정보검색시스템에서 조인 시퀀스 분리성 기반 논리곱 불리언 질의 최적화

(Conjunctive Boolean Query Optimization based on Join Sequence Separability in Information Retrieval Systems)

박 병 권 † 한 욱 신 †† 황 규 영 †††
(Byung-Kwon Park) (Wook-Shin Han) (Kyu-Young Whang)

요 약 논리곱 불리언 질의는 질의에 포함된 키워드들이 모두 나타나는 텍스트 문서들을 검색하는질의로서, 정보검색 시스템에서 가장 널리 사용되는 질의이다. 논리곱 불리언 질의는 검색의 정확도를 높이기 위하여 많은 수의 키워드로 구성된 긴 질의를 사용한다. 이 경우, 키워드 처리 순서가 성능에 크게 영향을 미친다. 기존 정보검색시스템에서는 휴리스틱에 의존하여 키워드 처리 순서를 결정하므로 최적을 보장하지 못한다. 동적 프로그래밍과 같은 기존의 데이타베이스 질의 최적화 알고리즘은 복잡도가 지수적으로 증가하므로($O(n2^{n-1})$), 키워드 수가 많은 논리곱 불리언 질의에는 적합하지 않다. 본 논문에서는 조인 시퀀스 분리성이라는 새로운 개념에 기반한 논리곱 불리언 질의 최적화 알고리즘을 제안한다. 조인 시퀀스 분리성이란 조인에 참여하는 릴레이션들이 어떤 조건을 만족할 경우, 최적 조인 시퀀스가 두 개의 서브 시퀀스로 분리된다는 성질이다. 이 성질을 활용하면 $O(n\log n)$만에 최적 조인 시퀀스를 구할 수 있다. 본 논문에서는 이러한 조인 시퀀스 분리성의 개념을 정형적으로 정의하고 이에 기반한 질의 최적화 알고리즘의 최적성을 이론적으로 증명한다. 그리고, 제안한 질의 최적화 알고리즘의 성능 평가를 위해, 비용 모델을 사용하여 다양한 시뮬레이션을 수행한다. 그 결과, 제안한 알고리즘의 성능이 기존의 휴리스틱 기반 질의 최적화 알고리즘들에 비해 100배 이상 우수함을 보인다. 또한, 동적 프로그래밍 알고리즘에 비해 질의 최적화 시간 면에서 기하 급수적으로 우수함을 보인다(키워드 개수가 10 개일 경우 600배 이상 우수함).

키워드 : 정보검색, 논리곱 불리언 질의, 질의 최적화, 조인 시퀀스 분리성

Abstract A conjunctive Boolean text query refers to a query that searches for text documents containing all of the specified keywords, and is the most frequently used query form in information retrieval systems. Typically, the query specifies a long list of keywords for better precision, and in this case, the order of keyword processing has a significant impact on the query speed. Currently known approaches to this ordering are based on heuristics and, therefore, cannot guarantee an optimal ordering. We can use a systematic approach by leveraging a database query processing algorithm like the dynamic programming, but it is not suitable for a text query with a typically long list of keywords because of the algorithm's exponential run-time ($O(n2^{n-1})$ for n keywords). Considering these problems, we propose a new approach based on a property called the join sequence separability. This property states that the optimal join sequence is separable into two subsequences of different join methods under a certain condition on the joined relations, and this property enables us to find a globally optimal join sequence in $O(n2^{n-1})$. In this paper we describe the property formally, present an optimization algorithm based on the property, prove that the algorithm finds an optimal join sequence, and validate our approach through simulation using an analytic cost model. Comparison with the heuristic text query optimization approaches shows a maximum of 100 times faster query processing, and comparison with the dynamic programming approach shows exponentially faster query optimization (e.g., 600 times for a 10-keyword query).

Key words : Information Retrieval(IR), Conjunctive Boolean Query, Query Optimization, Join Sequence Separability

· 본 연구는 첨단정보기술연구센터를 통하여 한국과학재단의 지원을 받았음
† 비 회 원 : 동아대학교 경영정보과학부 교수
bpark@daunet.donga.ac.kr
†† 종신회원 : 경북대학교 컴퓨터공학과 교수
wshan@knu.ac.kr
††† 종신회원 : 한국과학기술원 전자전산학과 교수
kywhang@mozart.kaist.ac.kr
논문접수 : 2003년 7월 30일
심사완료 : 2004년 4월 8일

1. 서 론

정보검색을 위한 질의 형태 중에서 불리언 질의는 대부분의 정보검색시스템이 지원하는 기본적인 질의 형태이다[1,2]. 논리곱 불리언 질의는 모든 키워드들이 AND로만 연결된 질의로서, 질의에 포함된 키워드들이 모두 나타나는 문서들을 검색하라는 것이다. 논리곱 불리언 질의에서 질의 결과의 정밀도(precision)를 높이기 위해 때로는 수십개 이상의 많은 키워드로 구성된 긴 질의(long query)가 사용된다[3]. 이 경우, 질의에 포함된 키워드들을 어떠한 순서로 처리하느냐가 질의 처리 성능에 큰 영향을 미친다.

논리곱 불리언 질의를 처리하기 위해서는 질의에 포함된 각 키워드가 어느 문서에 나타나는 지를 알려주는 색인이 필요하다. 널리 알려진 색인으로는 역색인(inverted index)과 시그너춰 색인(signature index)이 있다. 역색인은 시그너춰 색인에 비하여 대규모 문서 데이타베이스에 대한 성능이 우수하고 널리 사용되므로[4], 본 논문에서는 역색인을 이용한 논리곱 불리언 질의 처리를 다룬다.

질의 처리 성능을 높이기 위해서는 질의 최적화를 수행하여야 한다. 정보검색 분야에서는 논리곱 불리언 질의에 대하여 휴리스틱에 의존한 질의 최적화 방법을 사용하여 왔다[2]. 이러한 기존의 논리곱 불리언 질의 처리 방법은 최적을 보장할 수 없다는 문제를 가진다.

본 논문에서는 논리곱 불리언 질의 처리가 데이타베이스 질의 처리에 대응될 수 있음을 보인다. 그러면, 기존 데이타베이스 질의 최적화 방법을 논리곱 불리언 질의 최적화에 적용할 수 있다. 그러나, 기존 데이타베이스 질의 최적화 알고리즘으로 널리 사용되는 시스템 R 스타일의 동적 프로그래밍은 최적 질의 계획을 생성하지만 복잡도가 $O(n2^{n-1})$이므로[5] 키워드 개수 n의 증가에 따라 복잡도가 기하 급수적으로 증가하는 문제점을 가진다. 따라서, 키워드 개수가 많은 논리곱 불리언 질의 최적화에는 사용하기가 어렵다.

본 논문에서는 조인 시퀀스 분리성이라는 새로운 개념과 이에 기반한 질의 최적화 알고리즘을 제시한다. **조인 시퀀스 분리성**이란, 조인에 참여하는 릴레이션들이 조인키 정렬 릴레이션이고 조인 애트리뷰트가 같은 경우, 최적 조인 시퀀스가 조인 연산이 서로 다른 두 개의 서브 시퀀스로 분리되는 성질이다. 이 성질을 이용하면 분할-정복 접근(divide-and-conquer approach)이 가능하므로 최적 조인 시퀀스를 매우 빠르게 찾을 수 있다. 여기서, **조인키 정렬 릴레이션**이란 조인 애트리뷰트가 키이고 키에 의해 정렬되어 있는 릴레이션이다. 조인 시퀀스 분리성을 활용한 질의 최적화 알고리즘은 $O(n2^{n-1})$의 복잡도를 가지며, 조인 시퀀스 분리성에 따라 나누어진 두 개의 서브 시퀀스를 찾는 알고리즘과 서브 시퀀스 내의 최적 조인 순서를 찾는 알고리즘으로 구성된다.

분리성을 이용한 문제 해결 방법은 Whang[6] 등의 연구에서도 찾아볼 수 있다. Whang 등은 어떤 조건하에서 특정 조인 방법들이 분리성을 가짐을 보이고, 이를 이용하여 복잡한 물리적 데이타베이스 설계 문제를 간단한 몇 개의 개별 릴레이션 설계 문제로 바꿀 수 있음을 보였다. 그러나, 분리성을 이용하여 복잡한 문제를 간단하게 바꾸어 해결한 점에서는 동일하지만 분리성의 정의와 주체는 서로 다르다.

본 논문에서는 조인 시퀀스 분리성을 정형적으로 정의하고 이에 기반하여 제안한 질의 최적화 알고리즘의 최적성을 이론적으로 증명한다. 그리고, 다양한 실험을 통해 제안한 알고리즘이 기존의 휴리스틱보다 훨씬 우수함을 보이고, 동적 프로그래밍 알고리즘보다는 질의 최적화 시간 면에서 성능이 훨씬 우수함을 보인다.

본 논문의 구성은 다음과 같다. 제2장에서는 역색인 구조와 이를 이용한 기존의 논리곱 불리언 질의 처리 방법 및 그 문제점을 설명한다. 제3장에서는 질의 최적화의 이론적 기반이 되는 조인 시퀀스 분리성의 개념을 정의한다. 제4장에서는 조인 시퀀스 분리성에 기반한 질의 최적화 알고리즘을 제안하고 이의 최적성을 이론적으로 증명한다. 제5장에서는 다양한 실험을 통하여 제안한 질의 최적화 방법의 성능 평가 결과를 제시한다. 마지막으로 제6장에서는 결론을 맺는다.

2. 연구 배경

본 장에서는 논리곱 불리언 질의 처리 관련 연구 배경을 기술한다. 제2.1절에서는 논리곱 불리언 질의 처리에 이용되는 역색인의 구조에 대하여 살펴 본다. 제2.2절에서는 역색인을 이용한 논리곱 불리언 질의 처리가 하나의 데이타베이스 질의 처리에 대응될 수 있음을 보인다. 제2.3절에서는 기존 논리곱 불리언 질의 처리 방법의 문제점을 기술한다.

2.1 역색인 구조

역색인은 일반적으로 키워드 목록(keyword directory)과 포스팅 파일(postings file)로 구성된다[2,7,8]. 키워드 목록은 문서 데이타베이스에 나타난 모든 키워드들의 목록이고, 포스팅 파일은 포스팅 리스트들이 저장되어 있는 파일이다. 키워드 목록에는 키워드와 함께 그 키워드의 문서 빈도수(document frequency: df)와 포스팅 리스트(postings list)에 대한 포인터를 가지고 있다. 여기서, 문서 빈도수란 그 키워드가 나타난 문서의 개수를 말한다. 포스팅 리스트의 한 원소인 포스팅은 그 키워드가 어느 문서의 어느 위치에 나타났는지에 대

한 정보로서, <문서 식별자, 키워드 빈도수, 발생 위치 집합>으로 구성된다. 여기서, 키워드 빈도수는 그 문서 내에서 키워드 발생 회수를 나타낸다. 일반적으로, 포스팅 리스트의 포스팅들은 문서 식별자 순으로 정렬되어 저장된다고 가정하므로[2,7,8,9], 본 논문에서도 이를 따른다.

예 1. 그림 1은 역색인의 예를 나타내고 있다. 그림의 왼쪽 부분은 키워드 목록을 나타내고, 오른쪽은 포스팅 파일을 나타낸다. 그림의 키워드 목록에서 키워드 *multimedia*는 문서빈도수가 3이므로 가리키는 포스팅 리스트에는 세 개의 포스팅이 있다. 첫 번째 포스팅은 키워드 *multimedia*가 문서 식별자가 100인 문서에서 두 번 나타남을, 두 번째 포스팅은 문서 식별자가 200인 문서에서 한 번 나타남을, 세 번째 포스팅은 문서 식별자가 300인 문서에서 세 번 나타남을 각각 나타낸다. □

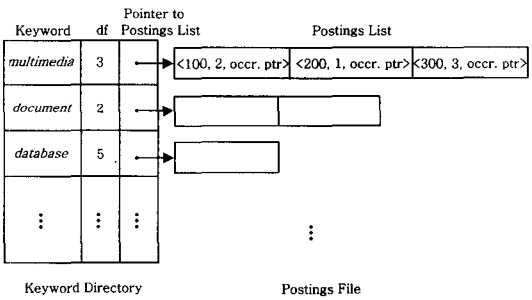


그림 1 역색인의 예

2.2 논리곱 불리언 질의 처리와 데이타베이스 질의 처리의 대응성

역색인을 이용하여 논리곱 불리언 질의를 처리할 때 질의에 포함된 각 키워드의 포스팅 리스트에 포함된 문서식별자들의 교집합을 구하기 위하여 기존의 정보검색 시스템에서는 다음과 같은 휴리스틱을 이용한다[2,8]. 첫째, 문서 빈도수가 가장 낮은 키워드의 포스팅 리스트에 포함된 문서식별자들을 후보 문서식별자 집합으로 한다. 그 이유는 후보 집합의 크기를 가장 작게 유지할 수 있기 때문이다. 둘째, 문서 빈도수가 그 다음으로 낮은 키워드의 포스팅 리스트에 포함된 문서식별자들과 후보 집합 간의 교집합을 구하고 이를 새로운 후보 집합으로 한다. 세째, 위의 둘째 과정을 반복해 나가면 마지막으로 얻어지는 후보 집합이 질의 결과이다.

위의 휴리스틱에서 후보 집합과 포스팅 리스트 간의 교집합을 구할 때 병합(merge) 또는 탐색(search)의 한 가지 방법만 사용된다[2]. 병합은 후보 집합과 포스팅 리스트를 각각 처음부터 차례대로 읽어 나가면서 양쪽에 공통된 문서 식별자들만 구해 나가는 방법이고, 탐색은 후보 집합의 각 문서 식별자가 포스팅 리스트에 포함되어 있는 지를 탐색하는 방법이다. 이 때, 포스팅 리스트가 큰 경우 빠른 탐색을 위하여 포스팅 리스트에 별도의 색인으로서 B-tree 를 구축할 수 있다[10].

논리곱 불리언 질의 처리를 데이타베이스 질의 처리와 비교해 보면 다음과 같은 대응 관계를 발견할 수 있다. 첫째, 하나의 포스팅 리스트를 릴레이션(relation)으로 간주할 수 있다. 그러면, 포스팅은 튜플(tuple)이 되고, 문서식별자는 키가 된다. 둘째, 논리곱 불리언 연산을 자연조인(natural join) 연산으로 간주할 수 있다. 여기서, 조인 애트리뷰트는 문서식별자이다. 그러면, n개의 포스팅 리스트에 대한 교집합 순서는 n개의 릴레이션에 대한 조인 시퀀스(join sequence)에 해당한다.

또한, 후보 집합과 포스팅 리스트 간의 교집합을 구할 때, 병합 방법은 병합 조인(merge join) 연산에 해당하고, 탐색 방법은 중포-루프 조인(nested-loop join) 연산에 해당한다. 여기서, 병합 조인은 조인에 참여하는 릴레이션들이 이미 조인 애트리뷰트에 의해 정렬되어 있으므로 별도의 정렬 과정이 필요 없다. 중포-루프 조인은 내부 릴레이션(inner relation)이 조인 애트리뷰트에 대해 색인이 구축되어 있을 경우, 색인을 사용하는 중포-루프 조인(indexed nested-loop join)이 된다. 본 논문에서는 색인을 사용하지 않는 중포-루프 조인과 색인을 사용하는 중포-루프 조인을 총칭하여 중포-루프 조인이라고 부른다.[1)]

논리곱 불리언 질의 처리는 일반적인 데이타베이스 질의 처리와는 다른 특수한 성질을 가진다. 첫째, 조인 애트리뷰트가 모두 키이다. 따라서, 모든 조인은 1:1 조인이므로 조인 결과 릴레이션의 카디날리티는 조인에 참여한 각 릴레이션의 카디날리티보다 작다. 즉, 조인을 여러 번 할수록 조인 결과 수는 점점 작아진다. 둘째, 조인에 참여하는 모든 릴레이션은 조인 애트리뷰트에 의해 정렬되어 있다. 따라서, 조인 결과 릴레이션도 조인 애트리뷰트에 의한 정렬을 유지한다. 그러므로, 조인 결과 릴레이션을 저장하여 다시 정렬할 필요가 없다.

2.3 기존 질의 처리 방법의 문제점

이미 언급하였듯이, 기존의 논리곱 불리언 질의 처리 방법은 휴리스틱에 기반한 질의 최적화 방법을 사용한다. 이를 데이타베이스의 조인 질의 처리로 해석하면, 조인 순서는 조인에 참여하는 릴레이션들의 카디날리티들을 오름차 순으로 정렬한 순서를 사용하고, 조인 연산은 병합 조인이나 중포-루프 조인 중 하나를 임의로 선

1) 이 외에도 데이타베이스 질의 처리를 위한 조인 방법으로는 정렬 비용을 줄이는 해쉬 조인(hash join) 등의 조인 연산들이 사용되고 있다. 그러나, 조인에 참여하는 릴레이션들이 이미 정렬되어 있으므로 본 논문에서는 병합 조인과 중포-루프 조인만을 고려한다.

택하여 사용한다. 그러나, 이러한 질의 처리 방법은 최적을 보장할 수 없다는 문제를 가진다.

이러한 문제를 해결하는 방편으로, 기존 데이타베이스 질의 최적화 방법을 논리곱 불리언 질의 최적화에 적용할 수 있다. 대부분의 데이타베이스 질의 최적화 알고리즘은 그림 2와 같은 왼쪽-깊은 조인 트리(left-deep join tree)에서의 질의 최적화를 다룬다[5,11,12]. 왼쪽-깊은 조인 트리에서 릴레이션이 n개 있으면 최대 $n!$ 개의 조인 순서가 존재하지만, 시스템 R 스타일의 동적 프로그래밍 알고리즘을 사용하면 $O(n2^{n-1})$만에 최적 조인 순서를 구할 수 있다[5]. 그러나, 이 동적 프로그래밍 알고리즘은 조인되는 릴레이션의 수가 많아지면 최적화 시간이 지수적으로 증가하므로 많은 키워드를 가진 논리곱 불리언 질의 최적화에는 사용하기가 어렵다. 본 논문에서는 일반적인 데이타베이스 질의 처리와는 다른, 논리곱 불리언 질의 처리의 특수한 성질을활용하여 조인 시퀀스 분리성 개념을 제안하고, 이를 이용하여 $O(n\log n)$만에 최적 조인 순서를 구할 수 있음을 보인다.

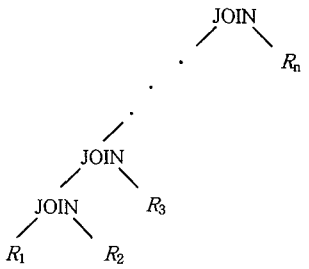


그림 2 n 개의 릴레이션에 대한 왼쪽-깊은 조인 트리의 예

3. 조인 시퀀스 분리성

본 장에서는 조인 시퀀스 분리성의 개념을 제안한다. 제3.1절에서는 필요한 용어들을 정의하고, 제3.2절에서는 조인 시퀀스 분리성의 개념을 정형적으로 정의한다.

3.1 용어 정의

정의 1. 조인에 참여하는 어떤 릴레이션의 조인 애트리뷰트가 그 릴레이션의 키일 경우, 그 애트리뷰트를 **조인키 애트리뷰트**라 정의한다.

정의 2. 조인에 참여하는 각 릴레이션이 튜플 크기가 같고, 조인 애트리뷰트가 같으며, 조인키 애트리뷰트에 의해 정렬되어 있을 경우, 이러한 릴레이션을 **조인키 정렬 릴레이션**이라 정의한다.

정의 3. n개의 릴레이션 $R_1, R_2, ..., R_n$ 을 왼쪽-깊은 조인 트리 형태의 순서로 조인할 때, R_1부터 R_{i-1}까지의 조인 결과와 R_i의 조인 중 R_i에 국한된 연산 -- 즉, 상대방 릴레이션에 대한 연산은 배제함 - 을 **R_i에 대한 조인**이라 정의한다. 단, 첫번째 릴레이션 R_1에 대한 조인은 R_1으로 정의한다.

정의 4. n개의 릴레이션에 대한 조인 시퀀스 S가 주어졌을 때, S[i]는 i번째의 릴레이션을 나타내고, S[i].OP는 S[i]에 대한 조인 연산을 나타내며, S[i].OP를 수행하기 위한 비용을 **S[i]에 대한 조인 비용**이라 부르며 JC(S,i)로 표시한다.

본 논문에서는 대부분의 데이타베이스 질의 최적화에서 가정하듯이[12] 조인 비용으로 디스크 입출력 회수를 사용한다.

정의 5. n개의 릴레이션에 대한 조인 시퀀스 S가 주어졌을 때, S에 따라 n개의 릴레이션을 조인하기 위한 비용을 **S에 대한 조인 비용**이라 부르며 C(S)로 표기한다.

C(S)는 S 내의 각 릴레이션에 대한 조인 비용의 합과 같으며, 수식 (1)과 같이 표현할 수 있다.

$$C(S) = \sum_{i=1}^{n} JC(S,i) \quad (1)$$

정의 6. n개의 릴레이션에 대한 조인 시퀀스 S가 주어졌을 때, S[i]를 임의의 j 번째로 이동한 조인 시퀀스를 S′이라고 하자.[2] 이 때, JC(S,i)와 JC(S′,j)가 항상 같을 경우, S[i].OP를 **비용 불변 조인 연산**(*cost-invariant join operation*)이라 정의한다.

보조정리 1. n개의 조인키 정렬 릴레이션에 대한 조인에서 어떤 릴레이션 R에 대한 병합 조인 연산 시 항상 R을 모두 스캔한다고 가정하면 병합 조인 연산은 비용 불변 조인 연산이다.

증명: n개의 조인키 정렬 릴레이션에 대한 조인에서는 조인 결과 릴레이션도 조인 애트리뷰트에 의한 정렬을 유지하므로 R에 대한 병합 조인 비용은 R을 순차 스캔하는 비용이므로 조인 시퀀스 상의 위치에 상관없이 항상 일정하다. □

정의 7. n개의 릴레이션에 대한 조인 시퀀스 S가 주어졌을 때, S[i]를 i 보다 큰 임의의 j 번째로 이동한 조인 시퀀스를 S′이라고 하자. 이 때, JC(S,i) 보다 JC(S′,j)가 같거나 작을 경우, S[i].OP를 **비용 감소 조인 연산**(*cost-decreasing join operation*)이라 정의한다.

보조정리 2. n개의 조인키 정렬 릴레이션에 대한 조인에서 어떤 릴레이션 R에 대한 중포-루프 조인 연산은 비용 감소 조인 연산이다.

증명: 주어진 조인 시퀀스 상에서 어떤 릴레이션 R에 대한 중포-루프 조인 시에, 현재까지의 중간 결과는 외부 릴레이션(outer relation) 역할을 하고 R은 내부

2) 시퀀스 상에서 i 번째 엔트리를 j 번째로 이동한다는 것은 i 번째에서 삭제하여 j 번째에 삽입하는 것을 의미한다.

릴레이션(inner relation) 역할을 한다. 따라서, 중간 결과 수가 작을 수록 R에 대한 중포-루프 조인 비용은 작아진다. 그런데, n개의 조인키 정렬 릴레이션에 대한 조인에서는 모든 조인 애트리뷰트가 키이므로 조인을 하면 할 수록 중간 결과 수가 같거나 작아지므로 R에 대한 중포-루프 조인 비용은 조인 시퀀스 상에서 그 위치가 나중일 수록 같거나 작아진다. □

3.2 조인 시퀀스 분리성의 개념

조인 연산에 비용 불변 조인 연산과 비용 감소 조인 연산들이 섞여 있을 경우, 비용 불변 조인 연산부터 먼저 수행하고 비용 감소 조인 연산을 나중에 수행하는 것이 유리하다. 왜냐하면, 비용 불변 조인 연산은 조인 순서를 바꾸어도 조인 비용이 변하지 않지만, 비용 감소 조인 연산은 나중에 할수록 조인 비용이 감소하기 때문이다. 이에 대한 정형적인 설명은 정리 1과 같다.

정리 1. 비용 불변 조인 연산과 비용 감소 조인 연산만을 가진 n개의 릴레이션에 대한 조인이 주어졌을 때, 모든 비용 불변 조인 연산들이 모든 비용 감소 조인 연산들보다 먼저 나타나는 최적 조인 시퀀스가 항상 존재한다.

증명: 귀류법(proof by contradiction)을 사용하여 증명한다. 먼저, 그러한 최적 조인 시퀀스가 존재하지 않는다고 가정하자. 즉, 모든 최적 조인 시퀀스에는 항상 어떤 비용 불변 조인 연산보다 먼저 나타나는 비용 감소 조인 연산이 적어도 하나 존재한다고 가정하자. 그러면, 그러한 최적 조인 시퀀스를 S라 할 때, S[i].OP가 비용 감소 조인 연산이고 S[i+1].OP가 비용 불변 조인 연산이 되는 i가 적어도 하나 존재한다. 이 때, S[i]와 S[i+1]을 서로 맞바꾼 조인 시퀀스를 S′이라고 하면, S[i+1].OP와 S′[i].OP가 비용 불변 조인 연산이므로 JC(S,i+1) = JC(S′,i)이 성립하고, S[i].OP와 S′[i+1].OP가 비용 감소 조인 연산이므로 JC(S,i) (JC(S′,i+1)이 성립한다. 따라서, C(S)와 C(S′)의 차이를 구해 보면 다음과 같다.

$$C(S)-C(S') = JC(S,i)+JC(S,i+1)-JC(S',i)-JC(S',i+1)$$
$$= JC(S,i)-JC(S',i+1) \geq 0$$

C(S) - C(S′) > 0일 경우 S가 최적이라는 가정에 위배된다. C(S) - C(S′) = 0일 경우 S′도 최적이 된다. S′에 아직 비용 불변 조인 연산보다 먼저 나타나는 비용 감소 조인 연산이 존재하면 위의 과정을 반복한다. 그러면 결국 모든 비용 불변 조인 연산들이 모든 비용 감소 조인 연산들보다 먼저 나타나는 최적 조인 시퀀스를 얻게 되므로, 그러한 최적 조인 시퀀스가 존재하지 않는다는 가정에 위배된다. □

정의 8. 어떤 조인 시퀀스 S가 주어졌을 때, 모든 비용 불변 조인 연산들이 모든 비용 감소 조인 연산들보다 먼저 나타나면 S는 **조인 시퀀스 분리성**(*join sequence separability*)을 가진다라고 정의한다.

따름정리 1. 병합 조인 연산과 중포-루프 조인 연산만을 가진 n개의 조인키 정렬 릴레이션에 대한 최적 조인 시퀀스는 조인 시퀀스 분리성을 가진다.

증명: 보조정리 1에 의하여 n개의 조인키 정렬 릴레이션에 대한 병합 조인 연산은 비용 불변 조인 연산이고, 중포-루프 조인은 비용 감소 조인 연산이다. 따라서, 정리 1과 정의 8에 의하여 증명됨. □

4. 조인 시퀀스 분리성에 기반한 질의 최적화 알고리즘

본 장에서는 조인 시퀀스 분리성에 기반하여 n개의 조인키 정렬 릴레이션에 대한 질의 최적화 알고리즘 OptJoinSeq 를 제안한다. 병합 조인과 중포-루프 조인 연산 만을 가진 조인키 정렬 릴레이션에 대한 최적 조인 시퀀스는 따름정리 1에 의하여 조인 시퀀스 분리성을 가지므로 병합 조인 연산을 하는 릴레이션들의 서브 시퀀스(이후 병합 조인 서브 시퀀스라 하고 S_I로 표기함)와 중포-루프 조인 연산을 하는 릴레이션들의 서브 시퀀스(이후 중포-루프 조인 서브 시퀀스라 하고 S_D로 표기함)로 분리된다.

알고리즘 OptJoinSeq 는 최적 조인 시퀀스의 S_I와 S_D를 구해 주는 알고리즘으로서, 그림 3과 같이 두 단계로 나누어 수행된다. 1) 질의에 주어진 모든 릴레이션들이 중포-루프 조인 연산만 한다고 가정하고, S_D에 모두 포함시킨 후 S_D 내에서의 최적 조인 순서를 결정하는 알고리즘 OptDecSeq를 호출한다. OptDecSeq는 중포-루프 조인 연산만 있다고 가정한다. 2) 그리디 정책을 사용하여 S_D에 속한 릴레이션들을 하나씩 S_I로 보내어 최적의 S_I와 S_D를 결정하는 알고리즘 OptSeparation을 호출한다. 제4.1절에서는 알고리즘 기술에 필요한 기호들을 정의한다. 제4.2절에서는 알고리즘 OptDecSeq에 대하여 설명하고 제4.3절에서는 알고리즘 OptSeparation에 대하여 설명한다.

4.1 기호 정의

질의 최적화 알고리즘 기술에 필요한 기호들을 표 1

Algorithm *OptJoinSeq* (*S*)
/* *S*: input join sequence */
1. *S′* := *OptDecSeq*(*S*)
2. ($S_I S_D$) := *OptSeparation*(*S′*)
3. **return** ($S_I S_D$)

그림 3 조인 시퀀스 분리성에 기반한 질의 최적화 알고리즘 OptJoinSeq

과 같이 정의한다. 표 1에서 CARD(R)은 릴레이션 R의 튜플 개수를 나타내며, RS(R)은 릴레이션 R의 전체 튜플들을 순차적으로 읽기 위한 비용을 나타낸다. 따라서, RS(R)∝CARD(R)이 성립한다. RA(R)은 릴레이션 R에서 주어진 조건을 만족하는 특정 튜플을 찾기 위한 비용을 나타낸다. 이 때, R에 색인이 구축되어 있으면 RA(R)은 색인을 이용하는 비용이 되고, 없으면 평균 순차검색 비용 RS(R) / 2이 된다. 본 논문에서는 R의 카디날리티가 정해진 임계값 t를 넘어서면(CARD(R) > t), 탐색을 빠르게 하기 위해 색인을 구축한다고 가정한다. 이 때, 색인을 이용하는 비용이 평균 순차검색 비용보다 작아지도록(RA(R) < RS(R) / 2), t 값을 설정한다. 또한, 색인 구조는 트리 구조를 사용하여 CARD(R)의 증가에 따라 RA(R)이 로그 스케일로 증가한다고 (RA(R)∝log(CARD(R))) 가정한다. N은 전체 데이타베이스에 나타나는 조인 애트리뷰트의 서로 다른 값의 총 개수를 나타낸다.

표 1 기호 정의

기호	정의/의미
CARD(R)	릴레이션 R의 카디날리티 즉, 튜플 개수
RS(R)	릴레이션 R의 전체 튜플들을 순차 스캔하는 비용
RA(R)	릴레이션 R에서 특정 튜플을 찾는 비용
t	릴레이션 R에 색인을 구축하기 위한 CARD(R)의 임계값(threshold)
N	전체 데이터베이스에 나타나는 조인 애트리뷰트의 서로 다른 값의 총 개수

4.2 SD 내에서의 최적 조인 순서를 결정하는 알고리즘

S_D 내에서의 최적 조인 순서를 구하는 문제는 S_D에 대한 조인 비용 $C(S_D)$를 최소로 하는 S_D를 구하는 것이다. 본 절에서는 이를 위하여, 조인 시퀀스에 대한 선택률 인자를 먼저 정의하고 이를 이용하여 $C(S_D)$에 대한 비용 모델을 수립한 후 이를 최소로 하는 S_D를 구하는 알고리즘 OptDecSeq를 제안하고 이의 최적성을 증명한다.

정의 9. 어떤 조인 시퀀스 S에 대한 **선택률 인자**를 F(S)로 표기하고 다음과 같이 정의한다.

$F(\wedge) = 1$ for null sequence $\wedge$

$F(R) = \frac{CARD(R)}{N}$ for single relation R

$F(S_1S_2) = F(S_1) \times F(S_2)$ for $S \equiv S_1S_2$

그러면, 조인 시퀀스 S의 조인 결과 수는 $N \times F(S)$로 추정할 수 있다.

보조정리 3 두 개의 조인키 정렬 릴레이션 R와 T가 있을 때 다음이 성립한다.

$$F(R) < F(T) \Rightarrow \frac{F(T)}{F(R)} \geq \frac{RA(T)}{RA(R)}$$

증명: Appendix A 참조 □

S_D가 $S_D = R_1R_2...R_m$으로 주어져 있고, S_I에 대한 조인 결과 수가 N_1로 주어져 있을 때(S_I가 NULL이면 N_I는 N이 됨), S_D에 대한 조인 비용 $C(S_D)$는 수식 (2)와 같이 추정할 수 있다.

$$C(S_D) = \sum_{i=1}^{m} JC(S_D, i) = N_I \sum_{i=1}^{m} (\prod_{j=1}^{i} F(R_{j-1}))RA(R_i)$$

where R_0 is null (2)

Monma[13] 등은 주어진 비용 함수의 값을 최소로 하는 순서화 문제에서 비용 함수가 API(*Adjacent Pair-wise Interchange*) 성질을 가지고 있으면 $O(m\log m)$만에 최적 순서를 구할 수 있음을 밝혔다. 따라서, 수식 (2) 의 $C(S_D)$가 API 성질을 가지고 있음을 보이면, $O(m\log m)$만에 S_D 내에서의 최적 조인 순서를 구할 수 있음을 알 수 있다.

정의 10. [13] 순서화 문제에서 임의의 두 항목 i, j가 주어졌을 때, 어떠한 서브 시퀀스 U, V에 대해서도 비용 함수 C가 수식 (3)을 만족하면 C는 API 성질을 가진다라고 한다.

$$r(i) \geq r(j) \Rightarrow C(UijV) \leq C(UjiV) \quad (3)$$

여기서, r은 순서화 하고자 하는 항목에 대해 정의된 순위함수이다.

본 논문에서는 수식 (2)의 $C(S_D)$가 API 성질을 가짐을 보이기 위해 다음과 같은 순위함수를 정의한다.

정의 11. 조인키 정렬 릴레이션 R에 대한 순위함수 $r(R)$을 다음과 같이 정의한다.

$$r(R) = \frac{1 - F(R)}{RA(R)}$$

순위함수 $r(R)$의 값은 F(R)과 RA(R)의 값이 작을수록 커진다. 여기서, F(R)의 값이 작으면 R과의 조인 결과 수가 작아지고, RA(R)값이 작으면 R과의 중포-루프 조인 비용이 작아진다. 따라서, S_D 내에서는 r 값이 큰 R부터 먼저 조인하는 것이 유리하다. 즉, r 값은 S_D 내에서 최적 조인 순서를 정하는 좋은 기준이 될 수 있다.

보조정리 4 수식 (2) 의 $C(S_D)$는 API 성질을 가진다.

증명: Appendix B 참조. □

정리 2. S_D 내에서의 최적 조인 순서는 순위함수 r 값이 큰 릴레이션부터 먼저 조인하는 것이다.

증명: 보조정리 4에 의하여 S_D 내에서의 조인 비용 함수가 API 성질을 가지므로, Monma[13]에 의거하여 어떤 인접한 두 릴레이션의 r 값이 내림차 순이 아닐 경우, 이 둘의 순서를 교환하면 조인 비용이 더 작은 순

서를 얻을 수 있다. 따라서, r 값이 큰 것부터 먼저 조인하는 것이 최적 조인 순서이다. □

그림 4는 정리 2에 따라 S_D 내에서의 최적 조인 순서를 구해 주는 알고리즘 OptDecSeq 을 보여 주고 있다. 알고리즘 OptDecSeq는 m 개의 릴레이션들을 순위함수 r 값 순으로 정렬하여야 하므로 $O(m\log m)$의 복잡도를 가진다.

Algorithm *OptDecSeq(S)*
/* Find the opt. nested-loop join seq. of S*/
1. Calculate the rank $r(R)$ for each relation R in S
2.Sort S by r
3.Return S

그림 4 S_D 내에서의 최적 조인 순서를 구해 주는 알고리즘 OptDecSeq

순위함수와 관련한 기존 연구로는 Ibaraki[14]가 색인이 구축되어 있지 않고 정렬되지 않은 n 개의 릴레이션에 대한 중포-루프 조인을 처리하기 위해 순위 개념을 이용하였고, Hellerstein[15,16]과 Chaudhuri[5]는 사용자 정의 함수를 가진 질의를 처리할 때 순위 개념을 이용하였다. 그러나, 조인키 정렬 릴레이션의 조인에 대해서는 아직 연구된 바가 없다.

4.3 최적의 S_I 와 S_D 를 결정하는 알고리즘

n개의 릴레이션을 S_I와 S_D로 분리할 수 있는 경우의 수는 총 2^n개가 존재한다. 따라서, 전역 탐색(exhaustive search)을 통해 최적의 S_I와 S_D를 구하려면 탐색 시간이 n 값에 따라 지수적으로 증가한다. 본 절에서는 $O(n\log n)$만에 최적의 S_I와 S_D를 결정할 수 있는 알고리즘 OptSeparation을 제안하고 이의 최적성을 증명한다.

S_I와 S_D가 각각 $S_I = R_1R_2...R_k$, $S_D = R_1R_2...R_m$으로 주어져 있을 때, S_I S_D에 대한 조인 비용 $C(S_I\ S_D)$는 수식 (4)와 같이 추정할 수 있다.

$$C(S_IS_D) = C(S_I) + C(S_D)$$

$$C(S_I) = \sum_{i=1}^{k} RS(R_i),\quad C(S_D) = N \times F(S_I) \times NLC(S_D)$$

$$NLC(S_D) = \sum_{i=1}^{m} (\prod_{j=1}^{i} F(R_{j-1}))RA(R_i) \qquad (4)$$

where R_0 is null

위의 수식에서 NLC(SD)는 S_I에 대한 조인 결과 수가 1일 때의 S_D에 대한 조인 비용을 의미한다.

정리 3. S = S_IS_D가 최적 조인 시퀀스이면, S_I에 속한 모든 릴레이션의 카디날리티는 S_D에 속한 모든 릴레이션의 카디날리티보다 작거나 같다.

증명: 귀류법을 사용하여 증명한다. S가 최적임에도 불구하고, S_I에 속한 어떤 릴레이션 R의 카디날리티가 S_D에 속한 어떤 릴레이션 T의 카디날리티보다 크다고 가정하자(즉, CARD(R) > CARD(T)). 그리고, 그림 5와 같이 S에서 T와 R의 위치를 서로 맞바꾼 조인 시퀀스를 S′이라고 하자.

$$S = \overbrace{S_{I1}\ T\ S_{I2}}^{S_I}\ \overbrace{S_{D1}\ R\ S_{D2}}^{S_D} \qquad S' = \overbrace{S_{I1}\ R\ S_{I2}}^{S'_I}\ \overbrace{S_{D1}\ T\ S_{D2}}^{S'_D}$$

그림 5 두 개의 조인 시퀀스, S와 S′

그러면, S와 S′에 대한 조인 비용 C(S)와 C(S′)을 구해 보면 다음과 같다.

$$C(S) = \sum_{U\in S_{I_1}} RS(U) + RS(T) + \sum_{U\in S_{I_2}} RS(U) + N \times F(S_{I_1})F(T)F(S_{I_2})(NLC(S_{D_1}) + F(S_{D_1})RA(R) + F(S_{D_1})F(R)NLC(S_{D_2}))$$

$$C(S') = \sum_{U\in S_{I_1}} RS(U) + RS(R) + \sum_{U\in S_{I_2}} RS(U) + N \times F(S_{I_1})F(R)F(S_{I_2})(NLC(S_{D_1}) + F(S_{D_1})RA(T) + F(S_{D_1})F(T)NLC(S_{D_2}))$$

따라서, C(S) - (C(S′)를 구해 보면 다음과 같다.

$$\begin{aligned} C(S) - C(S') &= RS(T) + N \times F(S_{I_1})F(T)F(S_{I_2})(NLC(S_{D_1}) + F(S_{D_1})RA(R)) - RS(R) \\ &\quad - N \times F(S_{I_1})F(R)F(S_{I_2})NLC(S_{D_1}) \\ &\quad + F(S_{D_1})RA(T)) \\ &= RS(T) - RS(R) + N \times F(S_{I_1})F(S_{I_2}) \\ &\quad (NLC(S_{D_1})(F(T) - F(R)) + \\ &\quad N \times F(S_{I_1})F(S_{I_2})F(S_{D_1})F(T)RA(R) \\ &\quad - F(R)RA(T) \end{aligned}$$

CARD(R) > CARD(T)이므로 RS(T) > RS(R)이고 F(T) > F(R)이다. 또한, 보조정리 3에 의해 $\frac{F(T)}{F(R)} \geq \frac{RA(T)}{RA(R)}$ 이 성립하므로 $F(T)RA(R) \geq F(R)RA(T)$ 이 성립한다. 따라서, C(S) > C(S′)이 성립하므로 S는 최적이 아님. □

그림 6은 주어진 조인 시퀀스 S를 최적의 S_I S_D로 분리해 주는 알고리즘 OptSeparation을 보여주고 있다. 입력으로 주어지는 S는 순위함수 값에 따라 릴레이션들이 정렬되어 있어야 한다. 처음에는 S 전체를 S_D로 두고 초기 비용을 계산한다. 다음에는, S_D 내에서 카디날리티가 가장 작은 릴레이션을 하나씩 선택하여 S_I로 옮긴다. 그러면, 총 n 번을 옮기게 되는데, 이들 중에서 가장 비용이 작은 S_IS_D를 선택하여 반환한다. 이 알고리즘은 카디날리티가 가장 작은 릴레이션을 차례로 선택하기 위하여 S 내의 릴레이션들을 카디날리티 순으로 한번 정렬하여야 하므로 $O(n\log n)$의 복잡도를 가진다.

정리 4. 알고리즘 OptSeparation 이 결정한 S_IS_D는 최적 조인 시퀀스이다.

증명. n 개의 조인키 정렬 릴레이션이 주어졌을 때,

```
Algorithm OptSeparation(S)
/* Separate S into the optimum S_I and S_D
   Note: S has been sorted by the rank in OptDecSeq. */
1.S_I := empty sequence /* initial value of S_I*/
2.   S_D := S /* initial value of S_D */
3.   Sort S in the increasing order of the relation cardinality and store the result in S-card
4.   Calculate the cost C(S) of the initial sequence
5.While moving relations from S_D to S_I one after another according to the order in S-card,
     calculate the cost C(S_I S_D) of the resulting sequence after each move
6.Choose the S_I S_D that has the minimum cost among the initial C(S) and those from Step 5
7.Return (S_I S_D)
```

그림 6 최적의 $S_I S_D$를 결정해 주는 알고리즘 OptSeparation

정리 2와 정리 3에 의해, 최적 조인 시퀀스는 다음 두 조건을 만족하는 n+1 개의 조인 시퀀스 중에 최소 비용을 가지는 것이다.

- S_I에는 n 개의 릴레이션들 중에서 카디날리티가 작은 순서대로 최소 0 개에서 최대 n 개까지 포함될 수 있다.
- S_D는 순위함수 값 순으로 정렬되어 있다.

알고리즘 OptSeparation은 순위함수 값 순으로 정렬된 조인 시퀀스를 입력으로 받아 이를 초기 S_D로 삼은 후, S_D에서 카디날리티가 작은 릴레이션부터 순서대로 하나씩 S_I로 옮긴다. 따라서, 위의 두 가지 조건을 만족하는 n+1 개의 조인 시퀀스들을 검사하게 된다. 그리고, 이 중에서 최소 비용을 가지는 조인 시퀀스를 선택하므로 최적 조인 시퀀스가 선택된다. □

5. 성능 평가

본 장에서는 논리곱 불리언 질의 최적화에 대하여 여러 가지 다양한 실험을 통해 제안하는 질의 최적화 알고리즘의 성능을 평가해 본다. 논리곱 불리언 질의의 경우, 포스팅 리스트가 하나의 릴레이션에 해당하므로, 키워드 w의 포스팅 리스트를 R(w)로 표기하면 R(w)는 문서식별자를 조인키로 하는 조인키 정렬 릴레이션이 된다. 따라서, 제4장에서 제안한 질의 최적화 알고리즘을 논리곱 불리언 질의 최적화에 적용할 수 있다. 그리고, CARD(R(w)) > t일 경우 w의 포스팅 리스트에 B-tree 색인을 구축한다. 본 장의 구성은 다음과 같다. 제5.1절에서는 실험 환경을 설명하고, 제5.2절에서는 논리곱 불리언 질의 처리 비용 모델을 설명한다. 제5.3절에서는 질의 최적화 알고리즘의 최적화 정도를 평가한다. 제5.4절에서는 질의 최적화 알고리즘의 최적화 시간을 평가한다.

5.1 실험 환경

논리곱 불리언 질의 최적화 실험에 사용된 통계치들은 표 2와 같다. 포스팅의 평균 크기 ρ는 평균 66 바이트로 하였다.[3] 문서식별자의 크기 α는 충분한 양의 문서를 식별할 수 있도록 4 바이트로 하였다. 포스팅 리스트에 B-tree 색인을 구축할 경우, B-tree 의 디스크 블럭 포인터의 크기 β는 4 바이트로, 디스크 블럭의 크기 γ는 4096 바이트로, 디스크 블럭 사용률 u는 평균 75%로 하였다. 논리곱 불리언 질의 최적화의 경우, 모든 조인 애트리뷰트가 문서식별자이므로 N은 전체 데이타베이스에 저장된 총 문서 수이다. 본 실험에서는 N = 10,000,000으로 하였다. 그리고, 모든 실험은 128MB의 메인 메모리를 가진 Pentium III 400MHz PC에서 Java 언어를 사용하여 시뮬레이션 하였다.

표 2 실험에 사용된 통계치

기호	정의/의미	값
ρ	포스팅의 평균 크기	66 bytes
α	문서식별자의 크기	4 bytes
β	디스크 블럭 포인터의 크기	4 bytes
λ	디스크 블럭의 크기	4096 bytes
u	디스크 블럭의 공간 사용률	0.75
N	전체 데이타베이스에 저장된 총 문서 수	10,000,000개

사용자 질의에 주어지는 키워드의 문서빈도수는 그 분포가 정해져 있지 않으므로 여러 가지 분포에 대한 실험이 필요하다. 그런데, 키워드 w의 문서 빈도수 df(w)는 w의 포스팅 개수 즉, CARD(R(W))와 같으므로 정의 9에 의해 df(w) = CARD(R(W)) = $N \times$ F(R(w))가 성립한다. 즉, df(w)와 F(R(w))는 서로 정비례한다. 따라서, df(w)의 분포는 F(R(w))의 분포와 같으므로, 본 실험에서는 F(R(w))가 그림 7과 같은 네 가지 형태의 베타분포 *beta*(α_1, α_2)를 가질 때에 대하여 각각 실험하였다. *beta*(1, 5) 분포에서는 문서빈도수가 작은 키워드들이 질의에 많이 주어지고, *beta*(5, 5) 분포에서는 문서빈도수가 중간인 키워드들이 질의에 많이 주어지며, *beta*(5, 1.5) 분포에서는 문서빈도수가 큰 키워드들이 질의에 많이 주어지고, *beta*(1, 1) 분포에서는 질의에 주어지는 키워드들의 문서빈도수가 균등하다.

5.2 논리곱 불리언 질의 처리 비용 모델

논리곱 불리언 질의는 조인키 정렬 릴레이션에 대한 조인과 동일하므로 논리곱 불리언 질의 처리 비용 모델도 수식 (4)와 같다. 수식 (4)에서 N은 통계치로서 주어져 있고, F(R(w))는 역색인의 키워드 목록에서 w의 문서빈도수 df(w)를 읽어 계산할 수 있다. 따라서, RS(R(w))와 RA(R(w))만 추정하면 된다. RS(R(w))는 키워드 w의 포스팅 리스트를 모두 읽는 비용이므로 포스팅 리스트가 저장된 디스크 블럭 개수와 같고 이를 수식 (5)와 같이 추정할 수 있다.

$$RS(R(w)) = \left\lceil \frac{\rho \times df(w)}{\lambda \times u} \right\rceil \quad (5)$$

3) 문서식별자 4 bytes, 문서내 발생회수 2 bytés발생위치(Line No, Word No) 6 바이트, 평균 발생위치 개수 10 개를 가정하면 포스팅의 평균 크기4 + 2 + 6 * 10 = 66 바이트이다

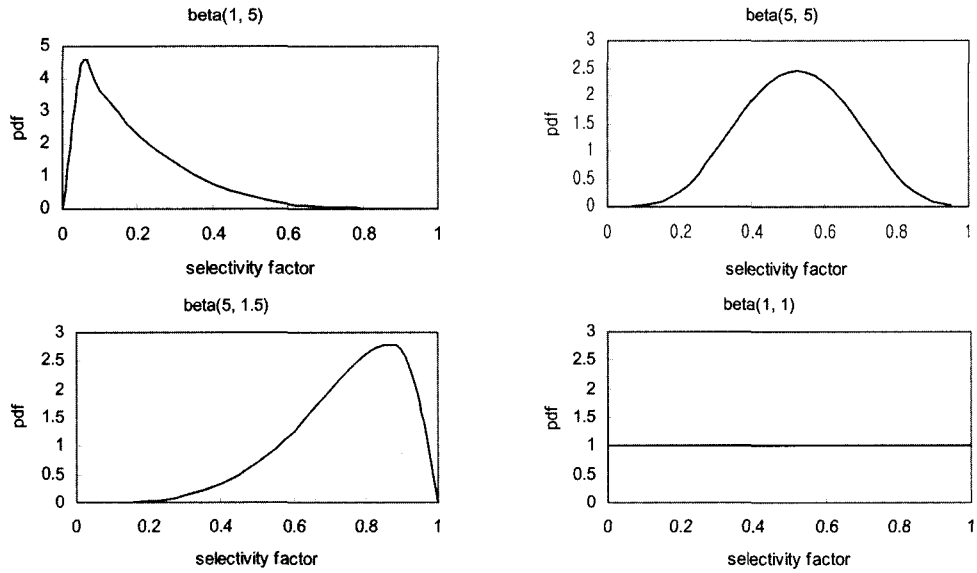


그림 7 사용자 질의에 대한 F(R(w))의 확률 분포

RA(R(w))는 키워드 w의 포스팅 리스트에서 특정 포스팅을 찾기 위한 비용이므로, CARD(R(w)) ≤ t 이면 RA(R(w))는 RS(R(w)) / 2와 같고 CARD(R(w)) > t 이면 B-tree 의 루트 노드부터 탐색하는 비용 즉, B-tree의 높이와 같다[17]. 그런데, 포스팅 리스트가 문서 식별자 순으로 정렬되어 있으므로, 구축된 B-tree는 포스팅 리스트가 저장된 각 디스크 블럭의 첫번째 포스팅만 가리킨다. 그러므로, B-tree의 인덱스 엔트리 개수는 포스팅 리스트의 디스크 블럭 개수와 같다. 따라서, RA(R(w)) 를 수식 (6)과 같이 추정할 수 있다.

$$RA(R(w)) = \begin{cases} \dfrac{RS(R(w))}{2} & if\ CARD(R(w)) \le t \\ \lceil \log_b RS(R(w)) \rceil + 1 & if\ CARD(R(w)) > t \end{cases} \quad (6)$$

여기서, b는 B-tree의 노드 블럭킹 계수(node blocking factor)로서, B-tree의 노드 엔트리 구조가 <문서 식별자, 포스팅 리스트 블럭 포인터, 서브 트리 포인터> 일 경우 수식 (7)과 같이 추정할 수 있다.

$$b = \left\lfloor \frac{\lambda \times u}{\alpha + \beta + \beta} \right\rfloor \quad (7)$$

5.3 최적화 정도에 대한 성능 평가

본 절에서는 질의에 주어지는 키워드의 개수를 2 개부터 32 개까지 3 개씩 증가시켜 가면서, 제안한 질의 최적화 알고리즘 OptJoinSeq(이후 OPT라 함)와 제2.2 절에서 설명한 중포-루프 조인을 사용하는 휴리스틱(이후 NLJ라 함) 그리고 병합 조인을 사용하는 휴리스틱(이후 MJ라 함)의 최적화 정도에 대한 성능을 평가한다. 평가 방법은 각 알고리즘에 의해 생성된 질의 처리 방법의 비용을 측정하여 서로 비교해 본다. 비용이 작을수록 최적화 정도가 우수하다.

표 2에 주어진 통계치를 사용하면 t ≥ 187일 때 RA(R(w)) < RS(R(w)) / 2가 성립한다. 그런데, 이 범위 내에서 t가 낮으면 많은 수의 포스팅 리스트에 B-tree가 구축되고, 높으면 적은 수의 포스팅 리스트에 B-tree가 구축된다. 따라서, 본 실험에서는 t가 낮을 때(t = 5000), 중간일 때(t = 50,000), 그리고 높을 때(t = 500,000) 등 세 가지 경우에 대한 성능 평가를 수행하였다.

t = 5000인 경우에 대한 실험 결과를 살펴보면, 그림 8과 같이 전반적으로 F(R(w))의 분포에 상관 없이 질의에 주어진 키워드의 개수가 많을수록 OPT와 기존 휴리스틱의 질의 처리 비용 차이가 크다. 실제로, 키워드가 32개 일때, 두 개의 휴리스틱 중 비용이 작은 것과 OPT를 비교해 보면, *beta*(1,5) 분포에서 31배, *beta*(5,5) 분포에서 8배, *beta*(5,1.5) 분포에서 2배, *beta*(1,1) 분포에서 21배까지 OPT 가 우수하다.

두 휴리스틱을 비교해 보면 어느 것이 더 항상 우수하다고 말할 수 없다. *beta*(1, 5)와 *beta*(1, 1) 분포에서는 키워드 개수가 작을 경우 MJ가 NLJ 보다 더 우수하지만, 키워드 개수가 많을 경우 문서빈도수가 작은 키워드가 질의에 많이 나타나므로 NLJ가 MJ 보다 더 우수하다. 그러나, *beta*(5, 5)와 *beta*(5, 1.5) 분포에서는 문서빈도수가 큰 키워드들이 대부분 질의에 나타나므로 MJ가 NLJ 보다 더 우수하다.

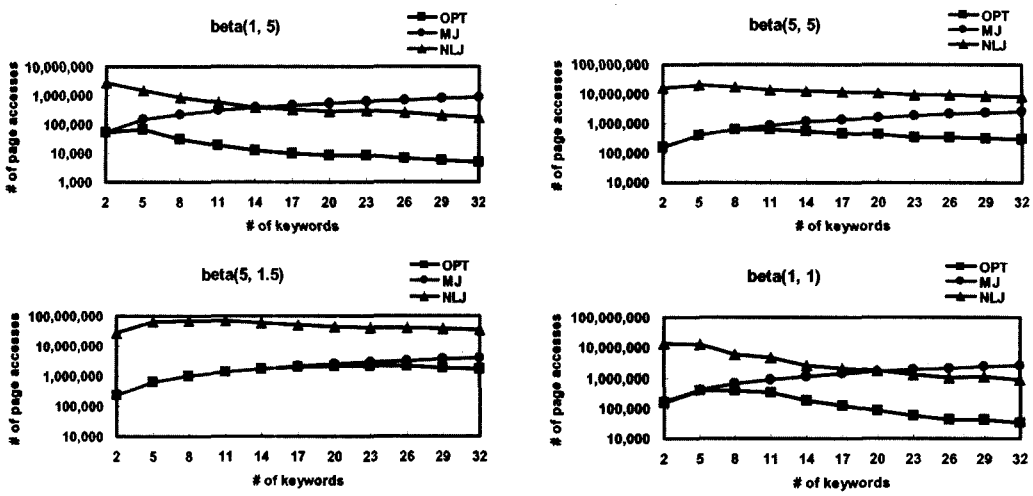


그림 8 질의 처리 비용 비교(t = 5000인 경우)

두 휴리스틱 알고리즘의 그래프 패턴을 살펴 보면 F(R(w))의 분포에 상관 없이 전반적으로 유사하다. MJ는 키워드 개수의 증가에 따라 비용이 증가하는 반면에 NLJ는 감소한다. MJ 비용은 질의에 포함된 모든 키워드의 RS(R(w)) 비용을 합한 것이므로 키워드 개수가 많을수록 증가한다. NLJ 비용의 경우, 질의에 포함되는 키워드의 개수가 많아지면 문서빈도수가 매우 작은 키워드들이 질의에 많이 포함되므로 이것들을 먼저 조인하면 조인 결과수가 더욱 작아져 NLJ 비용이 감소한다. 이러한 효과는 문서빈도수가 작은 키워드들이 질의에 나타날 확률이 높은 *beta*(1, 5) 분포와 *beta*(1, 1) 분포에서 더욱 크게 나타난다.

OPT의 그래프 패턴을 살펴 보면 키워드 개수가 작은 구간에서는 비용이 증가하고, 키워드 개수가 많은 구간에서는 비용이 감소한다. 이것은 OPT가 MJ나 NLJ와 달리 조인 방법을 한 가지만 사용하지 않고 두 가지를 모두 사용하기 때문이다. 즉, 키워드 개수가 아주 작으면 최적 조인 시퀀스 상에 병합 조인만 나타나므로 MJ의 패턴을 따라 비용이 증가하고, 키워드 개수가 많아지면 최적 조인 시퀀스 상에 중포-루프 조인이 나타나므로 NLJ의 패턴을 따라 비용이 감소한다. 이때, OPT의 비용이 감소하는 지점부터는 키워드 개수가 증가할수록 OPT와 MJ의 비용 차이가 점점 더 커지고, OPT와 NLJ의 비용 차이는 OPT의 비용이 감소하는 지점에서의 차이가 계속 유지된다. 이 차이는 OPT가 최적 조인 시퀀스 상의 전반부에서 병합 조인을 함으로써 얻어진 것이다. 한편, 문서빈도수가 매우 큰 키워드가 질의에 많이 나타나는 *beta*(5, 1.5) 분포에서는 최적 조인 시퀀스 상에서 병합 조인이 거의 대부분을 차지하므로 OPT와 MJ의 비용 차이가 크지 않다.

t = 50,000인 경우에 대한 실험 결과를 살펴보면 그림 9와 같다. t = 5000인 경우에 비해 B-tree가 구축된 포스팅 리스트의 수가 적어졌으므로 NLJ의 비용이 상대적으로 증가하였으나, 전반적인 그래프의 패턴은 t = 5000인 경우와 유사하다. 실제로, 키워드가 32개 일 때, 두 개의 휴리스틱 중 비용이 작은 것과 OPT를 비교해 보면, *beta*(1,5) 분포에서 188배, *beta*(5,5) 분포에서 8배, *beta*(5,1.5) 분포에서 2배, *beta*(1,1) 분포에서 23배까지 OPT가 우수하다.

t = 500,000인 경우에 대한 실험 결과를 살펴보면 그림 10과 같다. 이 경우는 B-tree가 구축된 포스팅 리스트의 수가 매우 적으므로 NLJ의 비용이 크게 증가하였다. 따라서, 모든 분포에서 MJ가 NLJ 보다 성능이 우수하다. 특히, *beta*(1, 5) 분포와 *beta*(1, 1) 분포에서 NLJ의 비용이 MJ 보다 훨씬 커짐에 따라 OPT와 휴리스틱의 성능 차이가 현격하게 커졌다. 실제로, 키워드가 32개 일때, 두 개의 휴리스틱 중 비용이 작은 것과 OPT를 비교해 보면, beta(1,5) 분포에서 186배, *beta*(5,5) 분포에서 8배, *beta*(5,1.5) 분포에서 2배, *beta*(1,1) 분포에서 59배까지 OPT가 우수하다.

5.4 최적화 시간에 대한 성능 평가

본 절에서는 동적 프로그래밍(DP)과 OPT 알고리즘의 질의 최적화 시간에 대한 성능을 비교 평가한다. 이를 위하여 DP와 OPT 각각에 대한 질의 최적화기를 구현하였다. 질의 최적화 시간에 대한 성능 평가는 최적 질의 처리 방법을 찾기 위해 탐색하는 경우의 수(number of enumerations)와 최적화에 소요된 CPU 수행 시간으로 측정한다. 탐색하는 경우의 수가 클수록 질

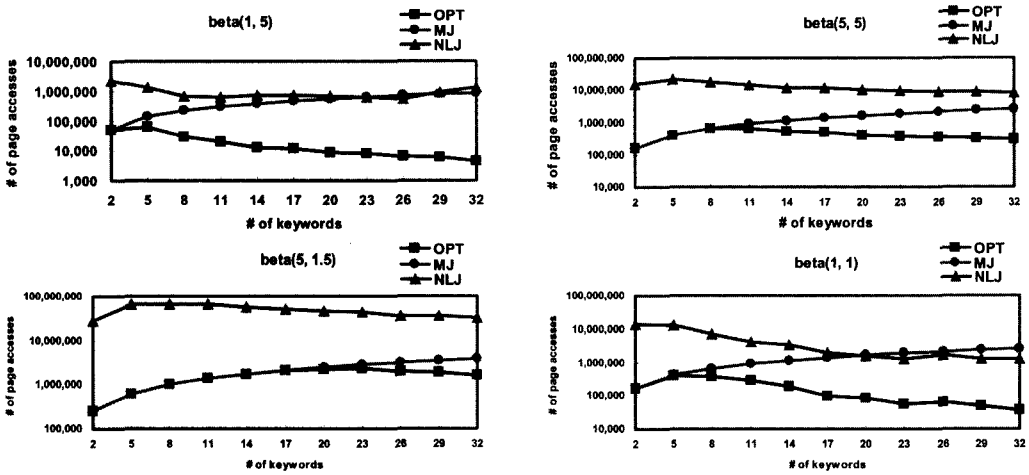


그림 9 질의 처리 비용 비교(t = 50,000인 경우)

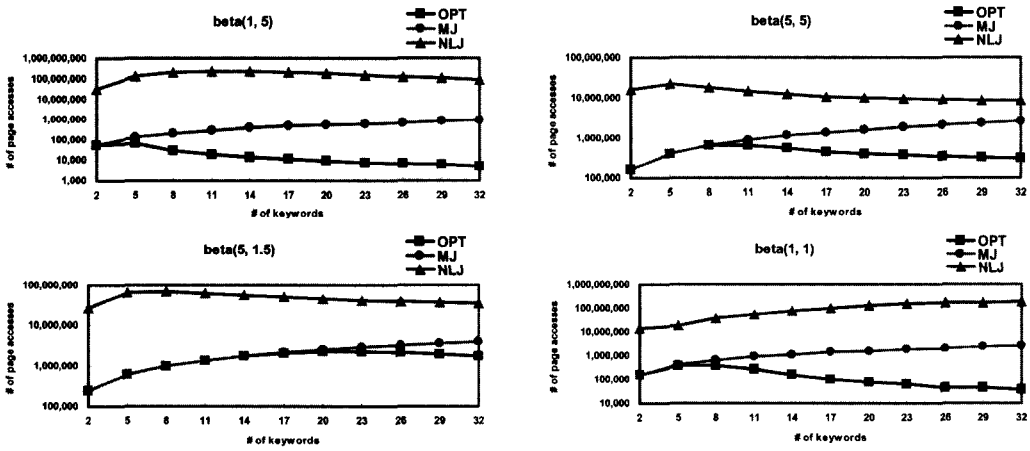
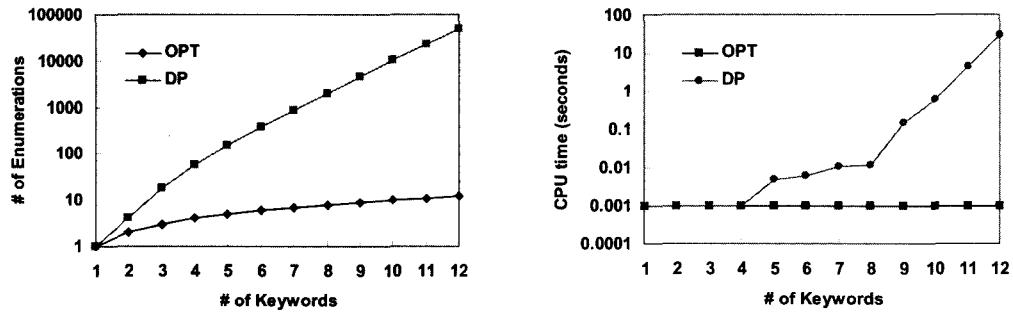


그림 10 질의 처리 비용 비교(t = 500,000인 경우)



(a) 탐색 회수 (b) 수행 시간

그림 11 선택률이 OPT와 DP의 질의 최적화 시간 비교

의 최적화 시간이 오래 걸린다.

그림 11(a)는 질의에 포함된 키워드의 개수가 1 개부터 12 개까지 증가함에 따라 OPT와 DP의 총 탐색 회수 실험 결과를 보여 주고 있다. 키워드 개수가 증가함에 따라 OPT는 탐색 회수가 거의 선형적으로 증가하는데 반해, DP는 기하 급수적으로 증가한다. 그림 11(b)

는 OPT와 DP의 질의 최적화 소요 시간 실험 결과를 보여 주고 있다. 키워드 개수가 증가하더라도 OPT는 질의 최적화 소요 시간이 거의 0 초에 가깝지만, DP는 키워드 개수가 12 개인 경우에 60 초 정도의 시간이 소요되었다.

6. 결 론

본 논문에서는 먼저 조인키 정렬 릴레이션에 대한 최적 조인 시퀀스를 찾기 위하여 조인 시퀀스 분리성 이라는 새로운 개념을 정형적으로 제안하였다. 이를 위해, 먼저 조인키 정렬 릴레이션과 비용 불변 조인 연산 및 비용 감소 조인 연산의 개념을 정의하였다. 그리고, 조인에 참여하는 릴레이션들이 모두 조인키 정렬 릴레이션일 경우, 최적 조인 시퀀스는 비용 불변 조인 시퀀스와 비용 감소 조인 시퀀스로 분리되는 조인 시퀀스 분리성이 있음을 정형적으로 밝혀 내었다. 조인 시퀀스 분리성을 이용하면 최적 질의 처리 방법을 구하는 알고리즘의 복잡도를 많이 줄일 수 있다.

다음으로, 조인 시퀀스 분리성 개념을 이용하여 새로운 질의 최적화 알고리즘을 제안하고,제안한 알고리즘의 최적성을 이론적으로 증명하였다. 제안한 질의 최적화 알고리즘은 두 부분으로 이루어지는데, 하나는 조인 시퀀스 분리성에 따른 두 개의 서브 시퀀스를 찾는 알고리즘이고, 다른 하나는 비용 감소 조인 서브 시퀀스 내에서의 최적 조인 순서를 찾는 알고리즘이다. 두 알고리즘 모두 복잡도가 $O(n\log n)$이다. 이는 기존의 데이타베이스 질의 최적화에서 널리 사용되는 동적 알고리즘 [5,18]의 복잡도가 $O(n2^{n-1})$인 것에 비해 매우 작다.

다음으로, 제안한 질의 최적화 알고리즘을 논리곱 불리언 질의 최적화에 적용하고 실험하였다. 실험 결과, 본 논문에서 제안한 질의 최적화 알고리즘 OPT는 기존의 휴리스틱 알고리즘들에 비해 질의 처리 성능이 최고 188배까지 우수함을 보였다. 또한, 동적 프로그래밍 알고리즘 DP와 질의 최적화에 소요되는 시간을 비교한 결과, 질의에 포함된 키워드의 개수가 10개 이상이 되면 DP는 소요 시간이 지수적으로 증가하지만, OPT는 $O(n\log n)$으로서 모든 실험에서 0.001초 이내로 매우 작음을 보였다.

본 논문의 중요성은 조인 애트리뷰트가 같은 조인키 정렬 릴레이션에 대한 질의 최적화를 위해 조인 시퀀스 분리성이라는 개념을 제안하고, 이에 기반한 질의 최적화 알고리즘을 정형화 하였다는 데 있다. 특히, 역색인을 이용한 논리곱 불리언 질의 최적화가 이의 한 예임을 보이고, 본 논문에서 제안한 질의 최적화 알고리즘을 적용하여 질의 처리 성능을 크게 개선하였다. 앞으로, 제안한 개념과 알고리즘에 기반한 여러 가지 응용 연구들을 도출하려 한다.

참 고 문 헌

[1] Jones, K.-S. and Willett, P., *Readings in Information Retrieval*, Morgan Kaufmann Publishers, 1997.

[2] Witten I.-H., Moffat, A., and Bell, C.-B., *Managing Gigabytes -- Compressing and Indexing Documents and Images*, Van Nostrand Reinhold, New York, 1994.

[3] NIST, TREC-2 ad hoc & TREC-3 routing topics 101 - 150, http://trec.nist.gov/

[4] Faloutsos, C., "Access Methods for Text," *ACM Computing Surveys*, Vol. 17, No. 1, pp. 49-74, 1985.

[5] Chaudhuri, S. and Shim, K., "Optimization of Queries with User-Defined Predicates," *ACM Trans. on Database Systems*, Vol. 24, No. 2, pp. 117-228, June 1999.

[6] Whang, K.-Y. and Wiederhold, G., and Sagalowics, D., "Separability -- An Approach to Physical Database Design," *IEEE Trans. on Computers*, Vol.C-33, No.3, pp. 209-222, Mar. 1984.

[7] Frakes, W.-B. and Baeza-Yates, R., *Information Retrieval -- Data Structures & Algorithms*, Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[8] Salton, G., *Automatic Text Processing -- The Transformation, Analysis, and Retrieval of Information by Computer*, Addison-Wesley, 1988.

[9] Zhang, C., Naughton, J., DeWitt, D., Luo, Q., and Lohman, G., "On Supporting Containment Queries in Relational Database Management Systems," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, Santa Barbara, May 2001.

[10] Whang, K.-Y., Park, B.-K., Han, W.-S., and Lee, Y.-K., "An Inverted Index Storage Structure Using Subindexes and Large Objectsfor Tight Coupling of Information Retrieval with Database Management Systems," *United States Patent No. 6349308*, Feb. 2002.

[11] Krishnamurthy, R., Boral, H., and Zaniolo, C., "Optimization of Nonrecursive Queries," In *Proc. Int'l Conf. on Very Large Data Bases*, pp. 128-137, Kyoto, Aug. 1986.

[12] Whang, K.-Y. and Krishnamurthy, R., "Query Optimization in a Memory-Resident Domain Relational Calculus Database System," *ACM Trans. on Database Systems*, Vol.15, No.1, pp. 67-95, Mar. 1990.

[13] Monma, C.-L. and Sidney, J., "Sequencing with Series-Parallel Precedence Constraints," *Mathematics*

of Operations Research, Vol. 4, No. 3, pp. 215-224, Aug. 1979.

[14] Ibaraki, Toshihide and Kameda, Tiko, "On the Optimal Nesting Order for Computing N-Relational Joins," *ACM Trans. on Database Systems*, Vol. 9, No. 3, pp. 482-502, Sept. 1984.

[15] Hellerstein, J.-M. and Stonebrake, M., "Predicate Migration: Optimizing queries with expensive predicates," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, pp. 267-276, Washington D.C., May 1993.

[16] Hellerstein, J.-M., "Practical Predicate Placement," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, pp. 325-335, Minneapolis, MN, May 1994.

[17] Elmasri, R. and Navathe, S.-B., *Fundamentals of Database Systems*, 3rd Edition, The Benjamin/Cummings, Redwood City, California, 2000.

[18] Selinger, P.-G., Astrahan, M.-M., Chamberlin, D.-D., Lorie, R.-A., and Price, T.-G., "Access Path Selection in a Relational Database Management System," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, pp. 23-34, Boston, May 1979.

부록 A: 보조정리 3의 증명

정의 9에 의하여 F(R) < F(T)이면 CARD(R) < CARD(T)가 성립한다. 그러면, 다음과 같은 세 가지 경우가 존재하므로 각각에 대하여 증명한다.

• $CARD(R) < CARD(T) \le t$ 인 경우:

R과 T 모두 색인이 없으므로 RA(R) = RS(R)/2, RA(T) = RS(T)/2이다. 그런데, RS(R)∝CARD(R), RS(T)∝CARD(T)이므로 RS(T)/RS(R) = CARD(T)/CARD(R)이 성립하고, 정의 9에 의하여 F(T)/F(R) = CARD(T)/CARD(R)이 성립하므로 결국 F(T)/F(R) = RS(T)/RS(R)이 성립한다. 따라서, F(T)/F(R) = RS(T)/RS(R) = (RS(T)/2)/(RS(R)/2) = RA(T)/RA(R)이 성립한다.

• $CARD(R) \le t < CARD(T)$ 인 경우:

T만 색인을 가지므로 RA(R) = RS(R)/2, RA(T) < RS(T)/2이다. 따라서, F(T)/F(R) = RS(T)/RS(R) = (RS(T)/2)/(RS(R)/2) > RA(T)/RA(R)이 성립한다.

• $t < CARD(R) < CARD(T)$ 인 경우:

R과 T 모두 색인을 가지므로 RA(R) ∝ log(CARD(R)), RA(T) ∝ log(CARD(T))이다. 따라서, F(T)/F(R) = CARD(T)/CARD(R) ≥ log(CARD(T))/log(CARD(R)) = RA(T)/RA(R)이 성립한다. □

부록 B: 보조정리 4의 증명

임의의 두 조인키 정렬 릴레이션 R_i, R_j에 대하여 $r(R_i) \ge r(R_j)$가 성립하면, 임의의 서브 시퀀스 S_1, S_2에 대해 $C(S_1R_iR_jS_2) \le C(S_1R_jR_iS_2)$가 성립함을 보인다. 이를 위해, C를 다음과 같이 재귀적으로 표현한다.

$C(\Lambda) = 0$ *for null sequence* Λ

$C(R) = N_t \times RA(R)$ *for relation* R

$C(S_1S_2) = C(S_1) + F(S_1)C(S_2)$ *for subsequences* S_1 *and* S_2

그러면, 다음이 성립한다.

$$C(S_1R_iR_jS_2) = C(S_1) + F(S_1)C(R_iR_jS_2)$$
$$= C(S_1) + F(S_1)(C(R_i) + F(R_i)C(R_jS_2))$$
$$= C(S_1) + F(S_1)(C(R_i) + F(R_i)(C(R_j) + F(R_j)C(S_2)))$$
$$= C(S_1) + F(S_1)C(R_i) + F(S_1)F(R_i)C(R_j) + F(S_1)F(R_i)F(R_j)C(S_2)$$

따라서, 다음이 성립한다.

$$C(S_1R_iR_jS_2) - C(S_1R_jR_iS_2)$$
$$= F(S_1)(C(R_i)(1 - F(R_j)) - C(R_j)(1 - F(R_i)))$$
$$= F(S_1)C(R_i)C(R_j)\left(\frac{1 - F(R_j)}{C(R_j)} - \frac{1 - F(R_i)}{C(R_i)}\right)$$
$$= F(S_1)N_tRA(R_i)N_tRA(R_j)\left(\frac{1 - F(R_j)}{N_tRA(R_j)} - \frac{1 - F(R_i)}{N_tRA(R_i)}\right)$$
$$= N_tF(S_1)RA(R_i)RA(R_j)(r(R_j) - r(R_i)) \le 0$$

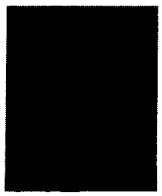
즉, $C(S_1R_iR_jS_2) \le C(S_1R_jR_iS_2)$ 이 성립한다. □

박 병 권

1986년 2월 서울대학교 산업공학과 학사 1988년 2월 한국과학기술원 경영과학과 석사. 1998년 8월 한국과학기술원 전산학과 박사. 1998년 9월~2000년 2월 삼성전자 중앙연구소 선임연구원. 2000년 3월~현재 동아대학교 경영정보과학부 조교수. 관심분야는 DBMS, XML, OLAP, Data Mining

한 욱 신

2003년 3월~현재 전임강사, 컴퓨터공학과, 경북대학교. 2002년 9월~2003년 2월 연구교수, 첨단정보기술연구센터, KAIST 2001년 9월~2002년 8월 포스트닥, 첨단정보기술연구센터, KAIST. 1996년 3월~2001년 8월 Ph.D, 전산학과, KAIST 1994년 3월~1996년 2월 MS, 전산학과, KAIST. 1990년 3월~1994년 2월 BS, 컴퓨터공학과, 경북대학교. 2002년 7월 방문연구원, 미국 HP Labs. 2001년 7월~2001년 8월 Visiting Scholar, 미국 HP Labs



황 규 영

1973년 서울대학교 전자공학과 졸업(B.S). 1975년 한국과학기술원 전기 및 전자학과 졸업(M.S.). 1982년 Stanford University(M.S.). 1983년 Stanford University(Ph.D.). 1975년~1978년 국방과학연구소(ADD), 선임연구원. 1983년~1990년 IBM T.J. Watson Research Center, Research Staff Member. 1992년~1994년 한국정보과학회 데이타베이스 연구회(SIGDB) 운영위원장. 1995년 한국정보과학회 이사 겸 논문지 편집위원장. 1999년~2000년 한국정보과학회 부회장. Editior: the VLDB Journal, 1990년~현재 Editior: Distributed and Parallel Databases: An International Journal, 1991년~1995년 Editor: International Journal of Geographical Information Systems, 1994년~현재 Associate Editor: IEEE Data Engineering Bulletin, 1990년~1993년 IEEE Transactions on Knowledge and Data Engineering, 2002년~현재. 1998년~2004년 Trustee, The VLDB Endowment. 1999년~2005년 Steering Committee Member, DASFAA. 1999년~현재 한국과학기술원 전자전산학과 전산학전공 교수 1990년~현재 첨단정보기술연구센터(과학재단 우수연구센터) 소장. 관심분야는 데이타베이스 시스템, 멀티미디어, GIS