skylines in High Dimensional Space," SIGMOD, 2006.

[10] Y. Tao, and D. Papadias, "Maintaining Sliding Window Skyline on Data Streams," TKDE, 2006.

[11] X. Lin, Y. Yuan, and W. Wang, "Stabbing the Sky: Efficient Skyline Computation over Sliding Windows," ICDE, 2005.

[12] C. Y. Chan, H. V. Jagadish, A. K. H. Tung, and Z. Zhang, "On high dimensional skylines," EDBT, 2006.

[13] N. Bruno, L. Gravano, and A. Marian, "Evaluating Top-k Queries over Web-accessible Databases," ICDE, 2002.

박 미 라

e-mail : happypmr@kut.ac.kr

2007년 한국기술교육대학교 인터넷미디어공학부 정보보호전공(학사)

2007년~현 재 한국기술교육대학교 정보미디어공학과 석사과정

관심분야 : 스카이라인 질의, 스트림 데이터 등

민 준 기

e-mail : jkmin@kut.ac.kr

1995년 숭실대학교 전자계산학과(학사)

1997년 한국과학기술원 전산학과(석사)

2002년 한국과학기술원 전산학전공(박사)

2003년~2004년 한국과학기술원 Post-Doc 및 초빙교수

2004년 한국전자통신연구원 선임연구원

2005년~현 재 한국기술교육대학교 인터넷미디어공학부 조교수

관심분야 : XML, 시공간DB, 스트림 데이터, 센서네트워크