XML 데이타의 경로 유사성에 기반한 클러스터링 기법

(A Clustering Method Based on Path Similarities of XML Data)

최 일 환 † 문 봉 기 †† 김 형 주 †††
(Ilhwan Choi) (Bongki Moon) (Hyoung-Joo Kim)

요 약 최근의 XML 저장소에 관한 연구들은 기존의 데이타 저장을 위해 주로 사용해 왔던 관계형 데이타베이스에 효율적으로 XML 데이타를 매핑하는 기법이나 XML 데이타를 위한 새로운 전용 저장소에 대한 연구들이 주를 이룬다. XML 전용 저장소에서 많이 사용되는 방식으로 XML 문서를 파싱하여 각 노드들을 개별적인 객체로 생성한 후 이를 저장하는 방식이 있다. 이러한 저장 방식에서는 개별적인 객체들의 물리적 배치, 즉 클러스터링이 성능에 영향을 미칠 수 있다. 본 논문에서는 하나의 XML 문서를 보다 효율적으로 저장하는 클러스터링 기법을 제안한다. 제안하는 기법은 데이타 노드들의 경로 유사도를 기반으로 클러스터링을 수행하여 질의 요청에 대한 결과를 반환할 때 발생하는 페이지 I/O를 줄인다. 또한 경로 질의 처리시 필요한 클러스터만을 이용하여 질의 처리를 수행하는 방법을 제안한다. 이는 질의 처리 과정에서 불필요한 데이타를 제외함으로써 결과적으로 탐색 공간의 크기를 줄일 수 있어 보다 효율적인 경로 질의 처리를 가능하게 한다. 이밖에 본 논문에서는 기존의 다른 클러스터링 기법들과 제안한 기법들과의 성능 비교를 수행하고, 이를 통해 적절한 클러스터링 기법을 이용하면 XML 저장소의 성능을 향상시킬 수 있음을 보인다.

키워드 : XML 저장소, 클러스터링, 경로 질의

Abstract Current studies on storing XML data are focused on either mapping XML data to existing RDBMS efficiently or developing a native XML storage. Some native XML storages store each XML node with parsed object form. Clustering, the physical arrangement of each object, can be an important factor to increase the performance with this storing method. In this paper, we propose re-clustering techniques that can store an XML document efficiently. Proposed clustering technique uses path similarities among data nodes, which can reduce page I/Os when returning query results. And proposed technique can process a path query only using small number of clusters as possible instead of using all clusters. This enables efficient processing of path query because we can reduce search space by skipping unnecessary data. Finally, we apply existing clustering techniques to store XML data and compare the performance with proposed technique. Our results show that the performance of XML storage can be improved by using a proper clustering technique.

Key words : XML storage, clustering, path query

1. 서 론

XML의 사용이 늘어남에 따라 XML 데이타를 효율적으로 관리하기 위한 저장소 및 질의 처리 등에 대한 활발한 연구들이 진행중이다[1,2,4-9]. XML 저장소에 관한 연구들은 파일 시스템, 관계형 데이타베이스 시스템, 객체지향 데이타베이스 시스템, XML 전용 데이타베이스 시스템 등 다양한 저장 장치를 통해 XML 데이타를 관리하도록 하였다[2,4,6-8]. 최근의 연구들은 일반적으로 많이 사용되는 관계형 데이타베이스를 이용하여 XML 데이타를 저장하거나, XML 전용의 저장소를 구

· 본 연구는 BK-21 정보기술사업단과 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성지원사업(IITA-2005-C1090-0502-0016)의 연구결과로 수행되었음
† 학생회원 : 서울대학교 컴퓨터공학부
ihchoi@oopsla.snu.ac.kr
†† 비 회 원 : University of Arizona Department of Computer Science Associate Professor
bkmoon@cs.arizona.edu
††† 종신회원 : 서울대학교 컴퓨터공학부 교수
hjk@snu.ac.kr
논문접수 : 2005년 6월 18일
심사완료 : 2006년 1월 26일

축하는 방향으로 집중되고 있는 추세이다[1,5,9]. 이 중 XML 전용 저장소에서 많이 사용되는 방식으로는 XML 문서를 파싱하여 엘리먼트(element), 애트리뷰트(attribute) 등의 각 노드들을 개별적인 객체로 생성한 후 이를 저장하는 방식이 있다. 이러한 저장 방식에서는 개별적인 객체들의 물리적 저장 위치가 성능에 영향을 미칠 수 있다. 즉, 저장된 XML 문서를 읽어들일 때, 데이타의 물리적 배치에 따라 페이지 I/O 횟수에 차이가 생기게 된다. 이렇듯 데이타의 물리적인 배치를 고려하는 기법을 *클러스터링*이라 하며 과거 객체지향 데이타베이스 시스템 등에서는 이에 대한 활발한 연구가 이루어졌다[10]. 하지만 XML 분야에서는 클러스터링에 대한 연구는 거의 이루어지지 않았다. 현재 대부분의 XML 저장소에서는 클러스터링에 대한 별다른 고려 없이, 노드가 문서에 나타나는 순서(document order)대로 XML 데이타를 저장한다(그림 1 참조). 이에 본 논문에서는 XML 저장소에서 데이타를 보다 효율적으로 저장하는 클러스터링 기법에 대한 연구를 수행하였다.

본 논문의 기여도는 다음과 같다. 첫째, 본 논문에서는 하나의 XML 문서를 저장할 때 문서내의 각 노드들을 효율적으로 클러스터링하는 클러스터링 기법(PSim 클러스터링)을 제안한다. 제안하는 기법은 각 노드들의 경로 유사도를 계산하여 비슷한 경로를 가지는 노드들을 함께 저장한다. 이에 앞서 우선적으로 경로에 기반한 클러스터링 기법과 레이블에 기반한 클러스터링 기법을 소개한다. 두 기법은 간단한 직관적인 클러스터링 기법으로 각각 PSim 클러스터링의 최소와 최대 범위를 한정한다. 둘째, 본 논문에서는 제안하는 클러스터링 결과를 경로 질의의 처리에 효과적으로 적용하기 위해 서명(signature)을 이용한 질의 처리 방법을 제안한다. 이를 통해 경로 질의 처리에 필요한 데이타 탐색 공간을 줄여 보다 효과적인 질의 처리를 수행할 수 있다. 그밖에, 기존의 다른 연구에서 수행되었던 클러스터링 기법과의 비교 분석을 통해 제안하는 기법의 성능을 살펴보고, 이를 통해 적절한 클러스터링 기법이 XML 저장소의 성능을 크게 향상시킬 수 있음을 보인다.

본 논문의 2장에서는 관련 연구로 XML 분야에서 연구되었던 클러스터링 이슈에 관한 연구들과 기존의 객체지향 데이타베이스에서 연구되었던 클러스터링 기법들을 살펴본다. 3장에서는 데이타 모델 및 본문에서 사용하는 몇 가지 기본 개념을 정의하고, 간단한 직관적인 두 가지 클러스터링 기법들을 소개한다. 4장에서는 제안하는 PSim 클러스터링 기법에 관하여 설명하고, 5장에서는 PSim 클러스터링 결과를 질의 처리에 활용하는 기법을 설명한다. 6장에서는 기존의 클러스터링 기법들

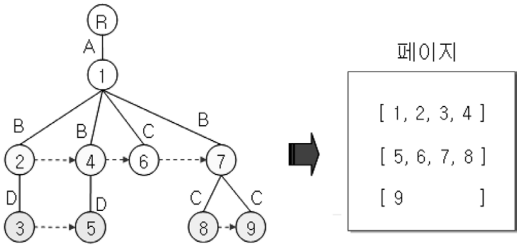


그림 1 일반적인 XML 문서의 저장

을 XML 문서를 저장하는데 적용하고 제안한 기법과의 성능 비교를 수행한다. 마지막으로 7장에서 결론 및 향후 연구에 대해서 언급하기로 한다.

2. 관련연구

클러스터링에 관한 연구들은 크게 문서의 구조적인 특성에 기반한 방식과 내용에 기반한 방식으로 구분할 수 있다. 문서의 구조에 기반한 연구들은 과거 객체지향 데이타베이스 분야에서 연구된 [11]과 같은 기법들이 대표적이다. [11]에서는 문서의 구조를 토대로 깊이우선(depth-first), 넓이우선(breadth-first) 방식을 제안하였다. 이들 중 깊이우선 방식은 XML 문서 원문에 나타나는 순서대로 엘리먼트들을 저장하는(document order) 방식과 유사하며, 클러스터링에 대한 특별한 고려를 하지 않는 많은 XML 저장소에서 이 방식을 사용하여 XML 문서를 저장한다. 하지만 정형화된 구조를 가지지 않는 XML 문서의 특성을 고려할 때, 일정한 구조에 기반하여 클러스터링을 수행하는 이러한 방식들은 한계를 갖는다.

XML의 경우 경로 질의를 보다 효율적으로 지원할 수 있도록 문서의 내용에 기반한 클러스터링 방식에 대한 연구들이 주로 수행되었다. XML의 경로는 태그(tag)라는 엘리먼트의 레이블들로 구성된다. 따라서 내용에 기반한 XML의 클러스터링 기법에서는 기본적으로 이러한 태그를 이용하여 클러스터링을 수행한다. [12]에서는 XML 문서를 저장하는 방식들을 클러스터링의 관점에서 비교, 분석하고, XML 문서를 저장할 때 적용할 수 있는 세 가지 클러스터링 방식에 대해 기술하였다. 첫 번째 방식은 DTD 분석 등을 토대로 실제로 연관성을 가지는 엘리먼트들을 클러스터링하는 방식으로 [8]에서 사용하는 방식이 여기에 해당한다. 두 번째 방식은 동일한 성격을 갖는 엘리먼트들을 함께 클러스터링하는 방식으로 [4,8]에서 사용하는 방식이다. 여기서 동일한 성격이란 동일한 태그명을 가지는 엘리먼트들 혹은, 형제 관계에 있는 엘리먼트들 등의 성격을 말한다. 이 경우 저장소로부터 XML 문서 원문을 다시 구성

하는 경우를 제외한 대부분의 상황에서 동일한 태그명에 따라 클러스터링 하는 경우가 보다 우수한 성능을 보여 주었다. 세 번째 방식은 XML 문서 원문에 나타나는 순서대로 엘리먼트들을 저장하는 방식으로 앞서 언급한 구조에 기반한 클러스터링 방식이다. [13]은 DTD에 기반한 두 가지 클러스터링 기법을 제안하였다. 첫 번째는 DTD에서 동일한 노드 타입을 가지는 엘리먼트들을 함께 저장하는 기법으로, 앞에서 언급한 [12]의 두 번째 방식과 유사하다. 두 번째는 스키마 그래프를 토대로 의미에 따라 서브 트리를 분류하고, 이에 기반하여 엘리먼트들을 분류, 저장하는 기법이다. 하지만 저장의 최소 단위인 의미 블록의 크기에 대한 고려가 없고, 이를 찾기 위한 휴리스틱이 매우 단순한 까닭에 이 기법을 사용한 클러스터링 기법이 상대적으로 큰 우위를 가지지는 못하는 단점을 지닌다.

기존의 내용에 기반한 클러스터링 기법들이 단순히 태그를 이용한 굵은 단위의 클러스터링이었다면, [14]에서는 경로 패턴에 기반한 보다 세밀한 단위의 클러스터링을 제공한다. 여기서는 서픽스(suffix) 트리를 사용하여 경로 패턴에 기반한 클러스터 색인(clustered index)인 PathGuide를 제안하였다. 제안한 방법은 동일한 경로 패턴을 가지는 엘리먼트들을 함께 저장하여 단순 질의뿐만 아니라 조상-자손 관계를 포함한 복잡한 형태의 경로 질의까지 효과적으로 처리할 수 있다. 하지만 PathGuide는 서픽스 단위로 클러스터링이 발생하기 때문에, 제한적인 경우에만 효과를 얻을 수 있다. 즉, 연속되는 부모-자식 관계의 긴 서픽스 형태를 가지는 질의를 처리하기에는 효과적이지만, 짧은 서픽스 형태를 가지는 보다 다양한 형태의 질의 처리에는 이득을 얻지 못한다.

이 밖에, 객체지향 데이타베이스에서는 사용자의 질의 패턴을 분석하여 이를 토대로 클러터링을 수행하는 연구들도 수행되었다[15,16]. 즉, 객체들의 연관성을 사용자의 질의 패턴으로부터 찾아내며, 이러한 기법들은 특히 자주 사용되는 질의에 대해 우수한 성능을 보인다[10]. 하지만 객체지향 데이타베이스에서는 객체의 멤버 함수 등을 통해서 빈번하게 사용될 노드들에 대한 정보를 유추할 수 있지만, XML의 경우 이러한 정보를 얻기가 어려운 문제를 갖는다.

3. 기본 개념

이 장에서는 본 논문에서 사용하는 XML의 데이타 모델과 몇 가지 기본 개념들에 설명한다.

3.1 데이타 모델과 개념 정의

정의 1. 하나의 XML 문서는 루트를 가지는 트리 Td로 다음과 같이 표현된다. T_d = (V, E, root, A), $V=V_i \cup V_t$. 여기서 Vi는 문서의 엘리먼트 혹은 애트리뷰트 노드들의 집합이고, V_t는 문서의 텍스트 노드들의 집합이다. 또한 $E \in V_i \times A \times V$로 노드들을 연결하는 간선을 의미하며, A는 모든 라벨의 집합, root $\in$ V는 T_d의 루트 노드를 의미한다. 또한 T_d의 각각의 노드들을 고유한 식별자를 갖는다.

정의 2. *클러스터*란 클러스터링을 통해 생성된 논리적 분할을 의미한다. 하나의 XML 문서 T_d에 대해 클러스터링을 수행하여 생성된 클러스터들을 각각 C_1, C_2, …, C_n이라고 하자. 임의의 클러스터 C_i에 포함된 모든 노드들을 $V(C_i)$라고 하면, T_d의 모든 노드 $V = \sum_{i=1}^{n} V(Ci)$ 가 된다.

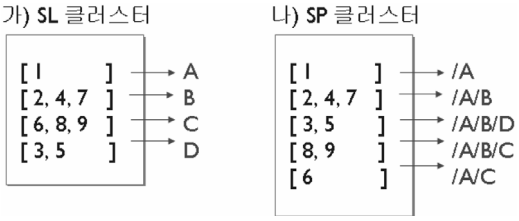


그림 2 SL 클러스터링과 SP 클러스터링

3.2 기본 클러스터링 기법들

본 절에서는 두 가지의 직관적인 클러스터링 기법들에 대해 소개한다. 이 기법들은 본 논문에서 제안하는 클러스터링 기법의 범위를 한정하는 역할을 한다.

첫 번째는 동일 레이블 클러스터링(SL 클러스터링) 기법으로, 동일한 태그명을 가지는 모든 노드들을 동일한 클러스터에 저장하는 기법이다. 하나의 클러스터에 저장되는 데이타 노드들은 문서 순서대로 저장되며, 각 클러스터는 저장된 데이타 노드의 레이블을 식별자로 가진다. 각 레이블 별로 하나씩 클러스터가 생성되므로, 클러스터링을 수행한 후 최종적으로 생성된 클러스터의 개수는 문서에 포함된 서로 다른 레이블의 개수와 같다. 그림 2의 가)에는 그림 1의 XML 문서에 대해 SL 클러스터링을 수행한 결과를 보여준다. 구현은 구조 조인 등에서 사용하는 레이블의 역색인 리스트를 이용하면 간단하게 구현이 가능하다.

두 번째는 동일 경로 클러스터링(SP 클러스터링) 기법으로, 루트로부터의 경로(절대 경로)가 동일한 모든 노드들을 동일한 클러스터에 저장하는 기법이다. 하나의 클러스터에 저장되는 데이타 노드들은 문서 순서대로 저장되며, 각 클러스터는 저장된 데이타 노드의 절대 경로를 식별자로 가진다. 이 기법은 절대 경로가 같은 노드들을 하나의 목표 집합(target set)으로 관리하는

DataGuide[17]와 같은 경로 색인을 이용하면 쉽게 구현이 가능하다. 각 절대 경로마다 하나씩 클러스터가 생성되므로, 클러스터링을 수행한 후 최종적으로 생성된 클러스터의 개수는 문서에 대한 DataGuide의 노드 수, 즉 목표 집합의 총 개수와 같다. 그림 2의 나)에서는 그림 1의 XML 문서에 대해 SP 클러스터링을 수행한 결과를 보여준다. 그림에서 보듯이 6번과 8번 노드는 동일한 레이블(C)을 가지므로 SL 클러스터링에서는 같은 클러스터에 저장되지만, 서로 다른 절대 경로(/A/C, /A/B/C)를 가지기 때문에 SP 클러스터링에서는 서로 다른 클러스터에 저장됨을 알 수 있다.

3.3 기본 클러스터링 기법들의 한계

SL 클러스터링과 SP 클러스터링의 차이는 클러스터링의 단위(granularity)에 있다. SP 클러스터링은 동일한 레이블을 가지는 노드들 중에서도 절대 경로에 따라 서로 다른 클러스터에 저장이 되기 때문에 SL 클러스터링에 비해 보다 미세 단위의 클러스터링을 수행한다. 이러한 단위의 차이는 질의 수행에도 영향을 미친다. 예를 들어, /A/D/X와 같은 절대 경로 질의를 수행할 때, SP 클러스터링 기법으로 저장된 경우 /A/D/X에 해당하는 클러스터에 저장된 노드를 반환하면 되므로 효율적이다. 하지만, SL 클러스터링의 경우 X라는 레이블을 가지는 엘리먼트들이 모두 하나의 클러스터에 저장된다. 즉 서로 다른 경로를 가지는 노드들이 섞여서 저장되므로 /A/D/X라는 특정 경로를 가지는 노드들이 인접하여 저장됨을 보장할 수 없다. 여기서 X에 해당하는 클러스터가 n개의 페이지에 저장된다면, 경로 /A/D/X에 해당하는 노드들이 최악의 경우 n개의 페이지에 걸쳐서 저장될 수 있다. 반면, //X와 같은 X를 레이블로 가지는 노드 전체를 반환하는 경로 질의가 들어오면 SL 클러스터링 기법으로 저장된 경우 레이블 X에 해당하는 클러스터에 저장된 노드를 반환하면 되므로 효율적이다. 하지만 SP 클러스터링의 경우 절대 경로 단위로 클러스터를 생성하여 저장하기 때문에, X로 끝나는 서로 다른 절대 경로를 가지는 데이타들이 여러 클러스터에 흩어져서 저장되므로 SL 클러스터링에 비해 비효율적이 된다.

이렇듯, SL, SP 클러스터링은 특정 형태의 질의에서만 효과를 보이는 한계를 가진다. 따라서 이를 해결하기 위해서는 질의 형태에 국한되지 않는 융통성 있는 클러스터링 기법이 필요하다. PathGuide[14]는 기본 클러스터링 기법의 한계를 벗어나기 위한 기법을 제안하는데, 여기서는 절대 경로가 같은 노드들을 기본 클러스터로 두고 이들의 서픽스를 기반으로 그룹을 이루도록 한다. 예를 들어, /A/X/Y, /B/X/Y/, /A/C/Y와 같은 클러스터 C1, C2, C3가 있으면, 우선 X/Y를 서픽스로 가지는 C1, C2 클러스터가 다시 새로운 클러스터 그룹 G1으로 통합된다. 이후 또다시 Y를 서픽스로 가지는 C3와 새로 생성된 클러스터 그룹인 G1이 새로운 클러스터 그룹 G2로 통합됨으로써 클러스터링을 완료한다. 생성된 클러스터 그룹 내에 통합된 기본 클러스터 및 클러스터 그룹들은 그 정보를 그대로 유지하여 해당 서픽스를 가지는 질의가 들어올 때, 효율적인 처리를 할 수 있도록 하였다. 이러한 방식은 SP를 기본으로 하고, 이들을 적절히 통합하여 SL을 생성하는 형태로 볼 수 있다. 따라서 SL과 SP 기법의 장점을 모두 가질 수 있다는 점에서 보다 융통성 있는 클러스터링 기법으로 볼 수 있다. 하지만 PathGuide에서 적용하는 서픽스 기반의 통합 방식은 연속되는 부모-자식 관계의 긴 서픽스 형태를 가지는 질의를 처리하기에는 효과적이지만, 짧은 서픽스 형태를 가지는 보다 다양한 형태의 질의 처리에는 한계를 가진다. 앞의 예에서 '//X/Y'와 같은 질의가 들어왔을 때, PathGuide는 G1을 살펴 손쉬운 처리가 가능하지만, '//A//Y'와 같은 질의가 들어올 경우 가장 큰 단위의 클러스터인 G2를 탐색하여 질의를 수행해야 한다. 만일 '/A/B/C/D/E//Y'와 같은 상당히 구체적인 질의가 들어오더라도 PathGuide에서는 기본적으로 Y를 서픽스로 가지는 클러스터 그룹 전체를 탐색해야 하는 한계를 가진다. 이에 4장에서는 기본 클러스터링 기법들인 SL, SP의 장점을 모두 포함하는 동시에, 다양한 형태의 질의에도 효과적으로 대처 가능한 클러스터링 기법인 PSim 클러스터링 기법을 제안한다. 제안하는 기법은 경로의 서픽스를 기반으로 통합을 수행하는 PathGuide와 달리 전체 경로의 유사도를 바탕으로 통합을 수행하기 때문에 보다 자유로운 형태의 질의에도 효율적으로 동작할 수 있다.

4. PSim 클러스터링

하나의 SL 클러스터에 저장된 데이타 노드들은 각각의 경로에 따라 다시 하나 이상의 SP 클러스터들로 분류될 수 있다. 따라서 SL과 SP 클러스터링 기법의 장점을 모두 유지할 수 있는 가장 간단한 방법은 SL 클러스터 내의 데이타 노드들을 다시 SP 클러스터링 기법에 따라 클러스터링을 수행하는 것이다. 다시 말하면, SP 클러스터링을 통해서 생성된 SP 클러스터들의 경로를 분석하여 동일한 레이블을 가지는 클러스터들을 그룹화하는 것이다. 이 경우, 클러스터링 효율성은 SL 클러스터를 구성하는 SP 클러스터들의 배치 순서에 따라 결정된다. 예를 들어, /A/B/C, /D/B/C, /X/Y/C의 경로를 가지는 세 개의 SP 클러스터 sp1, sp2, sp3가 있다고 하자. '//C'의 질의가 들어오면 sp1, sp2, sp3에 저장된 데이타 모두가 반환된다. 이 경우에는 SP 클러스터의 배치 순서는 질의의 결과를 반환하는데 아무런 영향

을 미치지 못한다. 반면 '//B/C'의 질의가 들어오면 sp1, sp2에 저장된 데이타만 반환된다. 따라서 가급적이면 유사한 경로 요소를 가지는 sp1과 sp2를 새로운 클러스터로 통합하여 가까이 배치하는 것이 유리하다. 이렇듯 다양한 형태의 질의에 융통성 있는 효과를 보이기 위해서는 SP 클러스터의 효율적인 배치가 중요하다. 이 장에서 설명하는 PSim 클러스터링 기법에서는 이러한 SP 클러스터들의 통합을 위해 SP 클러스터들의 절대 경로를 비교하여 유사한 절대 경로를 가지는 SP 클러스터들을 가까이 배치하도록 한다.

4.1 PSim 클러스터링의 수행 단위

PSim 클러스터링 기법은 클러스터링을 수행하는 최소 기본 단위로 하나의 데이타 노드가 아닌 SP 클러스터를 사용한다. 이는 기본적으로 동일한 경로를 가지는 노드들은 언제나 동일한 클러스터에 저장됨을 의미한다. 또한 클러스터링의 최대 단위는 데이타 노드의 레이블로 제한한다. 즉, SL 클러스터링과 마찬가지로 서로 다른 레이블을 가지는 데이타 노드들은 서로 다른 클러스터에 저장된다. 결과적으로 PSim 클러스터링은 SL 클러스터내의 데이타 노드들을 SP 클러스터로 분류하고, 분류된 SP 클러스터들을 적절히 통합하여 배치하는 작업이 된다. 이를 통해 절대 경로 질의를 효율적으로 수행할 뿐만 아니라, 다양한 형태의 상대 경로 질의도 효율적으로 수행할 수 있도록 한다.

4.2 경로 유사도

PSim 클러스터링에서 SP 클러스터들의 통합은 이들이 가지는 경로의 유사도에 기반하여 이루어진다. 실제로 모든 데이타 노드들간의 경로를 비교하여 유사도를 계산하는 작업은 상당히 부담스러운 작업이다. 하지만 PSim 클러스터링은 개별의 데이타 노드가 아닌 SP 클러스터를 클러스터링의 기본 단위로 사용함으로써 유사도를 계산하는 과정에서 부하를 상당히 줄일 수 있다.

서로 다른 SP 클러스터는 각각 서로 다른 절대 경로 정보를 식별자로 가진다. 따라서 이들 절대 경로를 비교하여 서로 다른 SP 클러스터 간의 경로 유사도를 구할 수 있다. 절대 경로의 비교는 서로 다른 두 문자열을 비교하는데 일반적으로 사용되는 편집 거리(edit distance)[18]을 응용하여 수행할 수 있다. 본 논문에서는 한 클러스터의 절대 경로가 하나의 문자열로 보고, 절대 경로에 포함된 각각의 레이블을 문자열을 이루는 하나의 문자로 가정하여 편집 거리를 구한다. 길이가 각각 n, m인 절대 경로 $/a_1/a_2/.../a_n$과 $/b_1/b_2/.../b_n$을 각각 A, B라고 하면, 이들 절대 경로간 편집 거리 $\delta(A,B)$는 [19]와 같이 동적 프로그램(dynamic programming) 기법을 사용하여 다음과 같이 구할 수 있다. A_i를 $/a_1/a_2/.../a_i$, B_j를 $/b_1/b_2/.../b_j$라고 하고, $\delta(A_i,B_j)$를 $\delta_{i,j}$라 하자. 이제 크기 $(n+1)\times(m+1)$을 갖는 편집 행렬 $D=(d_{i,j})$를 하나 생성하고, $d_{0,0}=0$, $d_{0,j}=\delta(\varepsilon,B_j)$, di,0 = δ(Ai, ε)로 초기값을 설정한 후, 편집 행렬을 다음의 공식을 이용하여 구성한다.

$$d_{i,j}=\min(d_{i-1,j-1}+\delta(a_i,b_j),\ d_{i-1,j}+\delta(a_i,\varepsilon),$$
$$d_{i,j-1}+\delta(\varepsilon,b_j),\quad (1\le i\le n, 1\le j\le m)$$

이 때, 편집 거리 $\delta(A,B)=d_{n,m}$이 된다. 이러한 절대 경로간 편집거리를 이용하여 식별자로 각각 id1, id2의 절대 경로를 가지는 두 개의 클러스터 sp1과 sp2의 경로 유사도는 다음과 같이 계산할 수 있다.

$$\text{경로유사도}(sp1,sp2)=1-\frac{\delta(id1,id2)}{\max(id1\text{의 경로길이},id2\text{의 경로길이})}$$

여기서 $0\le$경로유사도≤ 1이 되며, 서로 다른 두 경로가 유사할수록 1에 가까운 값을 가지게 된다. 만일 id1과 id2가 같다면, $\delta(id1,id2)$가 0이 되어 경로유사도(sp1, sp2)는 1이 된다. 반면, id1과 id2가 경로상에 일치하는 부분이 전혀 없다면, $\delta(id1,id2)$는 max(id1의 경로길이, id2의 경로길이)와 같게 되어 경로유사도(sp1, sp2)는 0이 된다.

4.3 PSim 클러스터링 알고리즘

하나의 SL 클러스터를 이루는, 즉 경로가 동일한 레이블로 끝나는 서로 다른 SP 클러스터들 간의 경로 유사도는 SP 클러스터를 정점(vertex)으로 가지고 정점들 간의 경로 유사도를 간선의 가중치(weight)로 가지는 가중 그래프(weighted graph) G = (V, E)로 표현할 수 있다. 그림 3은 *경로 유사도 그래프*의 예를 보여준다. 하나의 XML 문서에 대해 경로 유사도 그래프는 문서에 포함된 개별의 레이블마다 각각 하나씩 존재한다. 이 때 PSim 클러스터링은 이러한 경로 유사도 그래프를 토대로 SP 클러스터들을 분류하는 그래프 분할 작업(graph partitioning problem)이 되며, 모든 레이블의 경로 유사도 그래프에 대해 분할 작업을 수행함으로써 클러스터링 과정이 완료된다.

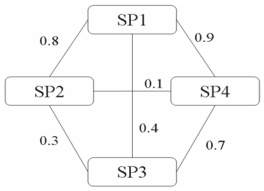


그림 3 경로 유사도 그래프

일반적으로 그래프 분할 알고리즘은 NP 완전 문제(NP complete problem)로 적절한 휴리스틱을 사용하여 해결한다. 본 논문에서는 경로 유사도 그래프의 분할을 위해서 [15]와 유사한 탐욕 알고리즘(greedy algorithm)을 수행한다. 그림 4에는 PSim에서 하나의 레이블에 대하여 경로 유사도 그래프 분할을 이용하여 클러스터링을 수행하는 알고리즘을 보인다. 알고리즘은 클러스터링의 분할을 고려하는데 있어서 경로 유사도가 높은 정점들을 우선적으로 고려하기 위해 정렬된 간선 리스트를 이용하여 가중치가 높은 간선부터 차례로 진행된다(3~14행). 이후 간선이 연결하는 두 정점이 저장된 클러스터를 찾아 이들을 통합하여 새로운 클러스터를 생성한다(4~11행). 물론 저장된 클러스터가 없는 경우에는 새로운 클러스터를 할당해준다(8~9행). 제안하는 알고리즘은 클러스터의 크기가 한 페이지(디스크 블록) 이상인 경우, 해당 클러스터에 더 이상 새로운 데이타를 저장하지 않도록 제한한다. 이는 하나의 클러스터의 크기가 페이지 단위 이상으로 커지는 경우에는 클러스터링으로 얻어지는 페이지 I/O의 이득이 의미를 가지지 못하기 때문이다. 최종적으로 상대적으로 경로 유사도가 떨어지는 나머지 정점들이 한 페이지에 저장될 수 있을 때, 이들을 하나의 클러스터에 저장한 후 클러스터링을 완료한다(12~14행).

그림 5에서는 그림 3의 경로 유사도 그래프에 대해 PSim 클러스터링이 수행되는 과정을 보여준다. 그림에서 2번과 5번 과정에서 이전 단계에서 발생한 클러스터가 통합되는 모습을 보여준다. 이 예제에서는 한 페이지에 저장 가능한 노드의 개수가 최대 10개이므로 3번 과정에서 클러스터 통합이 일어나지 않음을 알 수 있다.

5. PSim을 이용한 질의 처리

사용자로부터 질의가 들어왔을 때, 질의와 관련이 있

```
// The parameter 'edge_list' is the list of all edges in the
// similarity graph.
//
1   PSim_clustering(edge_list)
2     list sorted_elist = sort_by_weight(edge_list, desc);
3     for each edge e in sorted_elist do
4       <sp1, sp2> = vertexes which are connected by e;
6       cluster c1 = findCluster(sp1);
7       cluster c2 = findCluster(sp2);
8       if c1 == NULL then c1 = makeNewCluster(sp1);
9       if c2 == NULL then c2 = makeNewCluster(sp2);
10      if (sizeofCluster(c1)+sizeofCluster(c2))<PAGESIZE then
11        c1 = mergeClusters(c1, c2);
12      if (size of unclustered_vlist)<PAGESIZE then
13        makeNewCluster(unclustered_vlist);
14        break;
```

그림 4 PSim 클러스터링 알고리즘

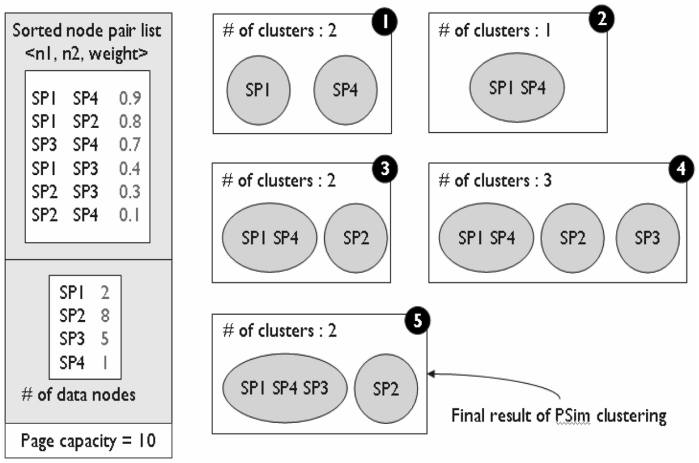


그림 5 PSim 클러스터링 예제

는 데이타 노드는 전체 문서의 일부분에 한정되어 있는 경우가 많다. 따라서 만일 필요한 일부분의 데이타만을 접근하여 질의 처리를 수행할 수 있다면, 질의 처리 시 데이타 탐색 공간의 크기를 줄일 수 있어 효과적인 질의 처리가 가능하다. 클러스터링을 통해 생성된 논리적 분할 단위인 클러스터는 데이타에 대한 부분 접근이 용이한 기반을 제공할 수 있다. 즉, 여러 개로 나누어진 클러스터들 중 질의 처리에 필요한 클러스터만 선택할 수 있다면, 선택된 클러스터에 저장되어 있는 데이타만을 탐색함으로써 질의 처리를 완료할 수 있다. 제안하는 PSim 클러스터링 기법은 데이타의 경로에 기반하여 문서를 분할한다. 이는 경로 질의를 질의문으로 사용하는 XML 질의에서 클러스터 선택을 용이하게 만든다. 즉, 분할된 각 클러스터에 포함된 데이타들의 경로 정보들을 미리 알 수 있다면, 경로 질의가 들어왔을 때 클러스터 선택을 쉽게 할 수 있다. 이 장에서는 PSim 클러스터링 기법으로 저장된 XML 문서에 대해 클러스터 단위로 접근할 수 있도록 하는 서명(signature)을 제안한다.

5.1 클러스터 서명

*클러스터 서명*은 클러스터에 저장된 데이타 노드들의 경로들을 대표하여 각각의 클러스터에 하나씩 할당된다. 이는 경로 질의를 수행할 때, 특정 레이블을 포함하는 경로를 가지는 데이타 노드가 클러스터에 존재하는지를 판단할 수 있도록 도와준다. 따라서 질의가 들어오면 질의에 부합하는 경로 정보를 가지는 클러스터를 선택할 수 있다.

클러스터 서명은 저장된 하나의 문서에 존재하는 서로 다른 레이블의 총 개수를 길이로 가지는 비트 문자열(bit string)로 생성되며, 문자열의 각각의 비트는 문서상의 각 레이블에 해당된다. 예를 들어 하나의 문서에 총 n개의 서로 다른 레이블 l_1, l_2, …, l_n이 존재한다고 하면 클러스터 서명은 길이가 n인 비트 문자열이 된다. 여기서 문자열의 첫 번째 비트는 l_1, 두 번째 비트는 l_2, …, n번째 비트는 l_n에 해당된다. 만일 클러스터에 저장된 데이타가 임의의 레이블 l_x를 경로에 포함하고 있다면 x번째 비트를 1로 설정한다. 이러한 과정은 데이타의 경로에 포함된 모든 레이블에 대해 이루어진다. 최종적으로 클러스터에 저장된 모든 데이타에 대해 작업이 수행되면 클러스터 서명이 구성이 완료된다. 따라서 클러스터 서명은 저장된 모든 데이타들의 경로에 포함된 레이블에 대한 정보를 표현할 수 있다.

클러스터 서명의 구성 과정은 클러스터에 저장된 실제 데이타 노드가 아닌 PSim 클러스터를 구성하는 SP 클러스터를 이용하여 보다 적은 부하를 가지고 구성할 수 있다. 우선 SP 클러스터에 대한 클러스터 서명을 생성한다. 하나의 SP 클러스터는 그에 해당하는 하나의 절대 경로를 가지므로 경로에 포함된 레이블들을 살펴 이에 해당하는 비트들을 설정함으로써 SP 클러스터 서명을 생성할 수 있다. 이후 PSim 클러스터 서명은 PSim 클러스터에 저장된 SP 클러스터들의 모든 서명들에 대해 비트 논리합 연산을 수행함으로써 얻어진다. 그림 6에서는 PSim 클러스터 서명을 생성하는 예제를 보여준다. 그림에서는 저장된 문서에 A, B, C, D, E, F의 총 6개의 서로 다른 레이블이 존재하여 여섯 자리의 길이를 가지는 비트 문자열이 서명이 된다. 그림 6의 가)에서는 SP 클러스터와 각각의 서명을 보여주며, 나)에서는 세 개의 PSim 클러스터에 저장된 SP 클러스터들과 이에 대한 논리합으로 클러스터 서명을 생성하는 과정을 보여준다.

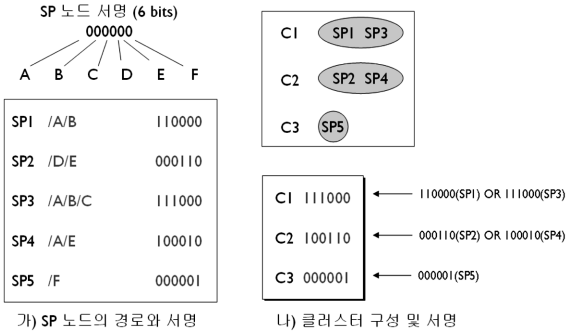


그림 6 클러스터 서명

5.2 클러스터 선택

각각의 클러스터에 할당된 클러스터의 서명은 별도의 인덱스로 따로 저장되어 관리되며, 이후 질의가 들어왔을 때 클러스터 선택을 위해 사용된다. 사용자로부터 질의가 들어오면 우선 질의에 대해 *질의 서명*을 생성한다. 질의 서명은 클러스터 서명과 마찬가지로 질의에 존재하는 레이블 항목에 대해 해당 비트를 설정함으로써 결정된다. 클러스터의 선택은 이렇게 생성된 질의 서명을 PSim 클러스터들의 각각의 클러스터 서명에 대해 비트 논리곱 연산을 수행함으로써 얻을 수 있다. 만일 논리곱 연산의 결과가 질의 서명과 일치할 경우, 해당 클러스터는 질의 처리 시 참조해야 할 클러스터로 선택된다. 왜냐하면 이 경우 질의 서명의 1로 설정된 모든 자리의 비트들에 대해 클러스터 서명 또한 1로 설정되어 있게 되기 때문이다. 이는 해당 클러스터가 질의문에 포함된 레이블 항목들을 모두 포함하는 클러스터임을 의미한다. 따라서 선택된 클러스터는 질의 결과에 해당하는 데이타를 포함할 가능성을 가진다. 예를 들어, 그림 6의 저장된 문서에 '//A//C'와 같은 질의가 들어온다고 하자. 이 질의에 대해 '101000'이라는 질의 서명이 생성되며,

이를 C1, C2, C3의 클러스터 서명인 '111000', '100110', '000001'에 대해 비트 논리곱을 수행하면 각각 '101000', '100000', '000000'의 결과를 얻을 수 있다. 최종적으로 질의 서명과 일치하는 결과를 보인 C1 클러스터가 질의 처리에 사용될 클러스터로 선정된다. 질의 처리는 선정된 C1에 대해서만 수행되며, C2와 C3 클러스터의 경우 질의 처리 시 고려 대상이 되지 않는다. 그림에서 보면 실제로 질의의 결과는 '/A/B/C'로 C1 클러스터에만 존재함을 확인할 수 있다.

이렇듯 클러스터 서명을 이용한 클러스터 선택은 질의 처리 시 데이타 탐색 공간을 효과적으로 줄일 수 있는 장점을 가진다. 하지만 경로상의 레이블의 순서를 고려하지 않고 단순히 존재 여부로 서명을 생성하기 때문에, 질의와 관계없는 클러스터가 선택될 가능성을 가진다는 단점이 있다. 예를 들어, '//A//C'의 질의에 '/C/B/A'라는 경로를 가지는 데이타를 포함한 클러스터가 선택될 수 있다. 6장의 성능 분석을 살펴보면 그럼에도 불구하고 대부분의 경우 데이타 탐색 공간을 크게 줄이는 효과를 가져옴을 알 수 있다.

6. 성능 분석

본 장에서는 제안하는 PSim 클러스터링 기법의 성능 평가를 위해, 본 논문에서 언급한 다른 클러스터링 기법들과 비교 분석을 수행한다. 비교 대상이 되는 기법들은 일반적으로 XML 저장소에 사용되는 문서의 순서대로 저장한 기법(RAW)와 경로를 기반으로 하는 기본 클러스터링 기법들인 SP, SL, 그리고 PathGuide(이하 PG)[14]이다.

6.1 실험 환경

실험에 사용한 XML 문서는 XMark[20]로 생성한 200,000개의 엘리먼트를 가지는 문서를 사용하였다. 테스트 질의로는 조상-자손 관계를 질의하는 '//'가 포함된 경로 질의로 총 1000개의 랜덤하게 생성된 질의문을 사용하였다. XML 문서의 저장은 객체 기반의 XML 저장소인 XDBox[21]를 사용하였다. XDBox는 XML의 노드를 객체 단위로 저장하며, XPath[22] 질의를 지원한다. XML 데이타의 저장에 사용되는 페이지 하나의 크기는 4KB이며, 버퍼 개수는 20개로 설정하였다. 동작 시스템은 펜티엄4 2.4GHz CPU와 512MB 메모리에서 Microsoft의 윈도우 XP 운영체제를 사용하였다.

6.2 다양한 클러스터링 기법들의 성능 비교

그림 7은 동일한 문서를 RAW, SP, SL, PG, PSim의 다섯 가지 클러스터링 기법으로 저장할 때 소요되는 시간 및 생성되는 클러스터의 개수를 보여준다.

그림에서 보듯이 RAW, SL 등의 굵은 단위의 클러스터링을 수행하는 기법들은 SP, PG, PSim 등의 세밀한 단위의 클러스터링을 수행하는 기법들에 비해 비교적 짧은 클러스터 생성 시간을 보여준다. 하지만 클러스터링은 동적으로 이루어지지 않고, 잦은 갱신이 일어나지 않는 문서의 경우 초기의 문서 저장 시에 한 번 이루어짐을 고려할 때, 클러스터 생성 시간은 큰 의미를 가지지 못한다. PG의 경우 동일한 SP 노드가 여러 그룹에 속하기 때문에, 클러스터의 크기가 최소 SP의 클러스터 하나의 크기에서 최대 SL 클러스터 하나의 크기까지 가변적이 된다. PSim의 경우, 경로가 유사한 SP 노드들을 하나의 클러스터로 통합하기 때문에 SP보다 적은 클러스터 개수를 보여준다.

이러한 클러스터 특성의 차이는 질의 수행 시 데이타 탐색을 위해 발생하는 페이지 I/O에 영향을 주게 된다. 즉, SL의 경우 비교적 단순한 태그 위주의 질의에서는 뛰어난 성능을 보이게 되지만, 질의가 복잡해 지면서 경로의 형태가 들어날수록 세밀한 단위의 클러스터링 기법들이 우위를 보이게 된다. 그림 8은 이들 기법 각각에 대해 동일한 랜덤 질의문을 수행한 결과를 보여준다. 그림에서 x축은 각각 조상-후손 관계를 묻는 //가 하나의 질의문에 포함된 개수를 나타낸다. 실험에서는 //의 개수를 1개에서 5개까지 변화시키며, //의 개수에 따라 각 200개씩 총 1000개의 랜덤 질의문을 사용하였다. y축은 질의 처리에 필요한 페이지 I/O의 총 합을 나타낸다.

이 실험에서 클러스터링 기법을 적용하지 않은 RAW의 경우 클러스터링을 적용한 다른 기법들에 비해 10배 이상의 페이지 I/O가 발생하였다. 따라서 XML 저장소에 문서를 저장할 때, 클러스터링 기법을 적용하는 것이 매우 효과적임을 알 수 있다. 기본 클러스터링 기법의 경우, //의 개수가 적은 경우에는 질의문이 여전히 긴 경로 형태를 유지하고 있기 때문에 3.3절에서 예상한 대로 SP가 SL에 비해 더 나은 성능(더 적은 페이지 I/O)을 보인다. 반면 //의 개수가 증가함에 따라 굵은 단위

	RAW	SP	SL	PG	PSim
생성 시간	118.391sec	344.875sec	246.594sec	346.015sec	352.734sec
총클러스터개수	1	536	77	가변	338
평균노드개수 (/1클러스터)	206130	385	2677	가변	610

그림 7 클러스터 생성 시간 및 특성

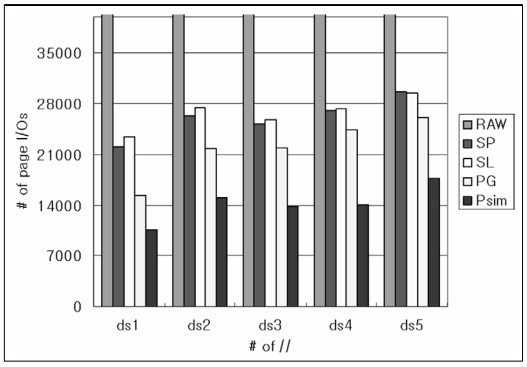


그림 8 다양한 클러스터링 기법들의 성능 비교

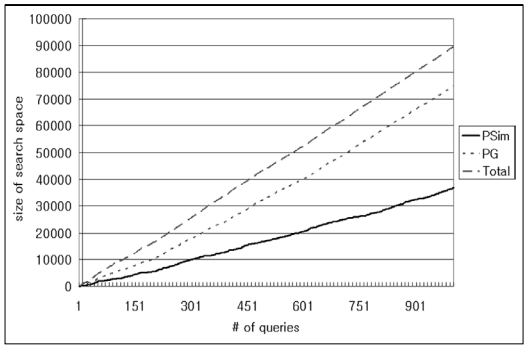


그림 9 질의 처리 시의 탐색 공간의 크기

의 클러스터링 기법인 SL이 더 나은 성능을 보이는 것을 알 수 있다. 여러 클러스터링 기법들 중에서는 기본 클러스터링 기법들(SL, SP)에 비해 이들을 확장한 PG, PSim이 가장 뛰어난 성능을 보였다. 이는 확장 클러스터링 기법들이 기본 클러스터링 기법들의 장점을 그대로 유지하면서 이들이 가지는 한계를 극복했기 때문이다.

특히, PSim의 경우 모든 경우에 있어서 PG보다 나은 성능을 보였다. 두 방식간의 성능 차이는 //의 개수가 증가함에 따라 더 차이가 남을 알 수 있다. 본 논문에서 예상했던 바와 같이, PG는 상대적으로 간단한 질의나 긴 서픽스 형태(예를 들어, /A//B/C/D와 같은)를 가지는 질의들에서만 좋은 성능을 보여준 반면, PSim은 //가 더 많이 포함된 복잡한 질의문이나 상대적으로 짧은 서픽스 형태(예를 들어, /A/B/C//D와 같은)를 가지는 질의문 등을 포함한 대부분의 질의에서 더 나은 성능을 보여주었다. 이러한 성능 차이의 주요한 요인 중의 하나로 탐색 공간의 크기를 들 수 있으며, 다음 절에서 이에 대해 살펴본다. 이처럼 그림 8의 결과는 제안하는 PSim 기법이 본 논문에서 언급한 다른 기법들 보다 다양한 형태의 질의를 보다 효율적으로 수행할 수 있는 유연한 기법임을 보여준다.

6.3 질의 처리 과정의 페이지 I/O 비교

그림 9에서는 PG와 PSim의 두 클러스터링 기법을 적용할 때, 질의 처리 시의 탐색공간의 크기를 보여준다. 본 논문에서 언급된 다른 기법에서는 탐색 공간을 줄이는 방법을 제공하지 않기 때문에, 여기서는 두 기법만을 비교하였다. 그림에서 'Total'은 탐색 공간을 줄이지 않는 경우의 전체 탐색 공간의 크기를 보여준다. 그림에서는 1000개의 질의문을 수행하고, 질의를 처리하기 위해 접근할 필요가 있는 데이타가 저장된 페이지의 수에 대한 누적값을 기록하였다. 그림에서 보여지듯이 PG와 PSim 두 기법 모두 'Total'과 비교하여 탐색 공간을 줄이는 효과가 있음을 알 수 있다. 특히, PSim이 PG에 비해 질의 처리에 훨씬 더 적은 탐색 공간을 필요로 함을 알 수 있다. 비록 PG가 긴 서픽스 형태를 가지는 질의의 경우에는 탐색 공간을 크게 줄일 수 있지만, 서픽스 기반의 클러스터링 방식으로 인한 한계로 많은 경우에 있어서 불필요한 데이타 탐색을 수행하였다. 이러한 결과는 5.2절에서 제안한 클러스터 선택 방법이 질의 처리에 필요한 탐색 공간을 효과적으로 줄일 수 있음을 보여준다.

7. 결론 및 향후 연구

최근의 XML 저장소에 관한 연구들은 기존의 데이타 저장을 위해 주로 사용해 왔던 관계형 데이타베이스에 효율적으로 XML 데이타를 매핑하는 기법이나 XML 데이타를 위한 새로운 전용 저장소에 대한 연구들이 주를 이룬다. 하지만 XML 문서를 저장 할 때, 저장되는 XML 데이타의 적절한 물리적 배치가 성능에 미치는 영향에 대한 연구는 별로 이루어지지 않았다. 이러한 클러스터링에 관한 연구는 이전 OODB 분야에서 많이 수행되었지만, 새로운 데이타 모델을 가지는 XML 분야에서 XML 데이타에 보다 적합한 효율적인 클러스터링 기법에 대한 연구는 충분히 의미를 가진다.

본 논문에서는 데이타 노드들의 경로 정보를 비교하여 유사한 경로를 가지는 노드들을 클러스터링하는 PSim 클러스터링 기법을 제안하였다. 제안하는 기법은 동일 경로를 가지는 노드들을 클러스터링을 위한 하나의 기본 단위로 설정하고, 서로 다른 경로들 간의 경로 유사도를 계산한다. 클러스터링은 이들 유사도에 대해 그래프 분할 기법을 적용하여 수행된다. 제안하는 기법은 특정한 형태의 질의에 한정되지 않고, 다양한 형태의 질의에서 효율적인 성능을 보임을 실험을 통해 살펴보았다. 또한 본 논문에서는 PSim 클러스터링 기법을 통해 생성된 클러스터들을 사용하여 질의 처리 시 필요한 페이지 I/O를 효과적으로 줄일 수 있는 방법을 제안하

였다. 각 클러스터들은 저장된 데이타들의 경로 정보를 대표하는 서명을 가지며, 이를 질의문과 비교하여 질의 처리 시 필요한 클러스터만을 선택할 수 있다. 따라서 질의와 관계없는 클러스터들은 질의 처리 시에 탐색할 필요가 없게 되며, 데이타 탐색 공간의 크기를 크게 줄일 수 있는 장점을 갖는다. PSim 클러스터링은 경로에 기반한 기본적인 클러스터링 기법들(SL, SP) 및 이들을 확장한 기법(PathGuide)과 비교하여 질의 처리 과정과 결과의 저장 공간에 있어서 보다 우수한 성능을 보인다.

제안하는 기법은 기본적으로 문서의 갱신이 빈번하지 않음을 가정한 기법이다. 따라서 향후 동적으로 갱신되는 문서에 대한 클러스터링 기법에 대한 연구가 필요하다. 그밖에 PSim 클러스터를 이용한 질의 처리 과정에서 선택된 클러스터에 저장된 데이타들에 대해 기존의 여러 질의 최적화 기법들의 적용에 대한 연구가 필요하다.

참 고 문 헌

[1] D. Barbosa, A. Barta, A. Mendelzon, G. Mihaila, F. Rizzolo, and P. Rodriquez-Gianolli, ToX-The Toronto XML Engine, in: Proc. WIIW '01, (Brazil, 2001) 66-73.

[2] Alin Deutsch, Mari Fernandez, and Dan Suciu, Storing semistructured data with STORED. In: Proc. SIGMOD 1999, (ACM Press, Philadelphia, 1999) 431-442.

[3] Excelon Corp, Excelon-the EBusiness Information Server, http://www.exln.com/

[4] D. Florescu and D. Kossman, Storing and Querying XML Data Using an RDBMS, IEEE Data Engineering Bulletin 22(3) (1999) 27-34.

[5] H. V. Jagadish, Shurug Al-Khalifa, Adriane Chapman, Laks V.S. Lakshmanan, Andrew Nierman, Stelios Paparizos, Jignesh M. Patel, Divesh Srivastava, Nuwee Wiwatwattana, Yuqing Wu and Cong Yu, TIMBER: A Native XML Database, VLDB 11(4) (2002) 274-291.

[6] C. C. Kanne and G. Moerkotte, Efficient storage of XML data, in: Proc. ICDE '00 (IEEE Computer Society, San Diego, 2000) 198.

[7] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom, Lore: A Database management system for semistructured data, SIGMOD Record 26(3) (1997) 54-66.

[8] Jayavel Shanmugasundaram, Kristin Tufte, Chun Zhang, Gang He, David J. DeWitt, Jeffrey F. Naughton, Relational Databases for Querying XML Documents: Limitations and Opportunities, in Proc: VLDB '99 (Morgan Kaufmann, Edinburgh, 1999) 302-314.

[9] Software AG. Tamino Information Server for Electronic Business, Technical Whitepaper, http://www.tamiono.com/tamino/Download/tamino.pdf

[10] Elisa Bertino , Amani A. Saad , M. A. Ismail, Clustering techniques in object bases: a survey, Data & Knowledge Engineering 12(3) (1994) 255-275.

[11] J. Banerjee, W. Kim, S-J. Kim, and J. F. Garaza, Clustering a DAG for CAD databases, IEEE Transactions on Software Engineering 14(11) (1988) 1684-1699.

[12] Feng Tian, David J. DeWitt ,Jianjun Chen, Chun Zhang, The Design and Performance Evaluation of Alternative XML Storage Strategies, SIGMOD Record 31(1) (2002) 5-10.

[13] Xiaofeng Meng, Daofeng Luo, Mong Li Lee, Jing An, OrientStore: A Schema Based Native XML Storage System, in: Proc: VLDB '03, (Morgan Kaufmann, Berlin, 2003) 1057-1060.

[14] Jiefeng Cheng, Ge Yu, Guoren Wang, Jeffrey Xu Yu, PathGuide: An Efficient Clustering Based Indexing Method for XML Path Expressions, in: Proc. DASFAA '03, (IEEE Computer Society, Kyoto, 2003) 257-.

[15] Carsten Gerlhof, Alfons Kemper, Christoph Kilger, Guido Merkotte, Partition-Based Clustering in Object Bases, in: Proc. FODO'93, Lecture Notes in Computer Science, vol. 730 (Springer, Chicago, 1993) 301-316.

[16] Manolis M. Tsangaris and Jeffrey F. Naughton, A Stochastic Approach for Clustering in Object Bases, in: Proc: SIGMOD 1991, (ACM Press, Colorado, 1991) 12-21.

[17] Roy Goldman and Jennifer Widom, DataGuides: enabling query formulation and optimization in semistructured databases, in: Proc. VLDB '97, (Morgan Kaufmann, Athens, 1997) 436-445.

[18] V. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals, Soviet Physics Dokl. 10(8) (1966) 707-710.

[19] R. A. Wagner, M. J. Fischer, The string-to-string correction problem, ACM 21(1) (1974) 168-173.

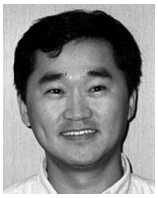
[20] Albrecht Schmidt, Florian Waas, Martin L. Kersten, Michael J. Carey, Ioana Manolescu, Ralph Busse, XMark: A Benchmark for XML Data Management, in Proc. VLDB '02, (2002) 974-985.

[21] Jongik Kim, Ilhwan Choi, Hyun-Sook Lee and Hyoung-Joo Kim, XDBox: Impelementation of XML object repository, in Proc: KISS Spring (Jeju, 2003).

[22] A. Berglund, S. Baog, D. Chamberlin, et al, XML Path Language(XPath), ver. 2.0, W3C Working Draft, Tech. Report, http://www.w3.org/TR/4, 2001.

최 일 환

1996년 서울대학교 컴퓨터공학과 학사
1998년 서울대학교 컴퓨터공학과 석사
1998년~현재 컴퓨터공학과 박사과정. 관심분야는 데이타베이스, XML, 질의처리



문 봉 기

1996년 미국 메릴랜드 대학 졸업(박사) 현 University of Arizona, Computer Science 부교수. ACM SIGMOD (2003), EDBT (2002), ISDB (2002), WWW (2002), VLDB (2001), ACM SIGMOD (2001) 위원회 위원 역임, 2002년 ACM SIGMOD Proceedings Chair 역임. 관심분야는 XML indexing, data mining, data warehousing과 parallel & distributed processing

김 형 주

정보과학회논문지 : 데이타베이스
제 33 권 제 1 호 참조