Convex hull 을 사용하는 layer 기반 방법의 문제점 분석

이기은*, 박영호*
*숙명여자대학교 멀티미디어과학과
e-mail : {kieun315, yhpark}@sookmyung.ac.kr

A Problem Analysis of Layer-based Methods using Convex Hulls

Ki-Eun Lee*, Young-Ho Park*
*Dept. of Multimedia Science, Sookmyung Women's University

요 약

인터넷의 발달로 데이터의 양이 기하급수적으로 증가함에 따라 대용량 데이터를 효율적으로 검색하는 top k 질의 처리의 중요성이 커지고 있다. top k 는 릴레이션에서 가장 높은 (또는 가장 낮은) 스코어를 가지는 k 개의 튜플을 반환하는 방법으로, 스코어는 사용자가 정의한 스코어링 함수를 통해 계산된다. 효율적인 top k 질의 처리를 위해서는 전체 데이터 집합 중 최소한의 서브집합만 읽어서 k 개의 결과를 구할 수 있어야 한다. 이를 위해 기존 연구들은 다양한 방법의 인덱스 생성방법을 제안했다. 본 논문에서는 그 중에서 convex hull 을 사용하여 layer list 를 생성하는 기존 연구를 조사하고 문제점을 도출한다. 기존 연구 문제점 분석은 향후 연구인 스카이라인을 사용하는 top k 질의 처리 연구의 기반이 될 것으로 예상한다.

1. 서론

인터넷의 발달로 데이터가 이질적이고 방대해짐에 따라 대용량 데이터를 빠르게 검색하는 효율적인 top k 질의 처리가 중요해 지고 있다. 데이터는 계속 증가하고 있고 빠른 시간 내에 원하는 검색 결과를 얻으려는 사용자의 요구는 커지고 있다. 따라서 중요한 k 개의 데이터를 효과적으로 찾는 것은 사용자의 큰 요구사항이 되었다.

top k 는 릴레이션에서 가장 높은 (또는 가장 낮은) 스코어를 가지는 k 개의 튜플을 반환하는 방법이다[1]. 스코어는 사용자 정의에 따른 스코어링 함수(scoring function)로 계산되며, 식 (1)과 같이 각 속성에 대한 가중치와 속성 값들의 조합을 통해서 계산된다.

$$f(t) = \sum_{i=1}^{d} p[i] * t[i] \quad (1)$$

여기서 p[i]는 튜플 t 의 i 번째 속성에 대한 사용자의 가중치이고, t[i]는 튜플 t 의 i 번째 속성 의 값이다. 속성에 대한 가중치는 사용자의 질의를 통해 들어온다. 따라서 i 번째 속성에 대한 가중치 p[i]를 d-차원에 표현한 벡터를 사용자 질의 벡터라고 한다. 여기서 d 는 쿼리 속성의 개수이다. 쿼리 속성은 튜플 t 가 가지고 있는 많은 속성들 중 사용자가 질의 시 고려하는 속성이다.

예를 들어, 사용자가 중고 자동차 매물을 검색한다고 할 때, 중고차에 대한 속성으로 마일 수, 연식, 가격, 엔진크기 등이 있을 수 있다. 사용자의 선호도에 따라 고려하는 속성과 각 속성에 대한 가중치가 다를 것이다. A 라는 사용자는 연식을 가장 중요시 하여 연식에 0.6 의 가중치, 가격에 0.3, 마일수에 0.1 의 가중치를 둘 수 있다(가중치의 합은 1). 사용자의 질의에 따라 중고차 객체에 대한 스코어는 스코어링 함수 식 (1)에 의하여 f(t) = 0.6*Age + 0.3*Price + 0.1*Mileage 와 같이 계산 될 수 있다.

top k 질의 처리의 가장 기본적인 방법은 사용자의 질의에 기반하여 각 튜플에 대한 스코어를 모두 계산하고, 튜플들을 스코어 기준으로 정렬하여 top k 를 검색하는 방법이다. 그러나 대용량 데이터베이스에서 소수의 k 개를 찾기에는 데이터베이스 전체를 읽어야 하기 때문에 적합하지 않다.

따라서 전체 데이터 집합 중 최소한의 서브집합만 읽어 k 개의 결과를 효과적으로 검색하는 방법에 대한 연구가 진행되고 있다.

top k 질의 처리 연구는 인덱스 생성방법에 따라 layer 기반 방법과 list 기반 방법으로 구분한다. 본 논문에서는 그 중에서 공간상에서 convex hull 이나 skyline 으로 layer 를 생성해서 인덱스를 구축하는 layer 기반 방법의 기존 연구를 조사하고 분석한다.

본 논문의 구성은 다음과 같다. 2 장에서는 기존의 top k 질의처리 방법의 대표적인 연구들을 분석한다. 3 장에서는 기존 방법의 문제점과 문제점을 해결하기 위한 방법에 대하여 간략하게 논의한다. 마지막으로 4 장에서는 기대효과와 함께 본 논문의 결론을 내린다.

2. 관련연구

본 장에서는 top k 질의 처리 방법 중 layer 기반 방법의 특징 및 장, 단점을 설명하고 기존 연구들의 문제점을 분석한다.

layer 기반 방법은 객체의 모든 속성의 값들을 이

용하여 객체들을 layer 들의 리스트(간단히, layer list)로 구성하는 방법이다. 즉, layer 는 속성 값들의 모든 정보를 포함하고 있다. Layer list 를 구성하면 optimally linear ordered set[2]에 의해 적어도 k 개의 layer list 만 읽으면 top k 를 구할 수 있다. 따라서 top k 질의 처리를 위해 layer list 를 생성하는 것은 효과적이다.

Layer-based 방법의 장점은 사용자 질의 벡터에 따른 질의 처리 성능 변화가 작으며, 모든 속성의 값들을 이용하여 layer 를 구성하기 때문에 성능상의 손해가 작다. 그러나 layer 를 생성하는 비용이 크기 때문에 생성 및 갱신 비용이 크다는 단점이 있다.

대표적인 layer 기반 방법 연구로 ONION[2], AppRI[3], PLindex[1], HLindex[4]가 있다.

2.1 ONION

ONION 방법은 다차원 공간에서 포인트 객체로 표현되는 객체들의 집합에 대하여 convex hull 의 정점으로 이루어진 layer 를 구성하여 인덱스를 만든다. 전체 객체들의 집합에 대하여 layer 를 구성하는데 먼저 convex hull 정점으로 이루어진 첫 번째 layer 를 구한다. 그 후에 남아있는 객체들의 집합에 대하여 convex hull 정점으로 이루어진 두 번째 layer 를 구하고, 같은 방법으로 계속해서 layer 를 구성한다. 그 결과, 가장 바깥쪽 layer 는 기하학적으로 안쪽 layer 들을 에워싸게 된다.

Chang 은 논문에서 optimally linearly ordered set 이라는 개념을 이용하여 ONION 방법을 통해 바깥쪽부터 최대 k 개의 layer 만 읽고서 top k 질의에 답할 수 있음을 증명했다. 즉, 가장 작은 k 개의 객체을 구한다고 할 때 i 번째 layer 에서 사용자 질의 벡터에 대한 스코어가 sub-layer 들에 있는 모든 객체보다 작거나 같은 적어도 하나의 객체가 있다. 따라서 top k 질의는 k 개의 layer 에 있는 객체들만 읽으면 구해질 수 있음을 증명했다.

그러나 ONION 방법은 convex hull 을 구성하는 비용이 커서 대용량 데이터를 처리하기에는 문제가 있다.

2.2 AppRI

AppRI 는 skyline 의 domination relation 을 이용해서 layer 사이즈를 줄임으로써 쿼리 성능을 향상시켰다. domination relation 은 객체 a 가 객체 b 에 의해 배제되는지를 판단하는 것으로 배제되는 객체를 precomputing 하여 ONION 에 비해 layer 의 개수를 늘리고, layer 사이즈를 줄여서 ONION 의 단점을 해결하려고 하였다.

그러나 domination relation 을 정확하게 계산하기 때문에 시간 비용이 크다는 단점이 있다.

2.3 PL index

PL-index 는 객체가 표현되는 공간을 subregion 으로 분할 하여 subregion 내에서 convex skyline 을 이용하여 layer 를 생성하는 방법이다. convex skyline 은 Heo 가 제안한 개념으로 convex hull 과 skyline 을 결합한 방법이다. 객체에 대해 skyline 을 구하여 convex hull 생성 대상이 되는 객체의 수를 줄이고, skyline 에 의해 필터링 된 객체를 기반으로 convex hull 을 생성하는 방법이다.

PL-index 는 공간 분할과 convex skyline 생성을 통해 layer 의 사이즈를 줄였지만 차원이 높아졌을 경우에 문제가 있다. 차원이 높아지면, 즉, 속성의 개수가 많아지면, 대부분의 객체가 skyline 포인트가 되기 때문에 단순히 convex hull 을 구하는 것과 convex skyline 을 구하는 데 별 차이가 없다. 따라서 convex hull 로 layer 를 구성하는 비용이 크다는 문제점이 여전히 존재한다.

2.4 HL index

HL-index 는 ONION 방식과 TA 방식[5]을 조합한 방법으로 layer 별로 TA 와 같은 방법으로 sorted list 를 구성한다. layering 단계에서 convex hull 을 찾고, listing 단계에서 각 속성에 대하여 오름차순으로 sorted list 를 구성한다. layering 단계와 listing 단계를 반복하여 layer 와 리스트를 구성한 후 layer 에 속하는 일부 객체들만 읽어서 top k 결과를 구한다.

HL index 는 데이터를 읽는 코스트는 작지만 고차원에서 convex hull 로 layer 를 구성하는 코스트가 크다.

3. 향후 top k 연구 방안

본 장에서는 2 장에서 설명한 기존 연구의 문제점에 대하여 향후 연구할 내용에 대해 간략하게 설명한다.

관련연구에서 살펴보았듯이 convex hull 은 고차원의 경우 생성 비용이 높아진다는 단점이 있다. 그 이유는 convex hull 포인트 사이의 임의의 면에 대해 새로 들어온 포인트가 convex hull 밖에 있는지 안에 있는지 일일이 계산해야 하기 때문이다.

이에 대한 해결책으로 convex hull 에 비해 상대적으로 비용이 저렴한 skyline 을 이용한다. 그러나 skyline 은 convex hull 에 비해 layer 사이즈가 크기 때문에 skyline 을 필터링하여 convex hull 사이즈로 줄이는 연구가 필요하다.

4. 결론 및 기대효과

본 논문에서는 top k 질의 처리를 위한 기존 연구 중 layer 기반 방법에 관한 기존 연구들을 살펴보았다. 기존 방법의 대부분은 convex hull 을 통해서 layer 를 생성하기 때문에 고차원 데이터에 대해서 layer 생성 비용이 크다는 단점이 있다. 이에 대한 해결책으로 상대적으로 생성 비용이 저렴한 skyline 으로 layer 를 생성하는 방법을 제안 할 수 있다. 그러나 skyline 은 convex hull 에 비해 layer 사이즈가 크기 때문에 일부 포인트를 필터링하는 연구가 필요하다.

참고문헌

[1] J. Heo, K. Whang, M. Kim, Y. Kim, and I. Song, "The partitioned-layer index: Answering monotone top-k queries using the convex skyline and partitioning-merging technique" In *Information Sciences: an*

International Journal, Vol.179 Issue.19, Sept. 2009

[2] Chang, Y. C., Bergman, L., Castelli, V., Li, C.-S., Lo, M.-L., and Smith, J. R., "The Onion Technique: Indexing for Linear Optimization Queries," In Proc. Int'l Conf. on Management of Data, *ACM SIGMOD*, pp.391-402, Dallas, Texas, May 2000.

[3] D. Xin, C. Chen, and J. Han, "Towards robust indexing for ranked queries." in *VLDB*, 2006.

[4] J. Heo, J. Cho, and K. Whang, "The Hybrid-Layer Index: A synergic approach to answering top-k queries in arbitrary subspaces," In *Proceedings of ICDE'2010*, pp.445-448, 2010.

[5] Fagin, R., Lotem, A., and Naor, M., "Optimal Aggregation Algorithms for Middleware," In *Proc. ACM Symposium on Principles of Database Systems (PODS),* pp.102-113, Santa Barbara, California, May 2001.