관계형 데이터 스트림에서 키워드 검색을 위한 질의 최적화[1]

황진호*, 김학수*, 김종진*, 이승미**, 손진현*
*한양대학교 컴퓨터공학과
**한양대학교 컴퓨터공학과 B21 AIS 사업팀
e-mail : {jhhwang, hskim, jjkim, smlee}@database.hanyang.ac.kr jhson@hanyang.ac.kr

Query Optimization for Keyword Search on Relational Data Stream

Jin-Ho Hwang*, Hak Soo Kim*, Jhong-Jin Kim*, Seung Mi Lee**, Jin Hyun Son*
*Department of Computer Science and Engineering, Hanyang University, Ansan
**BK21 AIS Team, Hanyang University

요 약

최근 관계형 데이터 스트림에서 키워드 검색에 관한 연구가 진행되고 있다. 키워드 검색을 통해 사용자는 시스템의 복잡한 내부 데이터 스키마나 질의언어에 대한 지식이 없이도 데이터 스트림에서 정보 검색이 가능하다. 하지만, 빈번하고 동적으로 변화하는 특성을 지닌 데이터 스트림에서 수행되는 연속질의 처리를 위해서 보다 효과적인 질의 최적화 방안이 요구된다. 따라서, 우리는 본 논문을 통해 계층적 클러스터링을 이용하여 중간결과 공유의 최대화를 통한 질의 최적화 방안을 제안한다.

1. 서론

최근 사용자가 데이터베이스 스키마와 질의언어에 대한 해박한 지식 없이도 원하는 정보를 검색할 수 있는 키워드 기반 검색 패러다임을 적용하고자 하는 연구가 활발히 진행되고 있다. 하지만 기존 관계형 데이터베이스에 거의 모든 시스템의 전사적 데이터가 저장되어 있음에도 이런 키워드 기반 검색은 국부적으로 적용되고 있다. 특히, 최근 그 활용이 증가되고 있는 데이터 스트림에서 키워드 기반 검색에 관한 연구는 초기 단계이며 스트림데이터의 특성을 반영한 질의 최적화에 대한 연구가 요구되고 있다.

우리는 본 논문에서 키워드 검색 질의에 대한 효과적인 처리로 시스템의 CPU와 메모리 부하를 줄일 수 있는 계층적 클러스터링(layered clustering)을 통한 질의 최적화 방안을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 키워드 검색에 관한 기존연구들에 대하여 살펴보고, 3장에서 본 논문에서 제안하는 계층적 클러스터링을 통한 질의 최적화 방안에 대하여 기술한다. 마지막으로, 4장에서 결론 및 향후 연구에 대하여 서술한다.

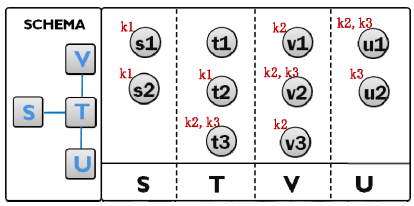
2. 관련연구

2.1 키워드 기반 검색 기법

기존 연구들을 통해 관계형 데이터베이스에 적용 가능한 다양한 키워드 기반 검색 기법들은 데이터 모델에 따라 크게 두 가지로 분류가 가능 한다.

첫째, 그래프-기반 기법 방식은 데이터베이스의 모든 데이터를 하나의 그래프로 표현하고, 그래프 안에서 사용자가 입력한 모든 키워드를 포함하고 있는 서브그래프를 찾는 방식이다[1][2][3]. 그래프기반 키워드 검색 기법은 소규모의 데이터베이스에서 적합하며 빠른 질의 처리 성능을 가지고 있다[4].

둘째, 관계-기반 기법은 질의 결과로 모든 키워드가 포함된 튜플(tuple)의 집합을 생성한다. 스키마에 정의된 관계를 기반으로 각 테이블의 튜플들을 조인하여 검색을 수행하는 방식이다[5][6]. 이러한 방식은 쉽게 관계형 데이터베이스 시스템에 적용이 가능하고 복잡한 쿼리도 처리가 가능하여 대용량 데이터베이스에도 그 적용이 용이하다 할 수 있다.



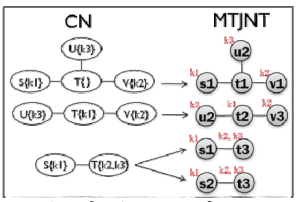
(그림 1) 예제 데이터 스키마와 튜플

그림1과 같은 데이터베이스에서 관계-기반 기법을

[1] 본 연구(논문)는 산업자원부 지원으로 수행하는 21세기 프론티어 연구개발사업(인간기능 생활지원 지능로봇 기술개발사업)의 일환으로 수행되었습니다. 이 논문은 2007년도 정부(과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임(No. R01-2007-000-20135-0).

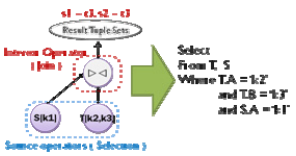
사용하여 사용자가 입력한 키워드 k_1, k_2, k_3를 검색하는 과정을 살펴보도록 하자. 그림1에서 각 테이블에 존재하는 튜플을 하나의 노드로 표현하고, 각 튜플이 포함하고 있는 키워드를 노드의 좌측 상단에 표시한다. 관계-기반 기법은 입력된 키워드들을 모두 포함하고 있는 튜플집합인 Minimal Total Join Network(MTJNT)를 추출하기 위해 Candidate Network(CN)들을 생성한다. 자세히 말해, 그림1에서 S와 T테이블이 관계를 가지므로, k_1키워드를 포함하는 튜플 s1와 s2는 튜플 t3와 조인될 수 있다. 이들이 조인 된다면 두 개의 튜플집합 s1-t3, s2-t3는 MTJNT가 된다.

아래 그림2는 그림1의 예제에서 추출될 수 있는 몇 개의 MTJNT와 CN의 관계를 표현하고 있다.



(그림 2) MTJNT와 CN

그림 3에서 S{k_1}-T{k_2,k_3}로 구성된 CN을 실행트리(operator tree)로 나타낸 것을 보여주고 있다. 실행 트리는 CN의 각 노드들을 리프노드로 가지며 중간 노드는 조인연산을 의미한다. 이러한 실행 트리를 실행함으로써 s1-t3, s2-t3와 같은 튜플집합을 추출할 수 있으며 아래의 SQL문을 실행 하는 것과 같은 결과를 이끌어 낼 수 있다.



(그림 3) Operator Tree와 SQL

2.2 데이터 스트림에서 키워드 검색

관계형 데이터 스트림에서 키워드 검색은 다음과 같은 사항들을 고려하여야 한다.

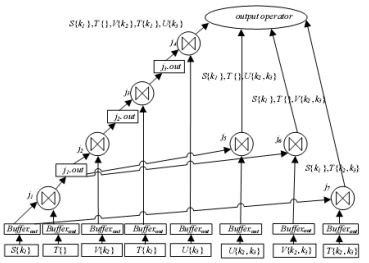
① 연속적으로 변화하는 데이터
② 연속질의 처리
③ 질의수행기간 동안 모든 실행트리 유지

관계형 데이터베이스에서의 키워드 검색은 데이터베이스에 존재하는 데이터들만을 고려하여 CN을 생성한다. 이와 달리 관계형 데이터 스트림에서는 슬라이딩 윈도우(sliding window)내의 연속적으로 변화하는 데이터를 처리하기 위하여 모든 출현 가능한 데이터를 고려하여 CN을 생성하여야 한다. 예를 들어, 그림1과 같은 데이터가 있을 때 윈도우 내에 현재 k_1 키워드를 포함하는 튜플 s1, s2가 존재 하더라도 k_2와 k_3키워드를 포함하는 새로운 튜플 s3가 추가되거나 기존 s1튜플이 삭제될 수 도 있다.

또한, 데이터 스트림의 특성에 의해 사용자가 입력한 키워드들을 검색하는 질의는 단일수행 질의(One-time Query)아닌 연속질의가 된다. 따라서, 생성된 CN으로 만들어진 실행트리들은 질의가 수행되는 동안(life-time) 시스템 내에 유지 되어야 한다.

이와 같은 데이터 스트림환경에서 키워드 검색을 수행하기 위한 고려사항들을 기존 연구[7]에서는 다음과 같이 처리하고 있다.

첫째, 모든 출현 가능한 데이터를 모두 고려하여 만들어진 많은 수의 CN중 중복을 제거하는 기법을 제안 하고 기존 CN Generator를 보완하였다. 둘째, CN으로부터 만들어지는 많은 수의 실행 트리들 사이에 중간결과(intermediate result)를 공유할 수 있도록 그림4와 같은 Operator Mesh를 제안하였다.



(그림 4) Operator Mesh

마지막으로, 결과값을 만들어 낼 수 없는 CN들을 한번에 가지치기(pruning)할 수 있는 Demand-driven operator execution 정책을 제안하였다.

그림 4는 Operator Mesh에 존재하는 $|SR|\cdot 2^{|k-1|}$개의 클러스터 중 S{k_1}노드를 루트노드로 하는 CN들의 클러스터이다. 따라서, 그림4에 나타난 클러스터는 S{k_1}노드가 아닌 다른 루트노드를 가지는 CN들로 이루어진 클러스터와 중간결과를 공유할 수 없다.

결과적으로, Operator Mesh에 존재하는 클러스터들 사이에 중간결과를 공유할 수 없다는 단점을 가지고 있다.

3. 계층적 클러스터링 을 통한 질의 최적화 기법

3.1 계층적 클러스터링

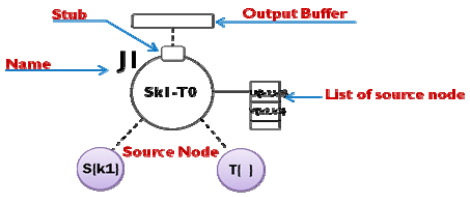
동일한 소스노드들을 조인하는 연산이 개별적, 반복적으로 수행 된다면 CPU와 메모리 등의 시스템 리소스를 중복적으로 소모하여 많은 비용을 초래한다. 따라서, 중복적인 조인연산 수행을 중간결과 공유를 통해 회피하도록 하여야 한다.

우리는 CN들을 계층적으로 클러스터링하고 이를 통해 모든 CN들 사이에 중간결과를 공유하여 위와 같은 문제를 효율적으로 처리할 수 있는 기법을 제안한다.

그림5는 계층적 클러스터링을 위해 새롭게 제안하는 단일-조인 노드(Single-join node)이다. 이 단일-조인 노드는 두개의 소스노드에 대한 조인임과 동시에 그 결과를 함께 공유하는 CN들의 클러스터이다.

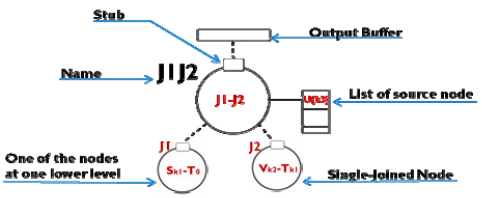
그림5에서 소스노드 S{k_1}과 T{}의 조인결과는

Output Buffer에 저장되며, 저장된 결과는 다른 단일 노드들과 조인에 재사용될 수 있다. 예를 들면, $S\{k_1\}$-$T\{\}$-$U\{k_2,k_3\}$와 $S\{k_1\}$-$T\{\}$-$V\{k_2,k_3\}$ 두 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 표현되고, J1은 $S\{k_1\}$-$T\{\}$의 조인을 나타냄과 동시에 $S\{k_1\}$-$T\{\}$의 조인 결과값을 공유하는 3개의 소스노드로 구성된 CN들의 클러스터가 된다. 이러한 단일-조인 노드들은 계층구조에서 최하위 계층(Level1)을 이루는 요소이다.



(그림 5) 단일-조인 노드

해당 노드의 실행 여부를 결정하는 조건들을 Stub이라는 돌출부에 담긴다.



(그림 6) 내부-조인 노드

위 그림6은 내부-조인 노드(Internal-join node)로 중간 혹은 최상의 계층에서 실행되는 노드를 포괄적으로 표현하는 노드이다. 그림에서 내부-조인 노드의 자식노드 J1과 J2는 단일-조인 노드이다. 따라서, J1J2 내부-조인 노드는 Level2에 위치함을 알 수 있다. 각 계층의 내부-조인 노드들은 이전 계층의 한 개의 노드와 한 개의 단일-조인 노드를 자식노드로 가지게 된다.

내부-조인 노드도 단일-조인 노드와 같이 조인 될 수 있는 소스노드의 리스트와 실행조건을 담고있는 Stub 그리고 Output Buffer를 가지고 있다. 위 그림 6은 $S\{k_1\}$-$T\{\}$-$V\{k_2\}$-$T\{k_1\}$-$U\{k_3\}$로 구성된 CN을 표현하고 있다.

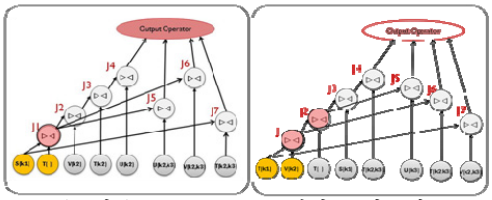
계층적 클러스터링은 위에서 소개된 단일-조인 노드와 내부-조인 노드로 구성된다. 계층의 최대 높이는 CN을 생성할 때 사용하는 파라미터 Tmax의 값에 의해 결정된다. Tmax는 CN이 가질 수 있는 최대 노드의 개수이다. 이는 모든 검색 키워드들을 모두 포함하는 MTJNT라 하더라도 그 조인의 횟수가 많다면 키워드들간의 연관도가 떨어져 사용자에게 의미 없는 결과일 수 있기 때문 파라미터 Tmax 를 두어 조인되는 노드의 최대 개수를 제한하는 것이다. 따라서, 계층의 최대 높이는 |Tmax/2|가 된다.

3.2 질의 최적화

계층적 클러스터링을 통해 모든 CN들은 공유자원에 따라 클러스터링 될 수 있다. 예를 들어, 그림1의 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,

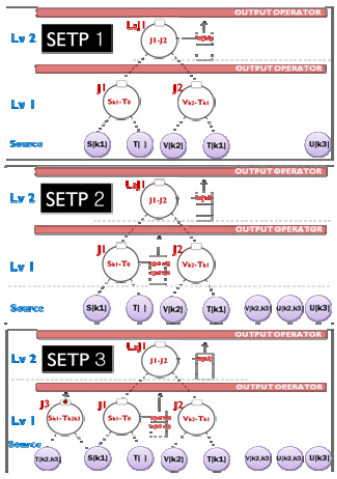
① S{k1} - T{} - V{k2} - T{k1} - U{k3}
② S{k1} - T{} - U{k2,k3}
③ S{k1} - T{} - V{k2,k3}
④ S{k1} - T{k2,k3}
⑤ T{k1} - V{k2} - T{} - S{k1} - U{K3}
⑥ T{k1} - V{k2} - T{} - U{k3}
⑦ T{k1} - V{k2} - T{k2,k3}
⑧ T{k1} - V{k2,k3}



(그림7) Operator Mesh내의 클러스터

기존 Alexander가 제안한 Operator Mesh는 그림7과 같은 $S\{k_1\}$와 $T\{k_1\}$노드를 루트로 가지는 두개의 클러스터가 개별적으로 Mesh내에 생성된다.

위 그림에서 보여지듯이 두 클러스터 사이의 중간 결과 공유는 이루어질 수 없다. 따라서, CN ①과 ⑤⑥⑦사이에 공유될 수 있는 $T\{k_1\}$-$V\{k_2\}$ 조인연산은 공유되지 못한다.

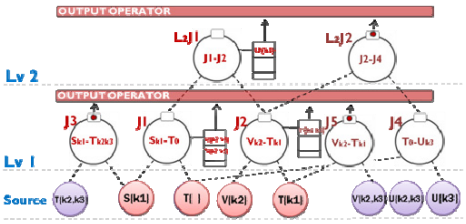


(그림 8) CN ①②③④통합 과정

그림 8은 본 논문에서 제안하는 계층적 클러스터링 기법을 사용하여 위의 CN들을 통합하는 과정을 나타낸다. STEP 1는 CN ① 을 실행트리로 변경한 모습, STEP 2는 STEP 1이후 CN ②와 ③을 추가한 모습이다. STEP 3은 CN ④를 추가한 모습으로 위 그림 7

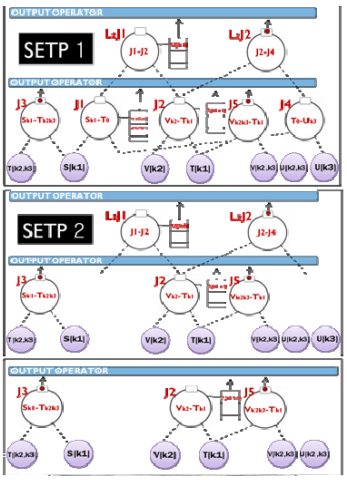
의 첫 번째 그림에서 S{k_1}을 루트로 하는 클러스터와 같이 4개의 CN을 통합한 모습을 나타내고 있다.

STEP 2에서 새로운 조인노드의 추가 없이 J1 단일-조인 노드의 리스트에 U{k_2,k_3}, V{k_2,k_3}를 추가 해 줌으로서 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 9) 완성된 계층적 클러스터링

그림 9는 T{k_1}으로 시작되는 모든 CN을 통합한 완성된 계층적 클러스터링의 모습을 나타낸다. J2의 조인 연산은 CN ⑤, ⑥, ⑦ 에 의해 공유되고 있음을 알 수 있다.



(그림 10) CN Pruning

위 그림10은 통합된 CN중 MTJNT를 생성할 수 없는 녀석들을 가치지기 하는 과정을 보여준다.

테이블 T의 모든 튜플이 입력된 키워드를 포함하고 있다면, 소스노드 T{}의 selection 연산의 수행결과 튜플은 존재하지 않는다. 따라서, T{}노드를 포함하고 있는 모든 CN들은 MTJNT를 생성할 수 없기에 실행되어서는 안 된다.

SETP 1에서 결과를 내지 못하는 T{}노드가 삭제된다. SETP 2에서 T{}노드를 자식노드로 가지는 조인 노드들이 삭제된다. 즉, 소스노드 T{}와 간선으로 연결된 J1, J4 단일-조인 노드는 삭제된다. 그리고 J1노드와 J4노드와 간선으로 연결된 L2J1노드와 L2J2노드도 삭제된다. 결과적으로, T{ }노드를 포함하는 CN ①, ②, ③, ⑤, ⑥은 가지치기되고 CN ④, ⑦, ⑧에 대한 노드들만이 남게 된다.

소스노드뿐 아니라 아무런 결과도 만들지 못하는 조인노드도 가지치기 될 수 있다. 예로 그림9와 같이 통합된 CN들 중 V{k_2}-T{k_1}조인을 수행하는 J2 조인노드가 아무런 결과도 만들지 못한다면 즉시 가지치기 된다. J2노드를 자식노드로 가지는 L2J1과 L2J2도 삭제되며, L2J2의 삭제로 결과를 만들어 내더라도 그 결과가 사용되지 않는 단일-조인 노드 J4도 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 있어서 슬라이딩 윈도우내에 존재하는 데이터에 따라 가지치기된 간선들은 다시 활성간선(alive dege)으로, 노드들의 실행Flag는 ON으로 변경될 수 있다.

4. 결론 및 향후 연구

우리는 데이터 스트림에서 키워드 검색 수행에 있어 시스템 리소스의 소모를 줄일 수 있는 계층적 클러스터링을 활용한 질의 최적화 기법을 제안하였다.

이 기법은 빈번하게 변화하는 데이터에 따라 불필요한 조인연산의 실행을 회피시킬 수 있으며, 중간결과 공유를 통해 중복적인 조인연산 처리로 인한 CPU와 메모리의 낭비를 줄일 수 있다.

향후 연구로는 본 논문에서 제안하는 방식을 적용한 키워드검색시스템을 구현하고 그 성능을 기존방식과 비교 분석하고자 한다

참고 문헌

[1] 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.

[2] 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.

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

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

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

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

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