선호 차원과 배척 차원을 모두 고려한 top-k 질의 처리 연구 조사

이준영*† , 서인**, 최동준***, 김경민**, 김동원*,
*포항공과대학교 컴퓨터공학과
**포항공과대학교 창의IT융합공학과
***포항공과대학교 수학과
e-mail:juneyoung.lee@sf.snu.ac.kr, iseo@dblab.postech.ac.kr,
dj603@postech.ac.kr, kmkim@dblab.postech.ac.kr,
eastcirclek@postech.ac.kr

Survey on Top-k Query Processing Considering Attractive and Repulsive Dimensions

Juneyoung Lee*† , In Seo**, Dong-june Choi***, Kyoungmin Kim**,
Dongwon Kim*
*Dept of Computer Science and Engineering, POSTECH
**Dept of Creative IT Engineering, POSTECH
***Dept of Mathematics, POSTECH

요 약

Top-k 질의란 주어진 조건을 만족하면서 높은 점수를 가진 상위 k개의 레코드를 요청하는 질의이다. 개체의 점수를 계산하는 랭킹함수가 단조함수가 아닐 경우 발생하는 기술적 어려움을 해결하기 위한 여러 연구가 있었다. 본 논문에서는 이들 중 각 차원이 선호 차원과 배척 차원으로 나뉘는 비단조 랭킹함수를 효율적으로 처리하는 기존의 top-k 질의 처리 기법들을 소개하고 비교한다.

1. 서론

Top-k 질의란 주어진 조건을 만족하면서 가장 높은(혹은 낮은) 점수를 갖는 상위 k개의 레코드를 요청하는 질의이다. Top-k 질의는 business intelligence, e-commerce, drug discovery 등의 분야에서 다양하게 활용될 수 있다 [1]. 예를 들어, 주식 투자에 적합한 회사를 k개 골라서 추천해주는 경우가 있다. 이때 사용자는 규모, 매출, 전망을 고려해서 top-k 질의를 요청할 수 있다. 또 다른 예로는 사용자로부터 거리가 가깝되 평점이 높은 음식점을 k개 추천하는 것이 있다.

랭킹함수란 인자로 받은 개체의 점수를 계산하는 함수이다 [1]. 데이터베이스가 질의에 빠르게 답하기 위해서는 색인 생성과 같은 과정을 거쳐야 하는데, 랭킹함수는 질의와 함께 주어지고, 랭킹함수와 질의에 따라 적합한 색인이 다르기 때문에 색인을 생성하는 작업이 어렵다.

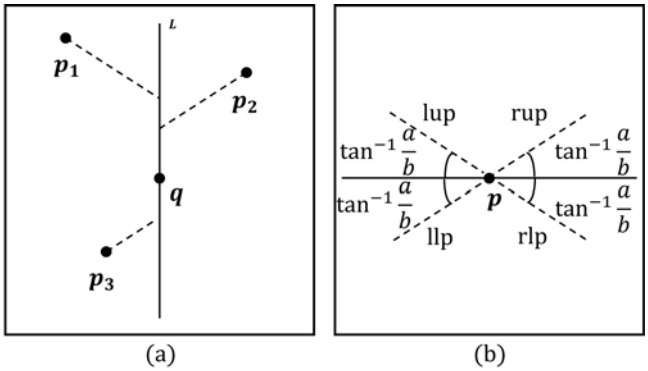
문제를 간단히 하기 위해서, 랭킹함수가 가진 속성을 활용해서 top-k 질의를 처리하는 알고리즘들이 제기되었다. 랭킹함수가 실수(real number)를 정의역으로 가질 때, 임의의 두 실수 x, y에 대해 $x \leq y$ 이면 $f(x) \leq f(y)$을 항상 만족할 때, 랭킹함수 f는 단조증가함수이다. 반대로, $f(x) \geq f(y)$를 만족할 경우, f는 단조감소함수이다. 만약 f가 단조증가 혹은 단조감소 함수일 경우 f를 단조함수라 한다. 만약 f가 실수 튜플을 정의역으로 가질 때, 모든 차원에서 단조성을 띈다면 f 또한 단조함수로 정의한다. 랭킹함수가 단조함수일 때 top-k 질의를 효과적으로 처리할 수 있는 기법은 여러 가지가 알려져 있다 [2, 3, 4]. 하지만 기존의 방법들은 랭킹함수가 단조함수가 아닐 경우 사용이 어려우며, 이것을 해결하기 위한 여러 연구가 있었다 [1, 5, 6].

이번 논문에서는 각 차원이 선호 차원과 배척 차원으로 나누어 졌을 때, top-k 질의를 효율적으로 처리할 수 있는 기존의 기법들을 조사하였다. 여기서 선호 차원이란 레코드와 질의를 고차원의 점으로 나타낼 때, 서로 비슷한 값을 가질수록 좋은 차원을 의미한다. 반대로 배척 차원은 레코드와 질의가 서로 먼 값을 가질수록 좋은 차원을 의미한다. 예를 들어, 레코드 및 질의를 2차원상의 점 (x, y)로 표현할 때, x는 선호 차원이고 y는 배척 차원일 경우, top-k 질의는 x값은 질의 점과 최대한 가깝고 y값은 최대한 먼 레코드들을 순서대로 반환하게 된다. 이때, 각 차원에서 질의 점과의 거리 계산이 필요하고, 두 점과의 거리를 계산하는 함수는 거리라는 개념의 특징상 대부분 비단조함수로 정의된다. 따라서 기존의 단조 랭킹함수만 지원하는 기법들은 사용할 수 없으며 비단조 랭킹함수들을 지원하는 top-k 질의 처리 기법이 필요하다.

본 논문은 2017년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. NRF-2014R1A2A2A01004454)

† 이 연구원은 포항공과대학교 컴퓨터공학과 소속으로 이 연구에 참여하였으며, 현재는 서울대학교 컴퓨터공학부 소속이다.

본 논문의 구성은 다음과 같다. 2절에서는 특정한 형태의 비단조 랭킹함수만을 지원하는 top-k 질의 처리 기법 SD-색인[1]을 소개한다. 3, 4절에서는 일반적인 비단조 랭킹함수를 모두 지원할 수 있는 top-k 질의 처리 기법인 BRS[5]와 점진적 탐험 기법[6]을 소개한다. 5절에서는 각 기법의 선호 차원과 배척 차원을 고려한 top-k 질의 처리 성능을 비교한다.



(그림 1) (a) 질의 q가 주어졌을 때 p_1, p_2, p_3 의 등치선
(b) 레코드 p가 가질 수 있는 등치선의 네 가지 형태

2. SD-색인 (SD-Index)

SD-색인[1]은 선호 차원과 배척 차원을 고려한 비단조 랭킹함수 중 특정한 형식의 함수만을 지원하는 메인 메모리 기반의 top-k 질의 처리 기법이다. SD-색인은 [선호 차원에 대한 맨해튼거리 - 배척 차원에 대한 맨해튼거리] 형태의 함수를 SD-Score로 정의하여 랭킹함수로 사용한다. 예를 들어, 레코드 및 질의를 2차원상의 점 (x, y)로 표현할 때, x는 선호 차원이고 y는 배척 차원일 경우, SD-색인은 아래와 같은 형태의 랭킹함수를 처리할 수 있다. 여기서 p와 q는 각각 레코드와 질의 점이고, a와 b는 각각 x와 y차원에 대한 가중치이다.

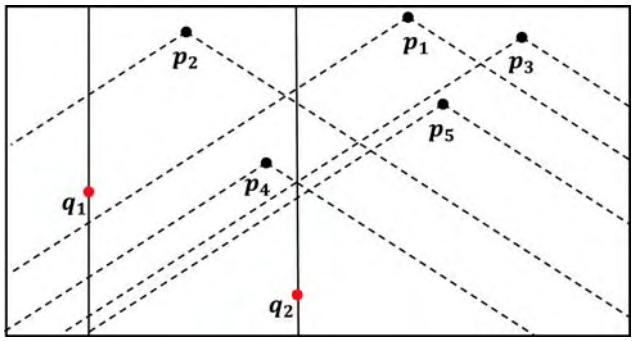
$$SD-Score(p,q) = a|p_x - q_x| - b|p_y - q_y|$$

1) 2차원에서 top-k 질의 처리

2차원에서 정의된 레코드 및 질의에 대한 질의를 빠르게 풀기 위해, SD-색인 방법은 등치선(isoline)을 정의한다. 어떤 레코드 p, 질의 q가 주어졌을 때, 랭킹함수 $SD-Score(p,q) = SD-Score(p',q)$[1]를 만족하는 점 p'를 2차원 평면에 그렸다고 가정하자. 이때, 2차원 평면에는 두 종류의 직선이 그려지게 되는데, 한 종류는 기울기가 a/b인 직선이고, 다른 한 종류는 기울기가 $-a/b$인 직선이다. 이 두 기울기는 질의 q에 영향을 받지 않으며, 따라서 a, b를 미리 알고 있을 때, 각 레코드 p에 대해 기울기 a/b와 $-a/b$를 가진 두 직선을 미리 그려둘 수 있다. 이것을 레코드 p의 등치선이라고 한다.

등치선은 y축에 평행하고 질의 q를 지나는 직선 L과 교차할 수 있으며, 레코드 p의 점수는 등치선과의 교차점의 y 좌표를 질의 q의 y 좌표에서 뺀 것과 정확히 같다. 예를 들어, 그림 1. (a)에서 $SD-Score(p_1,q)$ 직선 L과 p_1의 오른쪽 아래 등치선의 교점의 y 좌표를 q의 y 좌표로부터 뺀 것과 같다. 따라서, top-k 질의를 처리하는 것은 직선 L과 등치선들 사이의 교점들을 빠르게 구하는 것과 동일하다.

그림 2는 질의가 주어졌을 때 등치선과 직선 L간의 교점을 나타내는 그림이다. 질의 q_1의 경우, 점수가 가장 좋은 레코드는 순서대로 p_2, p_1, p_4, p_3 그리고 p_5 순이다. 질의 q2의 경우, 점수가 가장 좋은 레코드는 p_1, p_2, (p_3 = p_4), p_5 순이다.



(그림 2) 다섯 개의 레코드의 등치선(lp/rlp만 표기)들과 두 질의 q1, q2

등치선에 대해 자세히 설명하자면 다음과 같다. 그림 1. (b)에 나타난 바와 같이, 등치선의 종류는 총 네 가지가 있다. (left/right, upper/lower) 이 때 등치선이 x축과 이루는 각도는 모두 $\tan^{-1}\frac{b}{a}$ 로 같다. 즉, 어떤 질의 q가 주어지든, 등치선은 모두 같은 모양을 가지게 된다. 따라서 등치선을 미리 계산하기가 매우 수월하다.

등치선과 직선 L의 교점을 구하는 것은 레코드 p가 질의 q를 기준으로 어디에 있는지와 관련이 있다. 예를 들어, p가 q의 왼쪽에 있을 때 (그림 1. (a)의 p_1), 등치선 rup/rlp만 L과 교차하고, lup/llp는 교차하지 않는다.

등치선의 각도 (a/b)가 고정되었을 때, SD-색인 방법은 직선 L과 등치선들 사이의 교점을 빠르게 구하기 위해 트리구조의 색인을 제안한다. 이 색인의 핵심은, 어떤 질의 q가 주어졌을 때, q의 왼쪽에 있는 등치선들과 q의 오른쪽에 있는 등치선 중 가장 점수가 좋은 것들을 빠르게 가져오는 것이다. 총 4종류의 등치선이 있으므로, 최종적으로는 4개의 등치선을 서로 비교하게 된다.

색인의 단말 (leaf) 노드는 실제 레코드를 나타내며, x축 순으로 정렬이 되어 있다. 색인의 비단말(non-leaf) 노드는 자식 노드들의 등치선과 $x = \pm\infty$의 교점을 구하여 가장 점수가 좋은 것을 저장한다. 총 4개의 등치선이 있으므로, 비단말 노드는 총 4 개의 교점을 저장하게 된다.

[1] 함수값이 작은 레코드를 더 선호한다고 하자.

만약 어떤 질의 q가 주어졌을 때 top-k 질의를 처리하는 방법은 다음과 같다. 먼저, 색인 트리의 루트 노드에서 단말 노드까지 가는 경로를 찾는다. 이것을 분리 경로(separating path)라고 한다. 그리고 이 분리 경로로부터 왼쪽에 있는 색인 노드들로부터, 등치선 rup/rlp 중 가장 좋은 것을 얻을 수 있다. 그리고 분리 경로로부터 오른쪽에 있는 색인 노드들로부터 등치선 lup/llp 중 가장 좋은 것을 얻을 수 있다. 이것을 비교하면 가장 좋은 레코드를 하나 얻을 수 있다. k번째로 좋은 레코드를 얻는 과정은 위 알고리즘을 조금 변형하면 된다.

2) 2차원에서 top-k 질의 처리 (일반화)

위 경우는 랭킹함수가 고정되어서 a, b 값이 더 이상 바뀌지 않을 때 동작한다는 한계점이 있다. 일반적으로 랭킹함수는 질의와 함께 주어지기 때문에, 지금과 같이 a 및 b 값을 미리 알기는 힘들다. 이 문제를 해결하기 위해 SD-색인은 여러 등치선의 기울기 (a/b) 에 대해 미리 색인을 생성한 뒤, 새로운 랭킹함수가 오면 가장 가까운 기울기에 대해 만들어진 두 색인을 활용해서 결과를 빠르게 얻는 기법을 제안한다. 예를 들어, 새로운 랭킹함수의 a/b = 0.5이고, 이전에 a/b = 0.3, 0.6, 0.9 에 대해 이미 색인이 만들어 져 있다고 할 때, SD-색인 기법은 a/b = 0.3, 0.6인 두 색인을 잘 활용해서 a/b = 0.5일 때의 top-k 결과를 만들어 낸다.

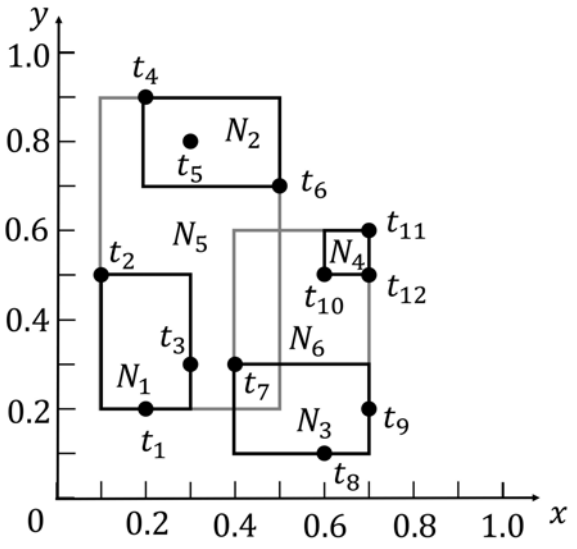
3) 고차원 데이터에서 top-k 질의 처리

SD-색인 방법은 3차원 이상의 top-k 질의 처리를 위해 2차원에서의 top-k 질의 처리 기법을 활용한다. 이 방법은 2차원 질의의 합으로 표현될 수 있는 특정한 꼴의 랭킹함수에 대해서 top-k 질의 결과를 올바르게 계산한다.

N차원 공간을 이루는 차원 중, 선호 차원들의 모음을 S, 배척 차원의 모음을 D 라고 하자. 정의에 따라, $|S|+|D|=N$을 만족한다. 그 다음, S의 각 원소를 D의 각 원소와 짝지어서, 1대1 대응을 만든다. 각 짝지어진 차원들에 대해선 모두 위에서 명시한 2차원 top-k 질의 문제로 해결될 수 있다. 짝지어지지 못한 차원의 경우, naive한 알고리즘을 이용해 빠르게 top-k 결과를 얻어올 수 있다. 각각의 작은 문제들에 대해 top-k 결과가 얻어지면, 이들을 임계값 기법(threshold algorithm)과 유사한 방법을 사용해서 최종 결과를 얻는다.

3. BRS 기법

BRS(Branch-and-bound Ranked Search)[5]는 비단조 랭킹함수를 지원하는 디스크 기반의 top-k 질의 처리 기법으로 모든 차원이 색인되는 하나의 R-트리를 사용한다. Top-k 질의를 처리하기 위해, BRS 알고리즘은 첫 번째로 R-트리의 각 노드에 대해 점수를 매기는 방법을 제시한다. R-트리의 단말 노드(leaf node)의 점수는 그 단말



(그림 3) 레코드 t1~t12 로부터 만든 R-트리

노드가 가진 데이터의 점수이다. 비단말 노드(non-leaf node)의 점수는 그 노드에 포함될 수 있는 노드들의 점수 중 가장 좋은 값(최댓값 혹은 최솟값)을 부여한다.

그림 3은 12개의 단말 노드 t_1 ~ t_{12}, 그리고 6개의 비단말 노드 N_1 ~ N_6로 이루어진 R-트리를 나타낸다. 편의상 루트 노드는 생략했다. 이 때, 비단말 노드 N_2의 점수는 (0.2, 0.7) ~ (0.5, 0.9)를 덮는 사각형 내에서 가능한 점들의 점수 중 가장 좋은 값이다.

두 번째로, 힙 자료구조를 사용해서 점수가 좋은 단말 노드 k개를 찾는다. 힙 자료구조에는 가장 처음엔 R-트리의 루트 노드만 들어가 있다. 그다음부터, 힙으로부터 가장 점수가 좋은 R-트리 노드를 뽑고, 그 노드의 자식 노드를 다시 힙에 넣는 것을 반복한다. k개의 단말 노드가 나올 때까지 이것을 반복한다. 임의의 R-트리 노드의 점수는 자식 노드들의 점수보다 항상 더 좋거나 같게 평가되므로, 중간에 더 최적일 수 있었던 비단말 노드를 빠뜨리는 일이 없으므로 이 알고리즘은 옳다.

랭킹함수가 단조함수일 때 비단말 노드의 점수는 쉽게 계산될 수 있다. 2차원 사각형의 경우 항상 네 꼭지점 중 하나가 가장 좋은 점수가 된다. 랭킹함수가 비단조함수일 때는, 랭킹함수의 기울기를 구해서 최댓값 또는 최솟값을 구하는 방법을 사용한다. 만약 모든 ∂x_i 에 대해서 $\partial f/\partial x_i$ 를 계산할 수 있다면, $\nabla f(\vec{x})=0$인 지점 $\vec{x}$를 찾음으로써 범위 내에서 최댓값 또는 최솟값을 찾을 수 있다.

4. 점진적 탐험 (Progressive Exploration)

점진적 탐험[6]은 BRS와 같이 색인을 사용한 디스크 기반의 top-k 질의 처리 기법으로 비단조 랭킹함수를 처리할 수 있다. 점진적 탐험은 모든 차원을 하나의 R-트리에 색인하는 BRS와 달리 각 차원별로 혹은 여러 차원의 집

합별로 분리된 색인을 유지하여 차원이 큰 데이터에 대해 효율적으로 top-k 질의를 처리할 수 있다. 점진적 탐험은 다수의 색인 노드들로 구성되는 결합 상태(joint state)들의 영역에 대해 점진적 검색을 수행하는 색인-병합 프레임워크를 활용한다. 예를 들어, 2개의 색인 T_1과 T_2가 있다고 가정하자. 여기서 T_1의 어떤 색인 노드 n_i와 T_2의 어떤 색인 노드 n_j의 결합 상태인 (n_i, n_j)를 구하기 위해 n_i와 n_j의 자식 노드들의 카티션(Cartesian) 곱을 자식 상태들에게 수행한다. 점진적 탐험은 이러한 결합 상태들이 가질 수 있는 점수 중 가장 좋은 값을 계산함으로써, 결합 상태들의 탐색 순서를 선택하고, 해답이 아닐 것으로 확실시되는 결합 상태들을 미리 제외한다.

그러나 점진적 탐험의 이러한 처리 방법은 비단조 랭킹함수를 처리하는데 있어 심각한 성능 저하를 초래할 수 있다. 점진적 탐험은 결합 상태들의 영역에 대한 점진적 검색을 수행하기 위해 하향식으로 검색을 수행한다. 결합 상태들의 영역을 루트부터 단말까지 순회(즉, 확장)하기 위해서, 점진적 탐험은 임계값 확장(threshold expansion)이라 불리는 알고리즘을 수행한다. 그러나 이 알고리즘은 다음에 확장할 결합 상태를 구하기 위해 카티션 곱의 개수만큼 자식 상태의 개수를 반복적으로 생성해야 하는 심각한 문제점을 가진다.

이 문제를 해결하기 위해 점진적 탐험 기법에서는 두 가지 해답을 합친 방법을 제시한다, 첫 번째는 확장을 게으르게(lazily) 진행하는 것이고, 두 번째는 레코드를 포함하지 않는 상태를 미리 검색 영역에서 제외하는 것이다.

5. 성능 비교

본 절에서는 2, 3, 4 절에서 소개한 SD-색인, BRS, 점진적 탐험 방법의 성능을 비교하기 위해 SD-색인 논문[1] 에서 실험한 결과를 소개한다. 실험 환경은 3.2GHz, 4GB 메모리이고 Debian Linux 4.0이다. BRS 기법, 점진적 탐험 기법, SD-색인 기법을 각각 자바로 구현하였다. 데이터셋은 균일(uniform0 / 상호 관계(correlated) / 상호 역관계(anti-correlated) 분포를 따르는 1000만 개의 합성 데이터를 사용하였다. 랭킹함수는 SD-색인에서 처리 가능한 SD-Score 함수를 사용하였으며, 가중치는 균일 분포를 이용해 0과 1사이의 값을 생성하여 사용하였다. 질의는 균일 분포를 이용하여 100개의 질의 점을 임의로 생성하였다. 상위 1개 및 5개의 결과를 얻는 실험을 했으며 (top-1 및 top-5) 각 데이터의 차원 또한 2, 4, 6, 8개로 변화시키며 실험하였다. 실험 결과, SD-색인 기법이 BRS 기법 및 점진적 탐험 기법에 비해 여러 데이터셋에서 수십 ~ 수백 배 더 빠르게 질의를 처리하였다. BRS는 대부분의 경우 점진적 탐험보다 더 빨랐으며 최대 1.8배 정도 차이가 났다. 일부 작은 데이터셋에서는 점진적 탐험이 BRS보다 더 빠른 경우도 있었으나 데이터셋의 크기가 증가할수록 BRS보다 느려졌다. 자세한 실험 결과는 SD-색인 논문[1]에 수록되어 있다.

6. 결론

본 논문에서는 각 차원이 선호 차원과 배척 차원으로 나뉘는 비단조 랭킹함수들을 처리할 수 있는 top-k 질의 처리 기법 세 가지를 소개하였다. SD-색인은 SD-Score라는 특정 랭킹함수만을 지원하는 기법으로 랭킹함수의 특징을 활용하여 top-k 질의를 효율적으로 처리한다. BRS 기법은 R-트리에서 각 비단말 노드에 점수를 매기고, 힙 자료구조를 이용해 가장 좋은 k개의 단말 노드를 찾을 때까지 비단말 노드들을 확장하여 top-k 질의를 계산한다. 점진적 탐험 방법은 여러 차원의 집합별로 분리된 색인의 색인 노드들로 구성되는 결합 상태들의 영역에 대해 점진적 검색을 수행하는 방법으로 임계값 확장이라는 알고리즘을 이용해 루트부터 단말까지 결합상태들의 영역을 순회하는 기법이다. 세 방법의 성능을 비교한 실험에서는 SD-색인이 다른 두 방법에 비해 수십 ~ 수백 배 빨랐다. 그러나 SD-색인은 다른 두 디스크 기반 기법과 달리 메인 메모리 기반의 기법임을 고려해야 하며, 특정한 형식의 랭킹함수만을 처리할 수 있다는 단점이 있다.

비단조함수에서 top-k 질의의 성능을 발전시키기 위해서는 아래와 같은 방법이 있다. 첫째로, SD-색인과 같은 기법을 제한적인 꼴이 아닌 조금 더 다양한 종류의 함수에 대해서 확장할 수 있을 것이다. 등치선을 직선이 아닌 곡선으로 확장함으로써 이것이 가능할 것이다. 둘째로, BRS 기법을 사용할 때 비단말 노드의 점수를 조금 더 효과적으로 매길 수 있을 것이다. 셋째로, 질의의 종류에 따라 세 기법 중 효율적이라 판단되는 것을 사용함으로써 전체적인 성능 향상을 보일 수 있을 것이다.

참고문헌

[1] S. Ranu, A. K. Singh. Answering Top-k Queries Over a Mixture of Attractive and Repulsive Dimensions. Proceedings of the VLDB, Endowment, Vol. 5, No. 3, 169-180, 2011

[2] R. Fagin. Combining fuzzy information: an overview. SIGMOD Record, 31(2):109-118, 2002

[3] R. Fagin. Fuzzy queries in multimedia database systems. In PODS, pages 1-10, 1998.

[4] R. Fagin A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In PODS, 2001.

[5] Yufei Tao, et al. Branch-and-bound processing of rankedqQueries, Information Systems, Volume 32, Issue 3, May 2007, pages 424-445

[6] Dong Xin, et al. Progressive and selective merge : computing top-k with ad-hoc ranking functions, SIGMOD '07 Proceedings of the 2007 ACM SIGMOD international conference on Management of data, Pages 103-114