DOI: 10.3745/KIPSTD.2009.16D.6.859

관계형 데이터 스트림에서 고급 키워드 검색을 위한 질의 최적화

주 진 웅[†] · 김 학 수[††] · 황 진 호[†††] · 손 진 현[††††]

요 약

관계형 데이터베이스 기반의 키워드 검색 기법에 대한 연구에서의 관심에도 불구하고 관계형 데이터 스트림 기반의 연구는 아직 미흡한 수준이다. 오늘날 스트리밍 데이터는 데이터 관리 측면에서 중요한 연구 토픽이기 때문에 관계형 데이터 스트림 기반의 키워드 검색 기법에 대한 연구는 매우 중요하다. 이러한 관점에서 본 논문은 관계형 데이터 스트림 기반의 키워드 검색 기법과 관련된 연구들을 먼저 분석하고 키워드 검색 질의를 처리하는 동안에 발생되는 조인 비용을 최소화하는 기법에 대해 초점을 둔다. 결과적으로 본 논문은 관계형 데이터 스트림에서 사용자를 위해 좀 더 의미 있는 질의 결과를 산출하기 위한 고급 키워드 검색 기법을 제안하고 효율적인 질의 처리를 위한 계층적 클러스터링을 사용한 질의 최적화 기법을 제안한다.

키워드 : 키워드 검색, 스트림 데이터, 클러스터링, 데이터베이스, 질의 최적화

Query Optimization for an Advanced Keyword Search on Relational Data Stream

Jinung Joo[†] · Hak Soo Kim[††] · Jin-Ho Hwang[†††] · Jin Hyun Son[††††]

ABSTRACT

Despite the surge in the research for keyword search method over relational database, only little attention has been devoted to studying on relational data stream.The research for keyword search over relational data stream is intense interest because streaming data is recently a major research topic of growing interest in the data management. In this regard we first analyze the researches related to keyword search methodover relational data stream, and then this paper focuses on the method of minimizing the join cost occurred while processing keyword search queries. As a result, we propose an advanced keyword search method that can yield more meaningful results for users on relational data streams. We also propose a query optimization method using layered-clustering for efficient query processing.

Keywords : Keyword Search, Data Stream, Clustering, Database, Query Optimization

1. 서 론

최근 센서 데이터, 위치 기반 데이터, RFID 태그 데이터와 같은 데이터 스트림에서의 질의처리 기법에 관한 연구가 활발히 진행되고 있지만, 이들 연구는 사용자가 데이터구조뿐만 아니라 XPath 및 XQuery와 같은 XML 질의 언어 혹은 SQL과 같은 질의 언어 등에 대한 사전 지식을 요구하는 검색 기법에 초점을 두고 있다. 이러한 장벽은 사용자가 쉽게 원하는 정보를 검색하는데 장애물이 되고 있는 실정이다. 오늘날 이러한 단점을 극복하기 위해서 정보 검색 분야에서 연구되어 온 키워드 검색의 장점을 적용하기 위한 연구가 활발히 진행되고 있다. 이러한 상황에도 불구하고 관계형 데이터베이스 기반의 데이터 스트림에서의 키워드 검색 기법에 대한 연구는 아직 미흡한 상황이다 [2]. 또한, 기존의 키워드 검색 기법은 사용자 질의 키워드를 모두 포함하는 결과에 대한 검색 성능에 초점을 두고 있어서 논리 연산자를 통한 키워드의 다양한 조합에 따른 의미 있는 질의 검색에는 한계를 드러내고 있다.

※ 이 논문은 2007년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었음(KRF-2007-313-D00757). 이 논문은 2007년도 정부(과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임(No.R01-2007-000-20135-0).
† 준 회 원 : 한양대학교 컴퓨터공학과 석사과정
†† 준 회 원 : 한양대학교 컴퓨터공학과 박사과정
††† 정 회 원 : LG전자 MC 연구소
†††† 종신회원 : 한양대학교 컴퓨터공학과 부교수
논문접수 : 2009년 7월 23일
수 정 일 : 1차 2009년 9월 11일
심사완료 : 2009년 10월 6일

이러한 관점에서 본 논문은 사용자가 논리 연산자를 통해서 보다 의미 있는 키워드 질의를 구성하고 이를 통해 보다 정확한 정보를 빠르게 검색할 수 있도록 하는 고급 키워드 검색 기법을 제안한다. 다시 말해서 제안된 기법은 사용자가 단순히 키워드만을 이용해 질의하는 것이 아니라 AND, OR, NOT과 같은 논리 연산자를 사용하여 키워드들 사이의 의미를 부여할 수 있도록 설계되었다. 또한 최근 그 활용이 증가되고 있는 데이터 스트림에서 키워드 검색 기법에 관한 연구는 아직 초기 단계이며 스트림데이터의 특성을 반영한 질의 최적화에 대한 연구가 요구되고 있다. 데이터 스트림에서 사용자가 질의한 연속질의(continuous query)에 의한 주기적인 질의 처리 수행은 시스템의 성능에 많은 영향을 주게 된다 [3]. 따라서 본 논문에서는 관계형 데이터베이스 기반의 데이터 스트림에서 사용자의 연속 질의에 대한 시스템 성능을 향상시키기 위한 계층적 클러스터링(layered clustering) 기반의 질의 최적화 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 키워드 검색에 관한 기존연구들에 대하여 살펴본다. 3장에서는 본 논문에서 제안하는 고급 키워드 검색시스템에 대하여 설명하고, 계층적 클러스터링을 통한 질의 최적화 방안에 대하여 4장에서 기술한다. 5장에서는 실험을 통해 제안한 시스템의 성능을 평가 분석하고 마지막으로 6장에서 결론 및 향후 연구에 대하여 서술한다.

2. 관련 연구

이 장에서는 관계형 데이터베이스와 데이터 스트림에서의 키워드 검색 기법에 대한 기존 연구들을 분석한다. 먼저 2.1절에서는 키워드 검색 기법에 설명하고, 2.2절에서 본 논문에서 개선하고자 하는 데이터 스트림에서 수행되는 키워드 검색 기법에 대해 기술한다.

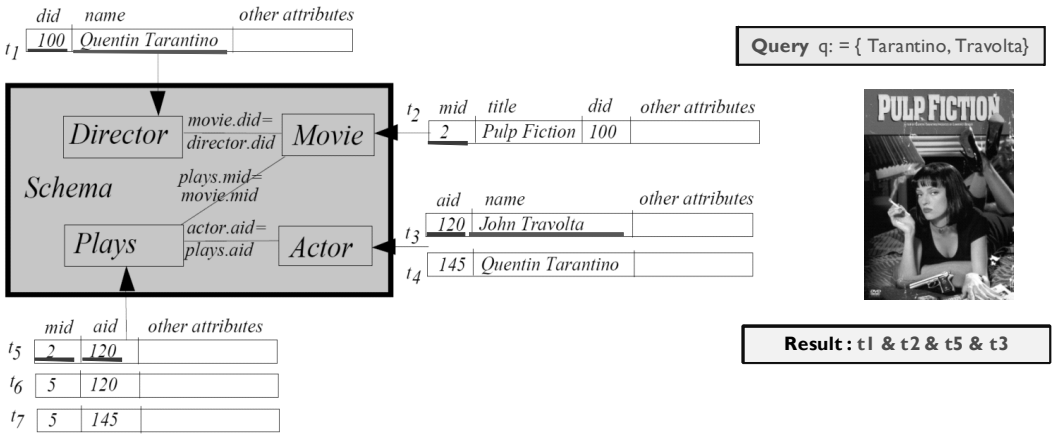
2.1 키워드 기반 검색 기법

현재까지 연구되어온 키워드 검색 기법들은 저장소로부터 추출되는 데이터 모델에 따라 크게 두 가지로 분류할 수 있다 [1]. 일반적으로 키워드 검색의 대상이 되는 저장소는 관계형 데이터베이스 또는 XML 데이터베이스에 초점을 두고 있다. 이것은 오늘날 데이터 저장소로서 가장 많이 사용되기 때문이며 이들 저장소로부터 추출되는 데이터 모델의 형태에 따라 “그래프 기반 키워드 검색 기법”과 “관계기반 키워드 검색 기법”으로 분류할 수 있다.

첫째, 그래프기반 키워드 검색 기법이다. 전체 데이터베이스 내에 저장된 데이터를 하나의 그래프로 변환하여 검색을 수행하게 된다. 다시 말해, 외래 키(foreign key) 관계를 이용해 데이터베이스의 모든 데이터를 하나의 그래프로 표현하고, 그래프 안에서 사용자가 입력한 모든 키워드를 포함하고 있는 서브그래프를 찾는 방식이다. DataSpot, BANKS I/II와 같은 연구들이 이에 속한다 [4-6]. 이러한 그래프기반 키워드 검색 기법은 생성되는 그래프에 대한 유지 관리 비용이 크기 때문에 소규모 데이터베이스에 적합한 반면에 한번 그래프가 생성되면 빠른 질의 처리 성능을 보장한다.

둘째, 관계기반 키워드 검색 기법이다. 관계기반 키워드 검색 기법은 질의 결과로 모든 키워드를 포함하는 튜플(tuple) 집합을 생성한다. 즉, 질의 시점에 키워드를 포함하는 튜플들을 검색한 후에 이 튜플들로부터 데이터베이스 스키마에 정의된 외래 키 관계를 기반으로 하여 모든 키워드를 포함하는 조인 가능한 후보 네트워크를 생성하여 각 테이블의 튜플들을 조인 또는 선택(selection)하여 검색을 수행하는 방식이다 [3, 7-12]. 이러한 관계기반 키워드 검색 기법은 질의 시점에 역 인덱스를 통해서 빠르게 후보 네트워크를 생성하여 튜플 집합을 생성하기 때문에 복잡한 쿼리도 처리가 가능하여 큰규모의 데이터베이스에서도 빠른 질의 처리 성능을 보장한다 [1].

(그림 1)은 관계기반 키워드 검색기법을 사용한 키워드 검색의 한 예이다. 영화정보를 담고 있는 관계형 데이터베이스에서 사용자는 관심을 가지고 있는 키워드 Tarantino와 Travolta를 통해 질의한다. 사용자 키워드들이 포함된 튜플들과 외래 키 관계를 가지는 튜플들을 조인하고 조인된 튜플들 중 사용자 키워드를 모두 포함하고 있는 조인된 튜플



(그림 1) 예제 데이터 스키마와 튜플[3]

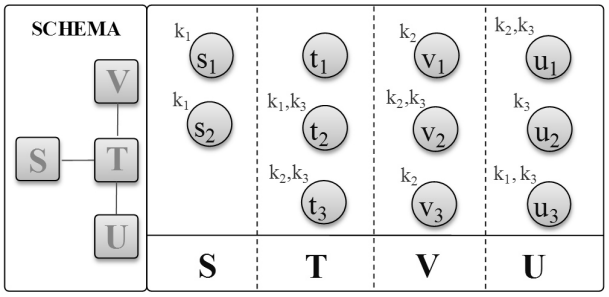
들을 검색의 결과로 사용자에게 돌려주게 된다. 아래 예제에서 입력된 두 키워드를 모두 포함하고 있는 튜플들의 내용을 통해 사용자는 Tarantino가 감독하고 Travolta가 주연으로 참여한 Pulp Fiction이라는 영화를 새로운 정보로 획득할 수 있다.

[정의 1] (Minimal Total Join Network of Tuples, MTJNT) : Join Network of Tuples(JNT) 란 데이터베이스의 튜플을 노드로 하는 트리 구조로써 부모 노드와 자식 노드에 해당하는 두 튜플들 사이에는 데이터베이스 스키마상에서 관계성(relationship)이 존재해야 한다. 즉, 트리 구조에서의 인접한 두 튜플을 t_1, t_2라 할 때 두 릴레이션 R_1, R_2가 각각 $t_1 \in R_1$, $t_2 \in R_2$ 를 만족한다면 R_1, R_2 에 대해 조인 연산이 가능해야 한다. 이러한 JNT가 Total 과 Minimal 두 가지 조건을 만족할 때에 이러한 JNT를 MTJNT라 정의한다. 여기서 Total 조건이란 질의 키워드 k_1, k_2, …… k_m 이 주어 졌을 때 모든 k_i가 JNT의 튜플들에 포함 되면 JNT는 Total 조건을 만족한다. 또한 Total 조건을 만족하는 JNT로부터 어떤 튜플을 제거하였을 때 더 이상 Total 조건을 만족하지 않는다면 이러한 JNT는 Minimal 조건을 만족한다. 이러한 MTJNT는 키워드 검색 시스템의 질의에 대한 최종 결과가 된다.

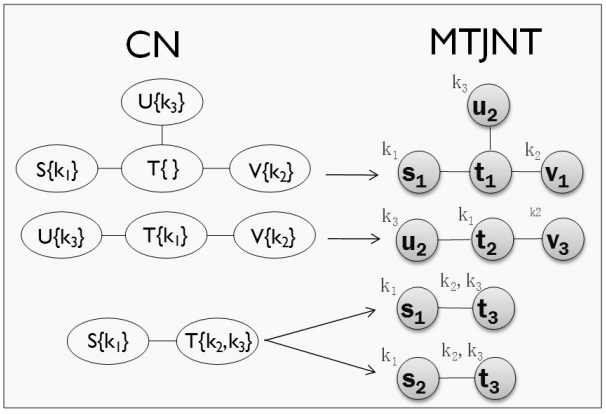
[정의 2] (Candidate Network, CN) : MTJNT의 모든 튜플들을 데이터베이스 스키마상에서 각 튜플에 해당하는 릴레이션과 튜플이 가지는 키워드의 쌍으로 맵핑시킨 것을 Candidate Network(CN)이라 한다. 검색 시스템은 질의 키워드에 대해 이를 포함하는 CN을 생성하고 이 CN을 데이터베이스 질의 언어로 변환하여 MTJNT를 추출한다.

모든 MTJNT를 찾는 가장 잘 알려진 기법은 Discover[7]에서 제안된 기법이다. 이는 튜플 집합 그래프 탐색을 통해 모든 MTJNT를 생성할 수 있는 모든 CN을 만들어 내는 것이다. 여기서 튜플 집합 그래프란 데이터베이스 스키마에 키워드들을 반영하여 만들어진다. 그래프내의 모든 노드들은 키워드 $K \subseteq \{k_1,...,k_m\}$을 포함하는 튜플들의 집합을 의미한다. 예를 들어, 노드 S{}는 관계(relation) S에서 어떠한 키워드도 포함하지 않는 튜플들의 집합을 의미한다.

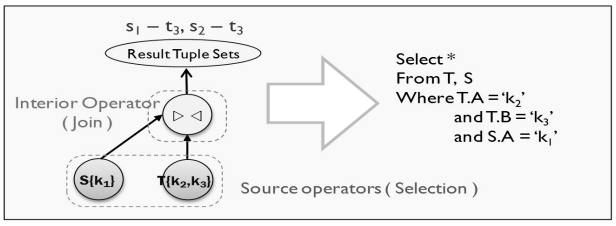
다시 말해, (그림 2)에서 테이블 S와T가 관계를 가지므로 키워드 k1을 포함하는 튜플 s_1과s_2는 튜플 t_3과 조인될 수 있다. 따라서 이들이 조인된다면 두개의 튜플 집합 s_1-t_3, s_2-t_3는 입력된 키워드를 모두 포함하는 가장 짧은 MTJNT가 된다. 이들을 추출할 수 있는 방법은 S{k_1}이라고 표현되는 테이블S에서 키워드 k_1을 포함하는 노드들과 T{k_2,k_3}로 표현되는 테이블 T에서 키워드 k_2, k_3을 포함하는 노드들 사이에 주-외래키 관계를 통해 조인하는 것이다. (그림 3)은 (그림 2)의 예제에서 추출될 수 있는 몇 개의 MTJNT와CN의 관계를 표현하고 있다. (그림 3)에서 보는 것처럼 하나의 CN은 하나 또는 다수의 MTJNT와 대응되며 이를 통해 키



(그림 2) 예제 데이터베이스 스키마와 튜플



(그림 3) 예제 CN과 MTJNT[3]



(그림 4) Operator Tree와 SQL

워드 검색의 결과인 모든 키워드를 포함하고 있는 MTJNT를 추출할 수 있다. (그림 3)에서 S{k_1}-T{k_2,k_3}로 구성된 하나의 CN을 실행트리(operator tree)로 나타낸 것을 보여주고 있다. 실행 트리는 CN의각 노드들을 리프노드로 가지며 중간 노드는 조인연산을 의미한다.

이러한 실행 트리를 실행함으로써 결과로 s_1-t_3, s_2-t_3와 같은 MTJNT를 추출할 수 있으며 이는 (그림 4)에 보이는 SQL문을 실행 하는 것과 같은 결과를 이끌어 낼 수 있다[4].

2.2 데이터 스트림에서 키워드 검색

위 2.1절에서 설명한 관계기반 키워드 검색 기법은 관계형 데이터베이스뿐만 아니라 데이터 스트림에서도 적용 가능하다. 관계형 데이터 스트림에서 키워드 검색은 Markowetz[3]에 의해 처음으로 제안되었다. 관계형 데이터 스트림에서 키워드 검색은 관계형 데이터베이스와 달리 데이터가 시간

에 따라 동적으로 변하기 때문에 아래와 같이 두가지 사항을 고려할 필요가 있다.

① 시간에 따라 연속적으로 변화하는 데이터
② 연속질의 처리에 따른 질의수행기간 동안 모든 실행트리 유지

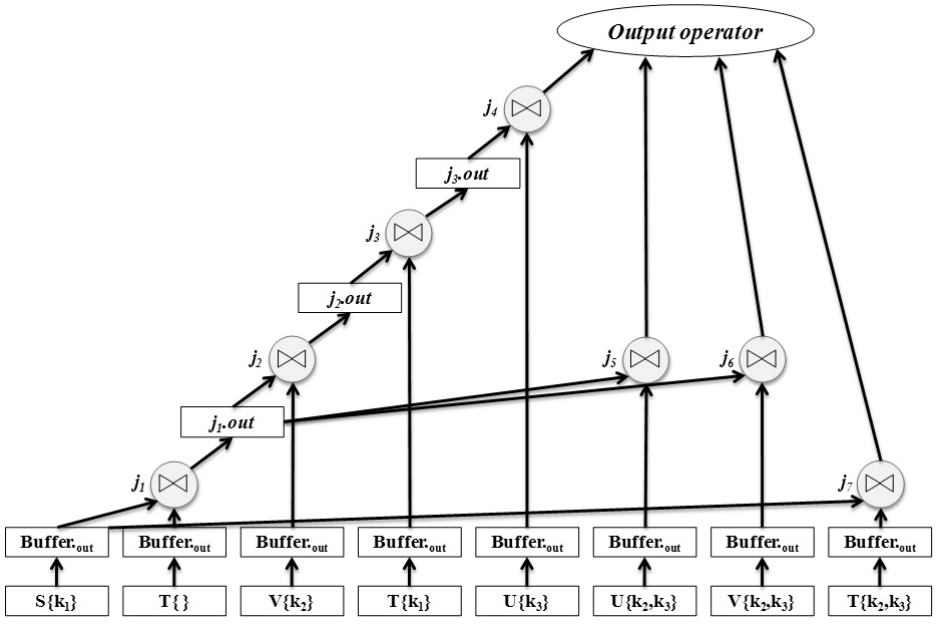
첫째, 시간에 따라 연속적으로 변화하는 데이터를 고려해야만 한다. 2.1절에서 설명된 것처럼 관계형 데이터베이스에서의 키워드 검색은 데이터베이스에 존재하는 데이터들만을 고려하여 CN을 생성한다. 이와 달리 관계형 데이터 스트림에서는 시간 슬라이딩 윈도우(Time Sliding Window) - 현재 시간으로부터 데이터가 유효한 과거 시간까지의 범위를 나타내며 시간 범위 밖의 데이터는 바로 삭제될 수도 있으나 시스템의 성능을 위해 시스템의 부하가 적은 시점에 일괄적으로 정리하는 경우가 대부분임 - 내에서 시간에 따라 연속적으로 변화하는 데이터를 처리하기 위하여 모든 출현 가능한 데이터를 고려하여 CN을 생성하여야 한다. 다시 말해, (그림 2)에서 테이블 S에 현재 존재하는 튜플은 s_1과 s_2 뿐이기 때문에 검색은 이 두 튜플들만을 고려해야만 한다. 하지만 데이터 스트림 환경에서 시간 슬라이딩 윈도우 내에 현재 키워드 k_1을 포함하는 튜플 s_1, s_2가 존재 하더라도 시간의 변화에 따라 키워드 k_2와 k_3를 포함하는 새로운 튜플 s_3가 추가되거나 기존 튜플 s_1이 사라질 수도 있다. 따라서 모든 출현 가능한 데이터를 고려하여 CN을 생성해야 한다.

두 번째, 연속질의 처리에 따른 질의수행기간 동안 모든 실행 트리를 유지해야만 한다. 데이터 스트림의 특성에 의해 사용자가 입력한 키워드들을 검색하는 질의는 단일수행 질의가 아닌 연속질의가 된다. 따라서 생성된 CN을 실행 트리로 변경한 후에 이들 트리는 질의가 수행되는 주기(life-time)까지 시스템 내에 유지해야 되기 때문에 이것은 실행 트리의 실행 주기 프로세스에 영향을 주게 된다.

이와 같은 데이터 스트림 환경에서 키워드 검색을 수행하기 위한 고려사항들을 기존 Markowetz의 연구[3]에서는 다음과 같이 처리하고 있다. 첫째, 현재 존재하는 데이터뿐 아니라 모든 출현 가능한 데이터를 모두 고려하여 만들어지는 많은 수의 CN 중 중복(duplication)을제거하는 기법을 제안하고 기존 CN 생성기(generator)를 보완하였다. 둘째, CN으로부터 만들어지는 많은 수의 실행 트리들 사이에 중간결과(intermediate result)를 서로 공유할 수 있도록 (그림 5)와 같은 Operator Mesh를 제안하였다. (그림 5)에 나타난 Operator Mesh는 $S\{k_1\}$-$T\{\}$-$V\{k_2\}$-$T\{k_1\}$-$U\{k_3\}$, $S\{k_1\}$-$T\{\}$-$U\{k_2,k_3\}$, $S\{k_1\}$-$T\{\}$-$V\{k_2,k_3\}$와 $S\{k_1\}$-$T\{k_2,k_3\}$ 네 개의 CN을 하나로 통합하여 그들 사이의 중간결과를 공유가 가능하게 만들었다.

마지막으로, 결과 값을 만들어 낼 수 없는 CN들을 한 번에 가지치기(pruning)하는 기법을 제안하였다. 하지만 이들의 Operator Mesh를 사용한 질의 최적화 기법에서는 키워드 k_1을 포함하는 동일한 노드를 루트 노드로 가지는 CN들 사이에서만 중간결과에 대한 공유가 가능하다. 자세히 말하면, (그림 5)는 Operator Mesh에 존재하는 $|SR| \cdot 2^{|k-1|}$개의 클러스터 중 노드 $S\{k_1\}$을 루트 노드로 하는 CN들의 클러스터이다. 따라서 (그림 4)에 나타난 클러스터는 노드 $T\{k_1\}$을 루트 노드로 가지는 CN들로 이루어진 클러스터와 중간결과를 공유할 수 없다.

결과적으로, Operator Mesh에 존재하는 클러스터들 사이에 중간결과를 공유할 수 없다는 단점을 가지고 있다. 본 논문에서는 이와 같은 문제를 해결할 수 있는 계층적 클러



(그림 5) S{k1}을 루트 노드로 가지는 Operator Mesh의 예[3]

스터링을 이용한 질의 최적화 기법을 제안하는데 초점을 둘 것이다. 또한, 앞서 언급한 논리 연산자 기반의 고급 키워드 검색을 제공하기 위해서는 AND, OR, NOT 연산자에 따른 조인 가능한 조인 튜플을 얻기 위한 CN 생성기를 고려해야만 한다.

3. 고급 키워드 검색

본 장에서는 고급 키워드 검색을 지원하기 위한 키워드 검색 기법을 제안한다. 먼저 3.1절을 통해 제안하는 기법을 간략히 소개하고, 3.2절을 통해 고급 키워드 검색을 지원할 수 있도록 보완 된 CN 생성기의 동작을 설명한다.

3.1 고급 키워드 검색 기법

지금까지 제안된 관계형 데이터베이스에서의 키워드 검색 기법들은 [7-8] 주어진 키워드들을 이용해 정보검색을 수행하며 그 결과 또한 단순히 모든 키워드들을 포함하고 있는 튜플들의 집합이었다. 관계형 데이터베이스 기반의 키워드 검색 기법에서는 현재 데이터베이스에 정적인 데이터만을 고려하며 단일수행 질의를 처리하여 한 번의 결과만을 생성하기 때문에 사용자는 검색된 결과에서 자신이 원하는 정보를 선택적으로 검토 수 있는 충분한 시간적 여유를 가질 수 있다. 하지만 데이터 스트림 환경에서는 시간에 따라 정보가 틀려지기 때문에 질의 시점에서 사용자에게 보다 정확하고 의미 있는 검색 결과를 제공할 필요가 있다. 이러한 관점에서 AND 연산자만 제공하는 단순한 질의 키워드는 보다 정확하고 의미 있는 검색 결과의 생성에 한계를 드러내기 때문에 OR, NOT과 같은 논리 연산자를 지원하는 고급 키워드 검색 기법을 요구한다.

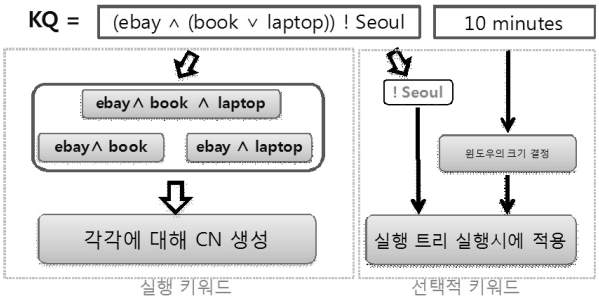
<표 1>에 나타난 연산자들을 이용하여 사용자들은 의미를 지닌 질의를 생성해 낼 수 있다. 아래는 위 연산자들을 사용하여 표현될 수 있는 질의 표기법(notation)이다.

$$Q = (A \wedge (B \vee C)) \wedge !D$$

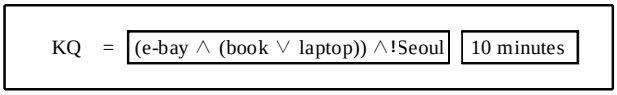
Q는 A와 B를 포함하는 모든 조인 튜플과 A와 C를 포함하는 모든 조인 튜플들 중 D를 제외한 나머지 조인 튜플들을 찾으라는 것을 의미하며 이때 모든 연산자들의 우선순위

〈표 1〉 논리 연산자

표 현 식	의 미
∧ : AND 연산자	이항 연산자 이며 키워드를 모두 포함하는 결과를 출력(예: a∧b a, b 모두를 포함하는 결과를 출력)
∨ : OR 연산자	이항 연산자 이며 키워드 중 하나 또는 둘 모두를 포함하는 결과를 출력(예: a∨b 질의 결과가 "a", "b", "a,b" 와 같이 출력)
! : NOT 연산자	단항 연산자 이며 키워드를 포함하지 않는 결과를 출력(예: a∧!b a를 포함하지만 b는 포함하지 않는 결과를 출력



(그림 6) 입력 키워드의 분류



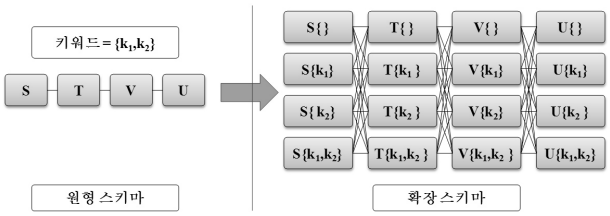
(그림 7) 고급 키워드 검색 질의 예제

는 동일한 것으로 간주한다. (따라서 위의 예에서는 괄호를 이용하여 우선순위를 처리하였다.) 위와 같이 단순한 연산자를 사용함으로서 사용자는 기존에 비해 더욱 표현력 있는 질의를 요청하는 것이 가능하다. 다음은 한 운송 전문 업체에서 사용하는 화물운송 제어 시스템(Commercial Vehicle Operations)을 통해 회사에서 끊임없이 유입되는 운송요청에 대해 본 논문에서 제안하는 고급 키워드 검색시스템을 통해 키워드 검색을 진행하는 예를 보여준다.

(그림 6)에서 검색 질의가 가지는 의미는 최근 10분 동안 도착한 배송 주문 중 e-bay에서 운송요청 된 모든 책과 노트북에 대한 항목을 검색하고 그 중 배송지가 서울인 항목을 제외한 결과를 출력하라는 것이다. 여기서 10분은 위의 질의가 연속질의로 시스템 내에 배치(deploy)될 때 슬라이딩 윈도우의 사이즈를 결정하게 된다. 입력된 질의는 두 개의 키워드들로 구분된다. 하나는 실행 키워드(executive keywords)로써 AND연산자에 의해 결합될 수 있는 키워드들을 의미하고 다른 하나는 선택적 키워드(optional keywords)로써 NOT연산자의 우측에 결합된 하나의 키워드와 슬라이딩 사이즈의 크기를 결정짓는 키워드로써 이들은 키워드 검색이 실행되는 범위를 결정짓는 키워드들이다. (그림 7)은 입력된 질의가 세 개의 실행 키워드와 두 개의 선택적 키워드로 분류된 예이다. 입력된 키워드들과 연산자들은 어휘 분석기(Lexical Analyzer)에 의해 토큰들로 변환되고 이러한 토큰들은 구문 분석기(Syntax Analyzer)에 입력되어 문맥 트리를 만들어 내게 된다. 이러한 과정을 통해서 입력된 소스 문자열은 실행 키워드와 선택적 키워드들로 분류된다.

3.2 CN 생성기

본 논문에서 제안하는 모든 CN들을 생성하는 방법은 관계기반 기법에서 널리 사용되는 확장 스키마(expanded schema)를 탐색하여 CN을 찾아내는 방식이다. 여기서 확장 스키마는 원형 스키마(original schema)에 따라서 각 릴레이션들에 사용자 질의 키워드들이 포함되어 있다고 가정하고 이들 키워드들의 부분집합으로부터 가능한 모든 관계를 나



(그림 8) 확장 스키마의 예

타내는 스키마이다. 확장 스키마의 예는 (그림 8)과 같다. 이러한 확장 스키마에서의 탐색이 필요한 이유는 앞서 2.2에서 기술한 바대로 동적인 스트림 데이터에 대한 연속 질의는 특정 시점에서 반복적으로 질의 결과를 수집해야 되기 때문에 초기 CN 생성의 오버헤드를 줄이기 위해서 가능한 모든 CN을 미리 생성해 놓게 된다. 각 CN으로부터 클러스터를구성한 후에 실행 계획에 의해서 반복적으로 수행하는 것이 동적인 스트림 데이터에서는 더 효율적으로 질의 처리를 수행할 수 있다.

또한 본 논문에서 제안한 고급 키워드 검색 시스템에서의 질의문은 기존 시스템이 제공하는 AND 연산자에 더해 OR 연산자와 NOT연산자를 포함하므로 이에 대한 처리가 가능한 새로운 CN 생성 알고리즘이 필요하다.

(그림 9)는 OR 및 NOT 연산자의 처리를 위해 확장된 CN 생성 알고리즘을 간략하게 나타낸다. 앞서 3.1에서 기술한 대로 사용자가 입력한 질의는 여러 개의 실행 키워드들 및 하나의 선택적 키워드로 분류된다. 각각의 실행 키워드는 하나 이상의 단일 키워드들이 AND 연산자에 의해 결합되어 있는 형태이며 이러한 단일 키워드들 중 모든 실행 키워드들에 공통적으로 포함되는 단일 키워드가 하나 이상 존재하며 이를 루트 키워드(root keywords)로 정의한다. 예를 들면 (그림 7)에서 세 개의 실행 키워드들에 공통적으로 포함되는 루트 키워드는 “ebay” 이다. 또한 실행 키워드들 중 가장 많은 단일 키워드를 포함하고 있는 실행 키워드, 다시 말하면 모든 실행 키워드들이 가진 단일 키워드를 모두 포함하고 있는 실행 키워드를 루트 실행 키워드(root executive keywords)로 정의한다. (그림 7)에서 루트 실행 키워드는 ebay ∧ book ∧ laptop 이다.

[정의 3] (루트 키워드): 루트 키워드를 RK, 실행 키워드를 EKi라고 하면, 다음과 같은 식을 만족하는 RK를 루트 키워드라 한다.

$$RK = EK_1 \cap EK_2 \cap \ldots \cap EK_n$$

[정의 4] (루트 실행 키워드) : 루트 실행 키워드를

InitCNGen(Expanded Schema *E*, Executive Keywords *EKs*, Optional Keywords *OKs*)
1. Initialize Root Keywords *RKs*
2. For all k_i of *RKs*
3. For all nodes n_{root} containing k_i
4. *CNGenerator*(*E*, n_{root})
5. Remove n_{root} from *E*

CNGenerator(Expanded Schema *E*, Node n_{root})
1. Initialize queue *q*
2. Construct a tree t_{first} consisting of a single node n_{root}
3. Insert t_{first} into *q*
4. While(*q* is not empty)
5. Tree t_{old} = *q.first*
6. For all node n_{new} in *E* that can *legally* be added to a node n_{old} of t_{old}
7. Create a new tree t_{new} by adding n_{new} as a child of n_{old}
8. if(t_{new} is a CN of root executive keywords)
9. Output t_{new}
10. Break
11. else if(t_{new} is a CN && $|t_{new}|$ reaches T_{max})
12. Output t_{new}
13. Break
14. else if(t_{new} has the potential of becoming a CN)
15. Insert t_{new} in *q*

(그림 9) CN 생성 알고리즘

REK, 실행 키워드를 EKi라고 하면, 다음 식을 만족하는 REK를 루트 실행 키워드라 한다.

$$REK = EK_1 \cup EK_2 \cup \ldots \cup EK_n$$

InitCNGen 알고리즘은 확장 스키마 E, 실행 키워드 EKs, 선택적 키워드 OKs를 입력으로 하며 먼저 EKs로부터 루트 키워드 RKs를 초기화한 뒤 RKs의 모든 단일 키워드들에 대해 이를 포함하는 확장 스키마 E의 모든 노드들에 대해 각각 CNGenerator 알고리즘을 수행하고, 수행이 끝난 뒤에는 해당 노드를 E로부터 제거한다.

CNGenerator에서는 입력된 노드를 루트로 하는 트리를 구성하고 이를 큐에 삽입한 뒤 아래와 같은 과정을 큐가 빌 때까지 반복한다.

① 큐로부터 n_{root}를 루트 노드로 하는 트리 t_{old} 를 구성한다.
② 확장 스키마 E로부터 생성되는 모든 노드 n_{new}를 t_{old}에 추가하여 t_{new} 를 생성한다.
③ t_{new}가 루트 실행 키워드에 대한 CN이면, 즉 루트 실행 키워드에 속한 모든 단일 키워드를 가진다면 이를 결과 버퍼로 출력한다.
④ t_{new}가 CN이며, t_{new}의 노드 수가 주어진 T_{max}와 같다면 이를 결과 버퍼로 출력한다. 여기서 T_{max}는 CN이 무한히 커지는 것을 방지하기 위한 임의의 한계치이다.
⑤ t_{new}에 노드를 추가했을 때 CN이 될 가능성이 있다면 큐에 삽입한다.

확장 스키마 E 로부터 새로운 노드 n_{new} 를 트리에 추가할 때 CN의 중복 생성을 피하기 위해 특별한 기법이 필요하다. 이는 E로부터 구성되는 모든 노드 n_{new} 에 완전 순서 식별자(total ordering id)를 부여하여 무조건적인 노드의 추가를 제한하는 것이다. 즉, 트리에 대한 노드의 추가는 다음과 같은 규칙을 준수해야(legally) 한다.

규칙 1 : 트리에 대한 모든 새로운 노드의 추가는 트리의 가장 오른쪽 리프 노드로부터 루트 노드에 이르는 경로 상의 노드들에 대해서만 이루어진다.
규칙 2 : 새롭게 추가되는 노드 식별자는 이미 존재하는 모든 형제 노드의 식별자보다 커야 한다.
규칙 3 : 선택적 키워드에 속한 단일 키워드, 즉 최초 질의에서 NOT 연산자의 오른쪽에 결합된 키워드가 포함된 노드는 트리에 추가하지 않는다.

규칙 1, 2는 기존 연구[3]로부터 제안된 기법이며, 이를 통해 CN의 중복 생성을 막을 수 있다. 또한 이러한 규칙을 통해 가장 오른쪽 리프 노드가 아닌 리프 노드가 유일한 키워드를 갖지 않으면 해당 트리가 CN이 될 가능성이 없음을 알 수 있다. 이는 가장 오른쪽 리프 노드가 아닌 리프 노드는 더 이상 확장이 불가능하기 때문이다. 규칙 3은 본 논문에서 제안한 고급 키워드 검색의 연산자 정의에 의한다. 단, 본 알고리즘에서의 선택적 키워드에는 슬라이딩 윈도우의 크기는 포함되지 않는다.

4. 계층적 클러스터링을 통한 질의 최적화

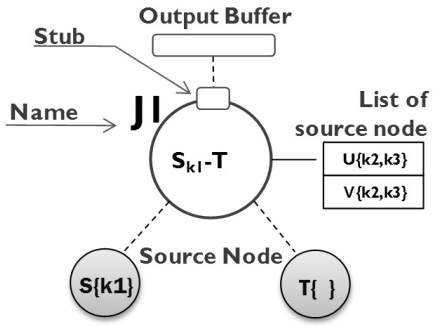
이 장에서는 우리는 새로운 질의 최적화 기법을 제안한다. 먼저 4.1절에서 CN에 대한 계층적 클러스터링에 대하여 설명하고, 4.2절에서 이러한 클러스터링을 통한 질의 최적화 기법에 대하여 보다 자세히 기술한다.

4.1 계층적 클러스터링

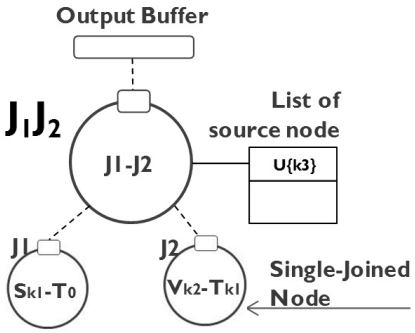
CN의 각 노드들을 조인하여 키워드 검색의 결과인 MTJNT를 추출하는 관계기반 기법에서는 많은 수의 조인 발생한다. 이러한 조인에서 동일한 소스 노드들을 조인하는 연산이 개별적, 반복적으로 수행 된다면 CPU와 메모리 등의 시스템 리소스를 중복적으로 소모하여 많은 비용을 초래한다. 따라서 이러한 중복적인 조인연산 수행을 중간결과 공유를 통해 회피하도록 하여야 한다. 하지만 이전연구에서 제안된 Operator Mesh는 동일한 루트노드를 가진 CN들을 하나로 클러스터링 함으로서 오직 제한적인 중간결과 공유만을 제공한다는 단점을 가지고 있다. 따라서 우리는 CN들을 계층적으로 클러스터링 하고 이를 통해 모든 CN들 사이에 중간결과를 공유하여 위와 같은 문제를 효율적으로 처리할 수 있는 기법을 제안한다.

(그림 10)은 계층적 클러스터링을 위해 새롭게 제안하는 단일-조인 노드(Single-join node)를 나타내고 있다. 이 단일-조인 노드는 두개의 소스노드에 대한 조인임과 동시에 같은 중간결과를 공유하는 CN의 클러스터이다. 자세히 말하자면, (그림 10)에서 소스노드 $S\{k_1\}$과 $T\{\}$의 조인결과는 Output Buffer에 저장되며, 저장된 결과는 다른 단일 노드들과 조인에 재사용될 수 있다. 예를 들면, $S\{k_1\}$-$T\{\}$-$U\{k_2,k_3\}$와 $S\{k_1\}$-$T\{\}$-$V\{k_2,k_3\}$라는 CN의 경우 하나의 J1 단일-조인 노드에 의해 $S\{k_1\}$-$T\{\}$를 조인한 중간결과를 공유하게 된다. 따라서, 두 CN은 그림과 같이 하나의 단일-조인 노드에 의해서 표현 되고, J1은 $S\{k_1\}$-$T\{\}$의 조인을 나타냄과 동시에 $S\{k_1\}$-$T\{\}$의 조인 결과값을 공유하는 세 개의 소스노드로 구성된 CN들의 클러스터가 된다. 이러한 단일-조인 노드들은 계층구조에서 최하위 계층(Level1)을 이루는 요소이다. 따라서, 단일-조인 노드는 그림과 같이 자신과 조인되는 소스노드들의 리스트를 가지게 된다. 만약 단일-조인 노드 자신이 CN이라면 리스트를 가지지 않는다. 또한, 해당 노드의 실행 여부를 결정하는 조건들을 Stub이라는 돌출부에 저장하게 된다.

(그림 11)은 내부-조인 노드(Internal-join node)를 나타낸다. 내부-조인 노드란 중간 혹은 최상의 계층에서 실행되는 노드를 포괄적으로 표현하는 노드이다. 그림에서 내부-조인 노드의 자식노드 J1과 J2는 단일-조인 노드이다. 따라서, 내부-조인 노드 J1J2는 Level2에 위치함을 알 수 있다. 하지



(그림 10) 단일 조인 노드



(그림 11) 내부 조인 노드

만, 이는 각 계층의 노드들은 이전 계층의 노드 두개를 자식노드로 가지게 된다는 것은 아니다. 단일-조인 노드로 구성된 최하위 계층을 제외한 모든 상위계층의 노드들은 이전 계층의 한 개의 노드와 한 개의 단일-조인 노드를 자식노드로 가지게 된다.

계층적 클러스터링은 위에서 소개된 단일-조인 노드와 내부-조인 노드로 구성된다. 계층의 최대 높이는 CN을 생성할 때 사용하는 파라미터 T_{max}의 값에 의해 결정된다. T_{max}는 CN이 가질 수 있는 최대 노드의 개수이다. 이는 모든 검색 키워드들을 모두 포함하는 MTJNT라 하더라도 그 조인의 횟수가 많다면 키워드들간의 연관도가 떨어져 사용자에게 의미 없는 결과일 수 있기 때문 파라미터 T_{max}를 두어 조인되는 노드의 최대 개수를 제한하는 것이다. 따라서, 계층의 최대 높이는 $\lfloor T_{max}/2 \rfloor$가 된다. 또한, 모든 계층에서 MTJNT를 생성해 낼 수도 있다.

4.2 질의 최적화

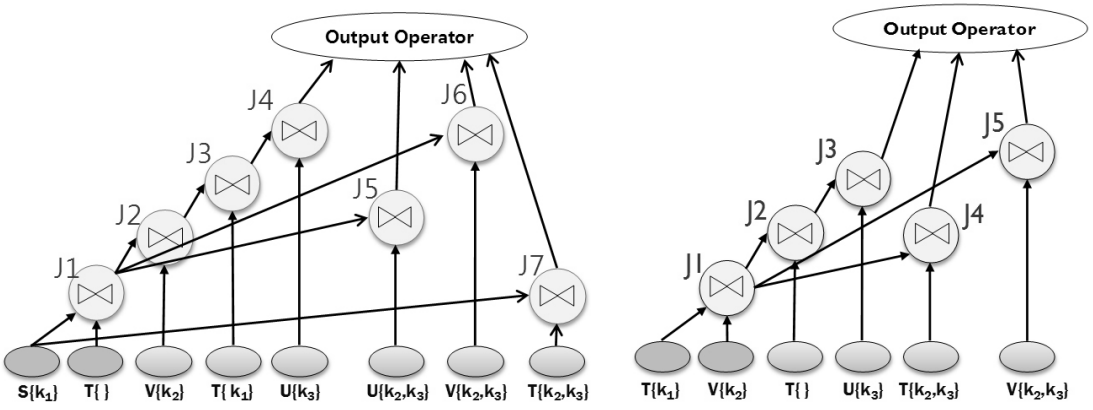
계층적 클러스터링을 통해 모든 CN들은 공유자원에 따라 클러스터링 될 수 있다. 예를 들어, (그림 2)의 예제 스키마와 데이터 튜플들과 같은 상황에서 키워드 검색이 진행 되고 스키마에 정의된 관계는 N:M 관계를 허용한다고 가정한다. 먼저 $S\{k_1\}$와 $T\{k_1\}$노드로부터 만들어 질 수 있는 CN이 다음과 같다고 할 때,

① $S\{k_1\}$ - $T\{\ \}$ - $V\{k_2\}$ - $T\{k_1\}$ - $U\{k_3\}$
② $S\{k_1\}$ - $T\{\ \}$ - $U\{k_2,k_3\}$
③ $S\{k_1\}$ - $T\{\ \}$ - $V\{k_2,k_3\}$
④ $S\{k_1\}$ - $T\{k_2,k_3\}$
⑤ $T\{k_1\}$ - $V\{k_2\}$ - $T\{\ \}$ - $U\{k_3\}$
⑥ $T\{k_1\}$ - $V\{k_2\}$ - $T\{k_2,k_3\}$
⑦ $T\{k_1\}$ - $V\{k_2,k_3\}$

기존 Markowetz가 제안한 Operator Mesh는 (그림 12)와 같이 노드 $S\{k_1\}$과 $T\{k_1\}$을 루트로 가지는 두 개의 클러스터가 개별적으로 Mesh내에 생성된다. 아래 그림에서 보여지듯이 두 클러스터 사이의 중간결과 공유는 이루어질 수 없다. 따라서, CN ①과 ⑤,⑥사이에 공유될 수 있는 $T\{k_1\}$-$V\{k_2\}$ 조인연산은 공유되지 못한다.

(그림 13)-(a) 는 본 논문에서 우리가 제안하는 계층적 클러스터링 기법을 사용하여 위의 CN들을 통합하는 과정을 나타낸다. STEP 1는 CN ①을 실행트리로 변경한 모습, STEP 2는 STEP 1이후 CN ②와 ③을 추가한 모습이다. STEP 3은 CN ④를 추가한 모습으로 (그림 12)의 첫 번째 그림에서 S{k1}을 루트로 하는 클러스터와 같이 4개의 CN을 통합한 모습을 나타내고 있다.

STEP 2에서 새로운 조인노드의 추가 없이 J1 단일-조인 노드의 리스트에 U{k2,k3}, V{k2,k3}를 추가 해 줌으로서



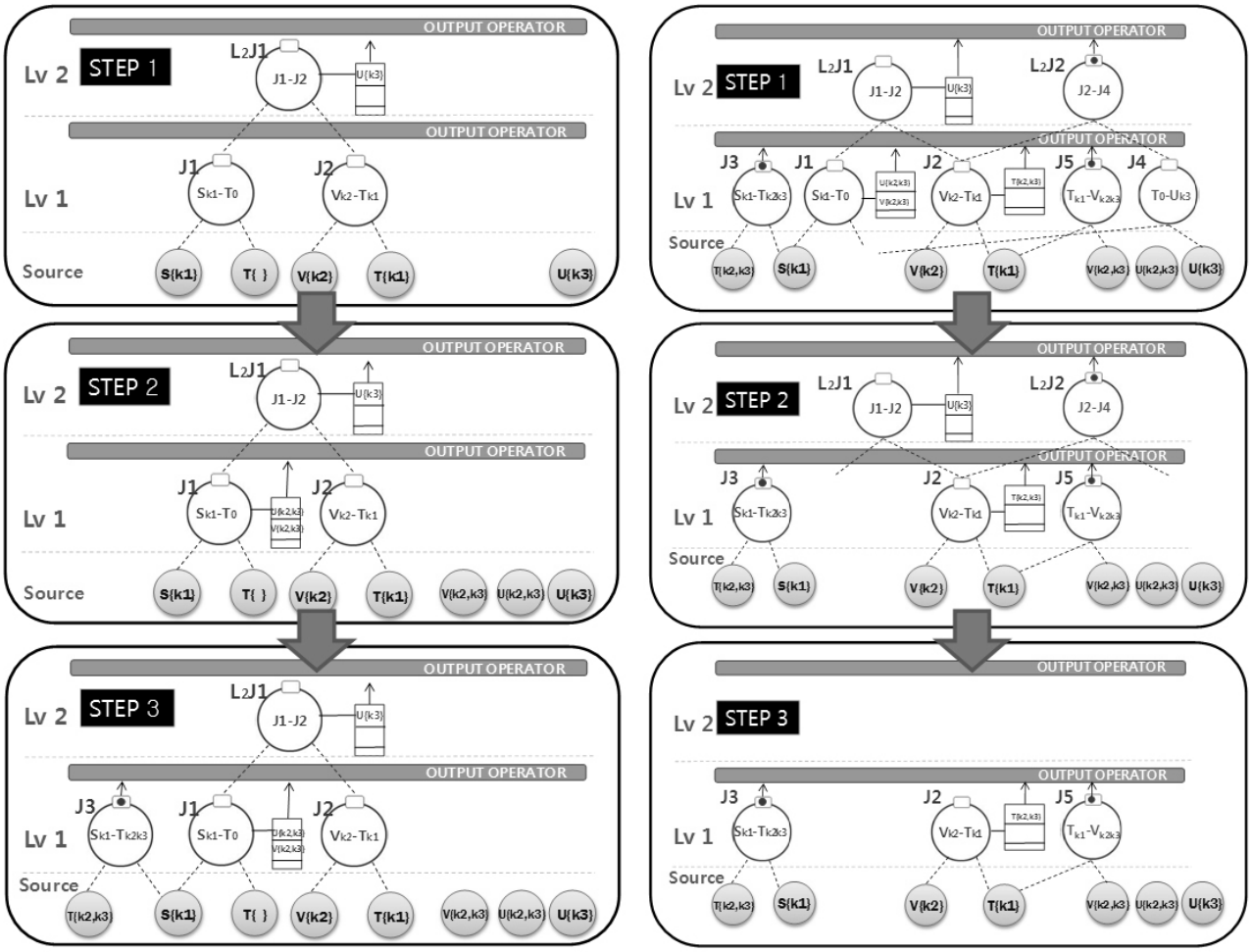
(그림 12) Operator Mesh 내의 클러스터

CN ②, ③을 삽입하였다. 또한, STEP 3에서 새롭게 추가된 단일-조인 노드 J3는 그 연산결과가 MTJNT를 생성하기에 리스트와 Output Buffer없이 그 수행 결과를 Output Operator에 넘겨준다. 여기서, Output Operator는 각 계층에서 생성되는 MTJNT를 수집하는 연산을 수행한다. (그림 14)는 T{k1}으로 시작되는 모든 CN을 (그림 13)-(a)의 마지막 단계에 추가하여 완성된 계층적 클러스터의 모습을 나타낸다. J2의 조인 연산은 CN ⑤, ⑥ 에 의해 공유되고 있음을 알 수 있다.

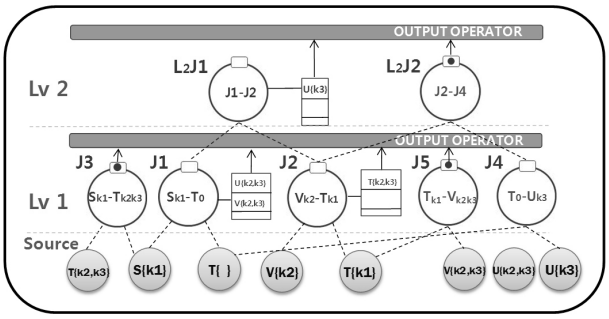
(그림 13)-(b) 는 통합된 CN중 MTJNT를 생성할 수 없는 CN들을 가치지기(pruning) 하는 과정을 보여준다. 만약 테이블 T의 모든 튜플이 입력된 키워드를 포함하고 있다고 가정하면, 소스 노드 T{ }의 selection 연산의 수행결과 튜플은 존재하지 않는다. 따라서, 소스 노드 T{ }를 포함하고 있는 모든 CN들은 MTJNT를 생성할 수 없기에 실행되어서는 안 된다.

STEP 1에서 결과를 내지 못하는 소스 노드 T{ }가 삭제된다. STEP 2에서 소스 노드 T{ } 를 자식 노드로 가지는 조인 노드들은 삭제된다. 즉, 소스 노드 T{ }와 간선으로 연결된 단일-조인 노드 J1, J4는 삭제된다. 그리고 J1, J4와 간선으로 연결된 계층2의 내부-조인 노드L2J1과 L2J2도 삭제된다. 결과적으로, 소스 노드T{ }를 포함하는 CN ①, ②, ③, ⑤는 가지치기되고 CN ④, ⑥, ⑦에 대한 노드들만이 남게 된다. 소스 노드뿐 아니라 조인연산에 의해 아무런 결과도 만들지 못하는 조인 노드들도 가지치기 될 수 있다. 예들어 만약 (그림 14)와 같이 통합된 CN들 중 V{k2}-T{k1}조인을 수행하는 단일-조인 노드 J2가 아무런 결과도 만들지 못한다고 가정하면 J2는 즉시 가지치기 된다. J2를 자식 노드로 가지는 L2J1과 L2J2도 삭제되며, L2J2의 삭제로 인해 결과를 만들어 내더라도 그 결과가 사용되지 않는 단일-조인 노드 J4 도 삭제된다.

여기서, 실제로 모든 노드들과 간선들이 삭제되는 것이 아니라 간선들은 비활성간선(dead edge)으로 표시되고, 노드들은 Stub에 저장되는 실행조건 Flag를 OFF로 설정된다. 따라서 연속질의 반복실행에 있어 슬라이딩 윈도우 내에 존재하는 데이터에 변화에 따라 조인 노드들이 결과를 갖게 된다면 가지치기된 간선들은 다시 활성간선(alive edge)으로 노드들의 실행Flag는 ON으로 변경될 수 있다.



(그림 13) (a) CN ①②③④ 통합과정 (b) 클러스터에서 가지치기

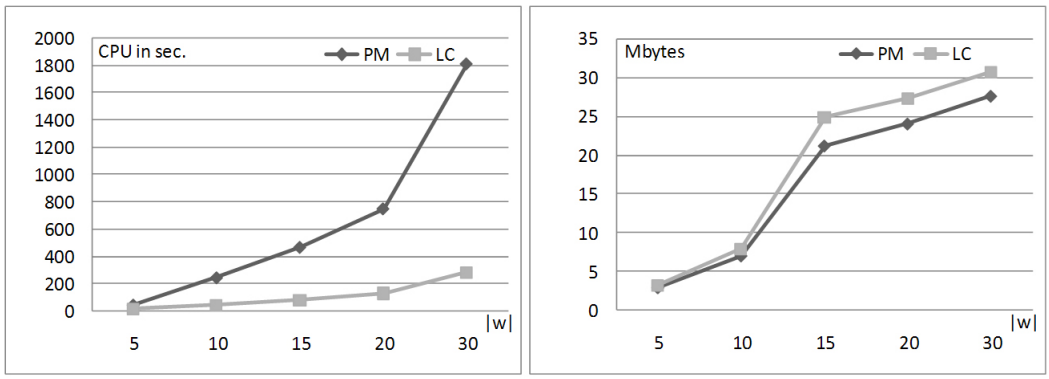


(그림 14) 완성된 계층적 클러스터

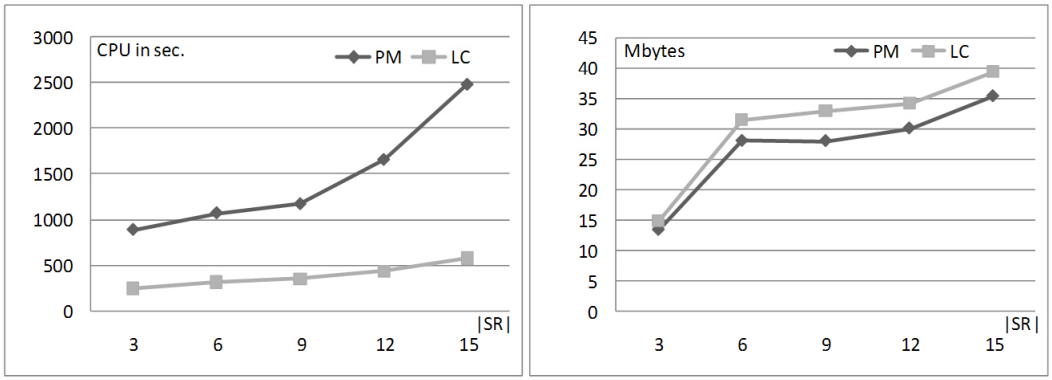
5. 실험 평가

이 장에서는 본 논문에서 제안한 고급 키워드 검색 기법의 성능을 평가한다. 실험에 사용된 시스템 환경은 Xeon 64bit 듀얼(dual) 2.0GHz CPU에 3GB RAM의 서버이며 MySQL 5.1을 사용하여 수행되었다. 모든 스트리밍 데이터는 데이터베이스에 존재하는 테이블에 저장될 수 있다고 가정하였으며 이들 각 테이블에는 각각 세 개의 속성(attribute)가 존재한다. 이들 중 두 개는 주 키 값과 이 테이블과 선택적으로 조인 될 수 있는 다른 테이블의 외래 키 값을 저장하기 위한 것이고 나머지 하나의 속성은 텍스트 데이터가 담겨진다. 이 텍스트 데이터에는 랜덤하게 하나 이상의 키워드가 저장된다. 위와 같이 구성된 데이터베이스의 모든 테이블에는 초당 1개의 데이터가 새롭게 추가된다. 측정된 성능의 객관적인 평가를 위해 Markowetz가 제안한 Partial-Mesh(PM) 알고리즘[3]을 구현하여 본 논문에서 제안한 계층적 클러스터링 (Layered - Clustering, LC)과 비교하였다.

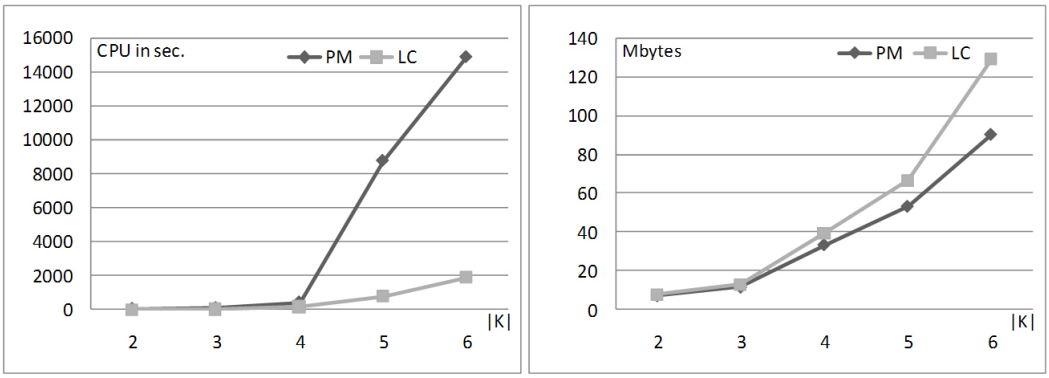
실험은 크게 세 가지로 나누어 수행하였다. 첫 번째는 슬라이딩 윈도우의 증가에 따른 성능의 측정이며, 두 번째는 테이블(stream relation)의 증가에 따른 성능 측정, 그리고 마지막으로 질의 키워드의 증가에 따른 성능 측정이다. 각각의 실험에서 성능 평가를 위한 척도는 두 가지이다. 첫 번째는 질의 처리 수행 시간을 나타내는 CPU 비용(cost)이며. 두 번째는 본 논문에서 제안된 질의 최적화의 효과를 확인하기 시스템의 메모리 사용량이다. 결과를 살펴보면 먼저 슬라이딩 윈도우(그림 15)의 경우 CPU 비용 및 메모리 사용량 모두 Partial-Mesh에 비해 우수한 성능을 나타내는 것을 확인 할 수 있는데 특히 CPU 비용의 경우 슬라이딩 윈도우의 크기가 증가 할수록 성능의 차이는 급격히 벌어지는 것을 볼 수 있다. 이는 더욱 많은 중간 결과의 공유가 보여주는 이점을 단적으로 나타내는 것이라 할 수 있다. 다음으로는 실험에 사용한 테이블의 증가에 따른 비교인데(그림 16) 앞에서와 마찬가지로 Partial-Mesh에 대해 계층적



(그림 15) 슬라이딩 윈도우의 증가



(그림 16) 테이블의 증가



(그림 17) 키워드의 증가

클러스터링이 더욱 우수한 성능을 나타내며 이는 테이블의 개수가 증가 할수록 더욱 두드러지게 나타난다. 마지막은 키워드의 증가에 따른 성능비교로써(그림 17) 마찬가지로 계층적 클러스터링이 상대적으로 더욱 우수하게 나타난다. 여기에서 특히 두드러지게 나타나는 현상은 키워드의 수가 일정 수 이상으로 증가하면 성능의 하락폭 - 즉, 소모되는 CPU 비용 - 이 기하급수적으로 증가한다는 점이다. Partial-Mesh의 경우 키워드의 개수가 5이상일 때부터 이러한 현상이 나타나며 6보다 커질 경우 실험에 사용된 시스템으로는 측정이 어려울 정도로 성능의 하락이 두드러졌다. 비교가 불가능하여 그림으로는 나타내지 않았으나 계층적 클러스터링의 경우에도 키워드의 개수를 더욱 증가시킬 경우 이러한 급격한 성능의 하락은 동일하게 나타났다. 이러한 현상은 슬라이딩 윈도우나 테이블의 증가에 비해 키워드의 증가가 생성되는 CN의 개수의 증가에 미치는 영향이 더욱 크기 때문인 것으로 보인다.

각각의 실험 결과에서 나타나듯이 CPU 비용의 경우, 모든 실험에 대해 본 논문에서 제안한 계층적 클러스터링이 Partial-Mesh에 비해 우수한 성능을 보이는 것을 알 수 있다. 그러나 메모리 소모의 경우 오히려 Partial-Mesh에 비해 약 8~10% 이상 많은 것으로 나타나는데 이는 계층적 클러스터링이 Partial-Mesh에 비해 더 많은 중간 결과를 공유하기 때문이며 결과적으로는 이로 인해 더욱 빠른 속도로 질의 처리가 가능해 진 것임을 확인 할 수 있다.

6. 결 론

최근 사용자에게 편리성을 제공하는 키워드 검색 패러다임은 많은 검색분야에서 그 적용이 폭넓게 이루어지고 있다. 우리는 슬라이딩 윈도우내에서 시간에 따라 지속적으로 변화하는 데이터 스트림에서 키워드 검색 수행에 있어 시스템 리소스의 소모를 줄일 수 있는 계층적 클러스터링을 활용한 질의 최적화 기법을 제안하였고 실험을 통해 그 성능을 평가하였다. 이 기법은 빈번하게 변화하는 데이터에 따라 불필요한 조인연산의 실행을 회피시킬 수 있으며, 중간 결과 공유를 통해 중복적인 조인연산 처리로 인한 CPU와 메모리의 낭비를 줄일 수 있다. 또한, 고급 키워드 검색 시스템을 제한하여 사용자가 의미를 부여한 질의를 생성할 수 있도록 하였고 이를 통해 사용자는 보다 가치 있는 결과를 획득할 수 있게 되었다. 향후 연구로 Top-k 랭킹 기법을 적용하고 연속질의의 배치와 실행에 중요한 질의 실행 계획을 추가하여 본논문에서 제안된 키워드 검색 시스템을 향상 시키고자 한다.

참 고 문 헌

[1] W. Wang, X. Lin and Y. Luo, Keyword Search on Relational Databases, In IFIP, 2007.

[2] B. Babcock, S. Babu, M. Datar, R. Motwani and J. Widom, Models and Issues in Data Stream Systems, In Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, pages 253-264, 2003.

[3] A. Markowetz, Y. Yang, D. Papadias, Keyword Search on Relational Data Streams,In ACM SIGMOD, pages 605-616, 2007.

[4] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar, Bidirectional Expansion for Keyword Search on Graph Databases. In VLDB, pages 505-516, 2005.

[5] S. Dar, G. Entin, S. Geva, and E. Palmon, DTL's DataSpot: Database Exploration Using Plain Language. In VLDB, pages 645-649, 1998.

[6] B. Aditya, G. Bhalotia, S. Chakrabarti, A. Hulgeri, C. Nakhe, Parag, and S. Sudarshan. BANKS: Browsing and Keyword Searching in Relational Database. In VLDB, pages 1083-1086, 2002.

[7] V. Hristidis and Y. Papakonstantinou, DISCOVER: Keyword Search in Relational Database, In VLDB, pages 670-681, 2002.

[8] V. Hristidis, L. Gravano, and Y. Papakonstantinou, Efficient IR-style Keyword Search over Relational Databases, In VLDB, pages 850-861, 2003.

[9] F. Liu, C.T. Yu, W. Wang, and A. Chowdhury, Effective

Keyword Search in Relational Database, In SIGMOD, pages 563-574, 2006.
[10] B. Yu, G. Li, and K. Sollins, Effective Keyword-based Selection of Relational Database, In ACM SIGMOD International Conference on Management of Data, pages 139-150, 2007.
[11] M. Sayyadian, A. Doan, and L. Gravano, Efficient Keyword Search over Heterogeneous Relational Databases, In Proceedings of IEEE 23rd International Conference on Data Engineering, pages 346-355 2007.
[12] S. Wang, Z. Peng, J. Zhang, J. X. Yu, and B. Ding, NUITS: A Novel User Interface for Efficient Keyword Search over Databases, In VLDB, pages 1143-1146, 2006.

주 진 웅
e-mail : roeknine@gmail.com
2009년 한양대학교 컴퓨터공학부(학사)
2009년~현 재 한양대학교 컴퓨터공학과 석사과정
관심분야 : DB시스템, 정보 검색 및 DB 시스템 응용, 데이터마이닝

김 학 수
e-mail : hsookim@hanyang.ac.kr
2004년 한양대학교 컴퓨터공학과(석사)
2006년~현 재 한양대학교 컴퓨터공학과 박사과정
관심분야 : 데이터베이스, 시맨틱 마이닝, 온톨로지

황 진 호
e-mail : jhhwang@cse.hanyang.ac.kr
2006년 상지대학교 컴퓨터공학부(학사)
2006년 한양대학교 컴퓨터공학과(석사)
2009년~현 재 LG전자 MC 연구소
관심분야 : Mobile, 데이터베이스, 임베디드 시스템, 파일 시스템

손 진 현
e-mail : jhson@hanyang.ac.kr
1998년 한국과학기술원 전산학과(석사)
2001년 한국과학기술원 전자전산학과(박사)
2002년 한양대학교 컴퓨터공학과 조교수
2008년~현 재 한양대학교 컴퓨터공학과 부교수
관심분야 : 데이터베이스, e-비즈니스, 유비쿼터스 컴퓨팅, 임베디드 시스템