SKYLINE을 이용한 개인 맞춤형 우선 순위 처리 기법

김준한*, 유헌창**
*고려대학교 컴퓨터정보통신대학원
**고려대학교 대학원 컴퓨터학과
e-mail : {lamyun10, yuhc}@korea.ac.kr

Personalized Priority Technique Using SKYLINE

Jun-Han Kim*, Heonchang Yu**
*Graduate School of Computer & Information Technology, Korea University
**Department of Computer Science and Engineering, Korea University

요 약

사용자들이 원하는 최적화 된 정보를 찾기 위해 다양한 기법을 이용하여 신속하고, 좋은 조건으로 찾기를 원한다. 그리고 사용자들 마다 다른 요구 조건들이 발생할 수가 있다. 기존 스카이라인의 탐색 기법에서는 요구 조건들이 한정적 이었기에 사용자의 다른 요구 조건들이 있어도 그것에 맞게 검색할 수 없었다. 개인 맞춤형 우선 순위 처리 기법을 통하여 다른 요구 조건들을 반영하여 만족도를 높일 수 있고, 불필요한 데이터들을 제거하여 스카이라인 탐색에 소요되는 시간을 감소시킬 수 있다.

1. 서론

오늘 날, 사용자들이 원하는 최적화 된 정보를 찾기 위해 다양한 기법을 이용하여 신속하고, 좋은 조건으로 찾기를 원한다. 사용자들의 요구 조건을 충족하기 위해 여러 분야 에서 노력이 이루어지고 있다. 이러한 노력 속에 스카이라인은 사용자들이 원하는 다양한 요구 조건으로 신속하고, 좋은 조건을 찾기 위해 사용되어지는 기법들 중에 하나의 대표적인 기법이다. 스카이라인은 사용자들의 요구 조건에 따라 많은 데이터들 사이의 지배 유무를 확인한다. 그리하여 지배되지 않은 데이터들의 최소 집합을 최적의 요구조건에 적당한 것으로 찾아 선정하고 구성하는 것이 스카이 라인이라고 한다.

스카이라인을 만들기 전에 사용자들 마다 다른 요구 조건들이 발생할 수가 있다. 기존 스카이라인의 탐색 기법에서는 요구 조건들이 한정적 이었기에 사용자의 다른 요구 조건들이 있어도 그것에 맞게 검색할 수 없었다. 따라서 이 논문에서는 사용자들의 요구 조건에 맞는 데이터를 가지고 스카이라인 탐색을 통하여 사용자의 만족도와 탐색의 성능을 향상시킬 수 있는 방법을 제안한다. 이를 위해 사용자들의 개인맞춤형 우선순위 처리기법을 제안한다.

이 기법을 통하여 사용자들의 다른 요구조건들을 반영하여 만족도를 높일 수 있고, 불필요한 데이터들을 제거하여 스카이라인 탐색에 소요되는 시간을 감소시킬 수 있다.

2. 관련연구

이 장에서는 연구의 바탕이 되는 스카이라인의 탐색 기법을 알아보고자 한다. 이를 위하여 2.1.1절에서는 스카이라인 기법 중 가장 기초가 되는 탐색 기법인 BNL(Block Nested Loops)에 대하여 알아보고, 2.1.2절에서는 BNL이 가지고 있는 문제점을 해결하기 위하여 데이터의 위상 기대치가 갖는 특징을 이용한 SFS(Sort Filter Skyline)에 대하여 설명한다.

2.1 스카이라인 탐색 기법

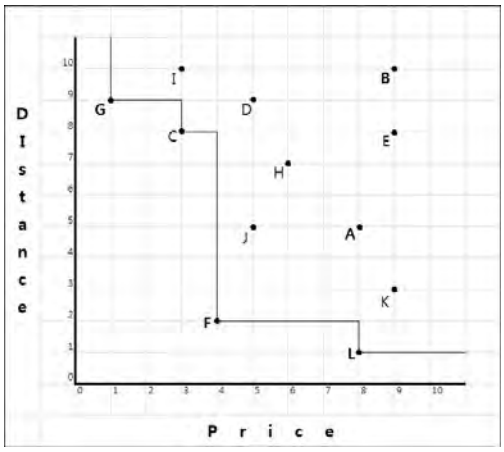
Hotel	Price	Distance
A	8	5
B	9	10
C	3	8
D	5	9
E	9	8
F	4	2
G	1	9
H	6	7
I	3	10
J	5	5
K	9	3
L	8	1

(그림 1) 숙박료와 거리의 데이터 값

(그림 1)은 스카이라인 처리를 위해 숙박료와 거리를 데이터 값으로 표현한 것이다. 데이터 값을 가지고 지배당하는 객체를 걸러내는 과정을 진행한다.

여기서 지배당하는 객체란 다른 객체들보다 우월하지 않아서 대상이 되지 못한 것을 말한다. (그림 1)에서 가격이나 거리가 작은 값에 우월하다고 가정할 때, J와 H의 경우 F보다 가격도 비싸고 거리도 멀기 때문에 J, H보다 F를 관심있는 객체로 판단할 것이다. 이때 F는 지배하는

객체라 하며, J, H는 지배당하는 객체라 불리며 스카이라인 결과에 포함되지는 않는다. 여기서 G, C, L 역시 마찬가지로 지배를 당하지 않기 때문에 스카이라인 결과에 포함된다. 이렇게 지배하고 있는 점들을 연결하여 스카이라인 결과를 나타낸 것이 (그림 2) 이다.



(그림 2) 스카이라인의 결과

(그림 2)와 같이 스카이라인의 탐색 기법들 중 대부분은 사용자들이 원하는 최적화된 정보를 찾기 위한 비교 질의 연산이다. 현재 연구되고 있는 탐색 기법이 많지만, 그 중에서 대표적인 탐색기법에는 BNL(Block Nested Loops)와 SFS(Sort Filter Skyline)가 있다.

2.1.1 BNL(Block Nested Loops)

BNL 이전의 스카이라인 탐색 기법들은 데이터베이스로부터 스카이라인을 탐색하기 위해 기존의 SQL(Structured Query Language)을 이용하거나 별도의 프로그램을 구현하여 사용하는 방식이 대부분 이었다. 하지만 SQL만을 이용하는 경우에는 복잡한 질의 구문을 요구하기 때문에 성능이 좋지 못했으며, 별도의 프로그램을 구현하는 경우에는 스카이라인을 탐색하기 위해 스카이라인에 특화된 인덱스 구조를 요구하는 경우가 대부분이었기 때문에 모든 상황에서 최적화된 동작을 보일 수 없었다. 이러한 문제를 해결하기 위해 기존의 SQL과 통합될 수 있는 SKYLINE OF 구문의 제안과 함께 해당 구문에 적합한 기법으로 BNL을 제안하였다.

```
SELECT … FROM … WHERE …
GROUP BY … HAVING …
SKYLINE OF [DISTINCT] d1 [MIN | MAX | DIFF], …, dm [MIN | MAX | DIFF]
ORDER BY …
```

(그림 3) SQL과의 통합을 위해 제안된 SKYLINE OF 구분

BNL은 스카이라인을 탐색하기 위해 윈도우(Window)라는 저장 공간을 이용한다. 스카이라인은 윈도우에 저장된 스카이라인 후보자들과 테이블로부터 순차적으로 읽어 온 입력 데이터들 사이의 비교 질의를 통해 탐색되게 되는데, 비교 질의 과정에서 윈도우에 저장된 스카이라인 후보자가 비교중인 데이터를 지배하게 되면 해당 데이터와의 비교 연산은 바로 종료되며 다음의 입력 데이터와 스카이라인 후보자와의 비교가 다시금 반복된다. 하지만 반대로 스카이라인 후보자가 비교중인 입력 데이터에 의해 지배당하는 경우에는 해당 스카이라인 후보자는 입력 데이터에게 지배되었기 때문에 더 이상 스카이라인이 될 수 없으므로 윈도우에서 제거한다. 이 같은 스카이라인 후보자와 입력 데이터 사이의 비교 질의에서 마지막까지 지배되지 않은 입력 데이터는 윈도우에 새로운 스카이라인 후보자로써 저장되고 다음 데이터와의 비교 질의에서 스카이라인 후보자로 사용한다. BNL은 위와 같은 비교를 테이블의 마지막까지 반복적으로 진행하며 최종적인 비교연산 이후에 제거되지 않고 윈도우에 남아있는 스카이라인 후보자들을 지배당하지 않은 데이터들의 최소 집합이기 때문에 해당 데이터들은 스카이라인으로 확정하고 이들을 스카이라인으로 반환하게 된다.

2.1.2 SFS(Sort Filter Skyline)

SFS는 데이터들의 위상 기대치와 함께 이들이 갖게 되는 위상 수학적 특성을 이용하여 BNL이 갖는 문제점들을 해결한 기법이다. 단조 스코어 함수(Monotone scoring function)를 준수하면서 생성된 스카이라인은 계산된 스코어를 기준으로 하는 선형 순서가 위상 수학적으로 정렬되는 특징을 갖고 있는데, SFS에서는 이 특징을 이용하기 위해 위상 기대 함수를 통해 위상 기대치를 계산하고 이를 비교 질의의 최적화를 위해 활용한다. 이때 사용되는 위상 기대 함수식은 다음과 같다.

$$E(d) = \sum_{i=1}^{k} \ln(d[a_i] + 1)$$

해당 수식에서, E(d)는 데이터 d에 대해 계산된 위상 기대치를 의미하고, k는 속성의 수, $d[a_i]$는 데이터 d의 i번째 속성을 의미하며 모든 속성들은 주로 정규화를 통해 위상 기대치가 갖는 효과가 최대화되도록 하여 사용한다. 이때 해당 위상 기대 함수를 통해 계산된 위상 기대치는 각각의 속성 값이 작을수록 작은 위상 기대치를 갖게 된다. 따라서 위상 기대치가 작을수록 스카이라인으로 선택될 확률과 함께 다른 데이터들을 지배할 확률이 높아지는 특성을 갖는다. 그렇기 때문에 SFS에서는 이러한 특성을 비교 질의 과정에서 최대한 반영하기 위해 계산된 위상 기대치를 이용하여 모든 데이터들을 오름차순으로 정렬한다. 이렇게 정렬된 데이터들은 스카이라인이 될 확률이 높은 데이터부터 낮은 데이터 순으로 정렬되었다고 할 수 있기 때문에 BNL과 동일하게 윈도우를 사용하여 비교 연산을 진행하더라도 윈도우에 입력됨과 동시에 스카이라인으로 확정되며, 윈도우를 유지함에 있어서도 스카이라인 후보자의 교체가 발생하지 않기 때문에 BNL과 달리 예측 가능

하면서도 보다 적은 자원의 소모로도 윈도우를 유지할 수 있게 된다. 또한 스카이라인이 될 데이터들을 우선적으로 윈도우에 저장하기 때문에 많은 데이터들이 적은 비교 질의 연산으로도 제거되어 전체적인 수행시간이 감소하게 된다.

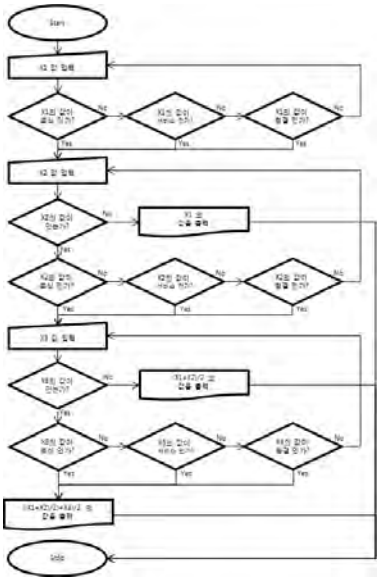
3. 개인 맞춤형에 적합한 데이터 처리 기법

이 장에서는 사용자가 요구하고 있는 기준으로부터 최적의 정보를 탐색하고 사용자가 요구하는 기준에 적합한 개인 맞춤형 우선순위 처리 기법을 제안한다.

개인 맞춤형 우선순위 처리 기법은 사용자가 원하는 다양한 요구 조건을 찾는 동시에 사용자의 요구 조건에 맞지 않은 데이터들을 제외하기 위한 기법이다.

현재 스카이라인을 탐색하기 위하여 가장 많이 사용되고 있는 기법은 SFS(Sort Filter Skyline) 이다. SFS(Sort Filter Skyline)는 데이터들의 위상 기대치가 갖는 특징을 이용하여 비교 질의 연산을 최적화함으로 전체적인 탐색의 성능을 향상시키는 기법이지만 데이터의 차원이 높아질수록 비교되어 지는 사용자의 요구 조건들이 크게 증가하는 문제가 있기 때문에 다차원의 데이터로부터 스카이라인을 탐색하는데 큰 어려움을 가지고 있다.

이러한 문제를 해결하기 위하여 사용자들의 다양한 요구조건을 충족시킬 수 있는 개인 맞춤형 우선순위 처리 기법을 제안한다.



(그림 4) 개인 맞춤형 우선순위 처리 순서도

(그림 4)와 같이 값을 개인의 우선순위에 따라 입력한 후, 결과 값에 따라 우선순위가 낮은 결과 값을 가지고 기존 데이터 값과 비교하여 불필요한 데이터들을 제외시킨다. 그 후 스카이라인 기법을 이용하여 결과를 나타냈을 경우 기존의 값과 비교해 보면 기존에는 지배되지 않은 값들도 사용자들이 원하는 우선순위에 의하여 제외되어 사용자들이 원하는 최적화된 데이터들을 찾을 수 있다.

4. 구현 및 결과 분석

이 장에서는 개인 맞춤형 우선순위 처리 기법의 이해를 돕기 위하여 사용자가 원하는 우선순위의 데이터를 가지고 기존의 스카이라인 기법과의 차이점을 확인해 보고 사용자에 따라 스카이라인 결과의 차이점을 예를 들어 설명한다.

4.1 SQL 구현

개인 맞춤형 우선 처리 기법에 대한 이해를 돕기 위하여 SQL 을 이용하여 예를 들어 자세하게 설명 한다.

SQL을 이용하기 위해선 3개의 변수값을 받아야 한다. 사용자에 맞게 변수값에 따라 CLASS FLAG1, CLASS FLAG2, CLASS FLAG3 로 이동 한다.

```
    @FLAG1      VARCHAR(50) = '',
    @FLAG2      VARCHAR(50) = '',
    @FLAG3      VARCHAR(50) = ''
                                        .
                                        .
                                        .
CLASS_FLAG1:
      EXEC ('SELECT * FROM (SELECT 업체,ROW_NUMBER() OVER(ORDER BY
'+@FLAG1+') AS 평균 FROM BIGDATA..HOTEL_DATA) AS A ORDER BY 평균 ASC')

CLASS_FLAG2:
      EXEC ('SELECT * FROM (SELECT 업체,(ROW_NUMBER() OVER(ORDER BY
'+@FLAG1+') + '+@FLAG2+')/2 AS 평균 FROM BIGDATA..HOTEL_DATA) AS A
ORDER BY 평균 ASC')

CLASS_FLAG3:
      EXEC ('SELECT * FROM (SELECT 업체,(((ROW_NUMBER() OVER(ORDER BY
'+@FLAG1+') + '+@FLAG2+')/2)+ '+@FLAG3+')/2  AS 평균 FROM
BIGDATA..HOTEL_DATA) AS A ORDER BY 평균 ASC')
```

사용자가 @FLAG1 의 값만 입력 하였을 경우, 우선 순위에 따라 결과를 출력한다. 사용자가 @FLAG1, @FLAG2 의 값을 입력 하였을 경우, @FLAG1을 우선순위에 따라 값을 차등 지급 한 후, @FLAG2 의 값을 더한 후 평균값에 따라 우선순위의 결과를 출력한다. @FLAG1, @FLAG2, @FLAG3 의 값을 입력 하였을 경우, @FLAG1 을 우선순위에 따라 값을 차등 지급 한 후, @FLAG2 의 값을 더한 후 평균값에 @FLAG3 의 값을 더한 후 평균값의 결과를 출력 한다. 결과 값에 따라 스카이 라인 기법을 적용 하여 사용자에게 값을 제공 한다.

4.2 예제를 통한 개인 맞춤형 우선순위 처리 기법

Hotel	조식	서비스	청결
A	1	3	10
B	9	4	6
C	7	3	4
D	3	10	2
E	2	4	10
F	9	4	4
G	10	4	2
H	9	3	1
I	2	2	10
J	1	4	1
K	8	2	3
L	4	2	5

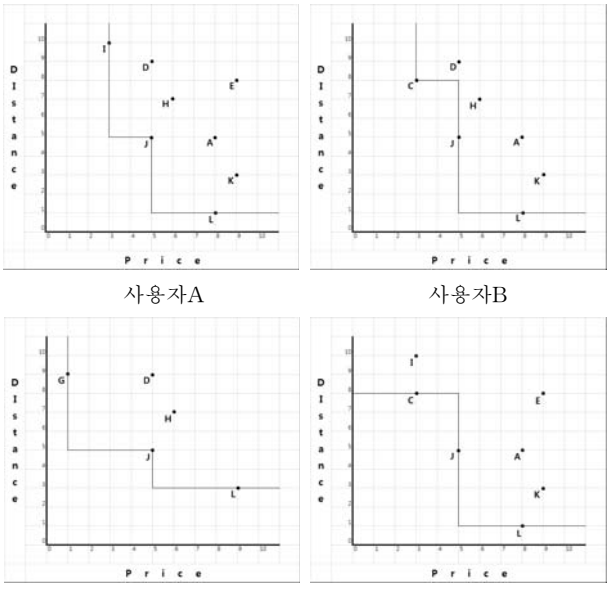
(그림 5) 업체 속성 점수표

(그림 5)는 호텔마다의 조식, 서비스, 청결도를 점수표로 입력해 놓은 데이터들이다. 값이 낮을수록 서비스의 질이 더 높은 것으로 정의한다.

사용자	X1	X2	X3
사용자A	서비스	청결	조식
사용자B	조식	청결	
사용자C	청결		
사용자D	조식	서비스	

(그림 6) 사용자 우선순위 표

(그림 6)은 사용자 마다 우선시 되는 데이터를 정리해 놓은 표이다. (그림 6)의 사용자가 우선시로 생각하고 있는 데이터를 가지고 그림4 의 순서도를 이용하여 각 업체마다의 점수표를 우선 순위화 해서 순위가 낮은 데이터를 제외한 후 스카이라인을 적용하여 표로 정리하였다. 여기에서 사용자가 우선시 생각하는 속성들의 평균값을 구해서 낮은 순위가 값들은 제외시킨 후 스카이라인을 적용하였을 경우 사용자마다 스카이라인이 틀리게 형성되는 것을 확인할 수가 있다.



(그림 7) 개인 맞춤형 우선순위 처리 기법을 적용한 예제

(그림 7)은 사용자마다의 우선시 되는 결과를 보여주고 있다. 기존의 스카이라인과 비교 했을 때, 결과가 틀린 것을 확인할 수 있다.

개인 맞춤형 우선순위 처리 기법을 통하여 불필요한 데이터들을 제거함에 따라서 기존의 스카이라인 보다 사용자들이 원하는 최적화된 데이터들을 효율적으로 찾을 수 있다.

5. 결론

사용자들이 원하는 최적화 된 데이터를 효율적으로 찾기 위하여 많은 양의 데이터로부터 스카이라인을 탐색하기 전에 개인 맞춤형 우선순위 처리기법을 적용함으로써 스카이라인 탐색에서 발생하는 비교 질의 연산을 최소화 시키는 역할을 한다.

마지막으로 개인 맞춤형 우선순위 처리기법은 다음과 같이 요약된다. 첫째, 기존의 스카이라인의 탐색에서는 제안돼 있는 조건들을 검색하였지만 사용자들마다 원하는 적합한 데이터를 최적화할 수 있다. 둘째, 불필요한 비교 질의를 최소화함으로써 스카이라인의 탐색에서 소요되는 시간을 감소시켰다.

향후 연구 과제는 다음과 같다. 첫째, 제안된 기법을 많은 데이터에 적용하여 소요되는 시간이 얼마나 감소하였는지 실험하는 것이다. 둘째, 제안된 기법의 알고리즘 설계와 고려하지 못한 부분들을 확인하여, 사용자들이 원하는 최적화된 스카이라인의 기법을 개선하는 것이다.

참고문헌

[1] Groz, B. and Milo, T. (2015), “Skyline queries with noisy comparisons,” in Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, ACM, pp. 185-198.

[2] Borzsony, S., Kossmann, D., and Stocker, K. (2001). "The skyline operator." In Proceedings of the 17th international Conference on Data Engineering, IEEE, pp. 421-430.

[3] Chomicki, J., Godfrey, P., Gryz, J. and Liang, D. (2005), “Skyline with presorting: Theory and optimizations,” Intelligent Information Processing and Web Mining, Vol. 31, pp. 595-604.

[4] Chomicki, J., Godfrey, P., Gryz, J., & Liang, D. (2003, March). Skyline with presorting. In ICDE (Vol. 3, pp. 717-719).

[5] Park, Y., Min, J. K., & Shim, K. (2013). "Parallel computation of skyline and reverse skyline queries using mapreduce." Proceedings of the VLDB ndowment, Vol 6(No 14), (pp. 2002-2013).

[6] Bøgh, K. S., Assent, I., & Magnani, M. (2013, June). "Efficient GPU-based skyline computation." In Proceedings of the Ninth International Workshop on Data Management on New Hardware. ACM. pp.5:1-6.