(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl. [7]
G06F 17/30

(45) 공고일자 2002년10월25일
(11) 등록번호 10-0358348
(24) 등록일자 2002년10월11일

(21) 출원번호 10-2000-0029993
(22) 출원일자 2000년06월01일

(65) 공개번호 특2001-0109665
(43) 공개일자 2001년12월12일

(73) 특허권자 (주) 아이티캠프
서울특별시 관악구 신림동 산56-1 138-416
송하주
서울 송파구 잠실1동 주공아파트 118-406
김형주
서울 관악구 신림동 산56-1

(72) 발명자 송하주
서울 송파구 잠실1동 주공아파트 118-406
김형주
서울 관악구 신림동 산56-1

(74) 대리인 김은구
김민철
전광출

심사관 : 류동현

(54) 엑스엠엘 데이터의 효과적인 검색을 위한 다중 경로인덱스 방법

요약

본 발명은 XML 데이터의 효과적인 검색을 위한 다중 경로 인덱스 방법으로서, 확장된 경로식에 포함된 각각의 단일 경로에 대해 고유한 경로 식별자를 부여하며, 인덱스 키값과 경로 식별자를 조합하여 저장함으로써 하나의 인덱스만을 이용하여 다수의 경로에 대한 검색과 단일한 경로에 대한 인덱스 기능을 동시에 지원하는 다중 인덱스 방법의 제공을 목적으로 한다.

본 발명은 다중 경로 인덱스는 확장 경로식으로 표현되는 질의에 대한 검색 성능이 향상되며, 다수의 경로에 대한 인덱스를 통합적으로 제공함으로써 인덱스 관리가 용이 하다. 또한 확장 경로뿐만 아니라 개개의 경로에 대한 검색 기능을 제공할 수 있으며, 경로 식별자를 이용하는 방식에서 기존의 단일 경로 인덱스 기법에 쉽게 응용해서 사용할 수 있는 효과가 있으며, B+-트리를 크게 수정하지 않고 사용할 수 있어 실용성이 우수하다.

대표도

도 4

색인어

XML, 인덱스, 다중경로, 컴퓨터, 데이터베이스, 스키마, 경로식별자

명세서

도면의 간단한 설명

도1은 종래의 XML DTD(document type definition)프로그램.

도2는 XML DTD의 데이터를 객체 데이터베이스에 저장하기 위한 객체 데이터베이스의 스키마.

도3은 본 발명에 따른 다중 경로 인덱스 방법의 흐름도.

도4는 본 발명에 따른 다중 경로 인덱스 방법에서 경로식별자를 할당방법의 흐름도.

도5는 본 발명에 따른 경로식별자를 할당하는 알고리즘.

도6은 본 발명에 따른 다중경로 인덱스에서 단말 노드의 레코드 구조도.

도7은 본 발명에 따른 XML문서를 객체 데이타베이스에 저장한 예시도.

도8은 본 발명의 실험실시에서 사용된 경로의 구조도.

도9는 본 발명의 실험실시에서 경로의 개수에 대한 포인트 질의의 성능비를 도시한 그래프.

도10은 본 발명의 실험실시에서 경로의 개수에 대한 영역 질의의 성능비를 도시한 그래프.

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

본 발명은 XML 데이터의 효과적인 검색을 위한 다중 경로 인덱스 방법에 관한 것으로, 보다 상세하게는 데이타베이스에서 여러 개의 경로, 즉 다중 경로를 간략히 나타내어 확장 경로 질의를 효율적으로 처리하기 위한 방법에 관한 것이다.

인터넷 상의 데이터 교환을 XML 데이터를 효율적으로 저장하고 검색하기 위한 많은 연구들이 최근 진행되고 있다. 특히 관계형 데이터베이스(relational database management system : RDBMS)나 객체지향 데이터베이스(object-oriented database management system : OODBMS)를 이용하여 XML 데이터를 보다 효과적으로 저장하고 검색하기 위한 연구가 활발하다. 객체지향 데이터베이스는 데이터 모델이 XML의 특성을 자연스럽게 반영할 수 있고, XML에 대한 질의어를 기존 객체 질의어를 확장하여 지원할 수 있기 때문에 XML의 저장장치로서 각광을 받고 있다.

종래에는 확장된 경로에 해당하는 개개의 경로의 개수가 많아지면 전체 인덱스의 개수가 많아지게 된다. 따라서 인덱스 저장 공간이 늘어나고 검색 성능도 떨어지게 된다. 특히 포인트 질의(point query)의 경우에는 더욱 성능이 나빠지는 문제점이 있었다.

또한 확장 경로에 해당하는 개개의 단일 경로를 데이터베이스 관리자가 알아내어 각각에 대해 인덱스를 설정해야 하며, 인덱스를 삭제할 때에도 관련된 확장 경로에 대해 설정된 인덱스가 어떤 것들이 있는지를 관리자가 파악해야 한다. 이 과정에서 확장된 경로에서 일부 단일 경로에 대해서는 인덱스가 존재하지 않을 수도 있어서, 일관성 있는 인덱스 관리가 어렵고 인덱스를 이용하는 이득이 감소하게 되는 문제점이 있었다.

발명이 이루고자 하는 기술적 과제

본 발명은 상기와 같은 종래기술의 문제점을 해결하기 위한 것으로, 확장된 경로식에 포함되는 각각의 단일 경로에 대해 고유한 경로 식별자를 부여하며, 인덱스 키값과 경로 식별자를 조합하여 저장함으로써 하나의 인덱스만을 이용하여 다수의 경로에 대한 검색과 단일한 경로에 대한 인덱스 기능을 동시에 지원하는 다중 인덱스 방법의 제공을 목적으로 한다.

발명의 구성 및 작용

본 발명에 따른 XML 데이터의 검색을 위한 인덱스 방법은,

주어진 확장 경로식에 해당하는 개개의 단일 경로를 찾아내고 각각에 대해 경로식별자를 할당하는 단계; 상기 단일 경로별로 데이터베이스 내의 실제 데이터를 검색하여 인덱스 정보수집를 수집하는 단계; 상기 경로 식별자 및 수집된 상기 인덱스 정보를 다중 경로 인덱스에 저장하는 단계;을 포함한다.

상기 경로식별자를 할당하는 방법은, 경로 식별자를 할당하는 알고리즘에 완화경로식을 입력을 하는 제1단계; 상기 경로 식별자를 할당하는 알고리즘에서 경로검색(search-path)함수를 호출하여 상기 완화경로식에서 전경로의 가장 뒤쪽 엘리먼트와 후경로의 가장 앞쪽 엘리먼트 사이의 모든 경로를 탐색하는 제2단계; 상기 제2단계에서 탐색된 모든 경로와 해당 하부엘리먼트를 합쳐서 경로 익스텐트 집합에 삽입하는 제3단계; 상기 익스텐트가 구해지고 나면 찾아진 각각의 단일 경로에 대해 경로 식별자를 할당하는 제4단계;를 포함하는 것을 특징으로 한다.

상기 경로 식별자를 할당하는 알고리즘은, 완화 경로식을 입력받으며, 완화 경로식의 전경로의 가장 뒤쪽 엘리먼트와 후경로의 가장 앞쪽 엘리먼트 사이의 모든 경로를 상기 데이타베이스에서 검색하며, 검색된 각각의 단일 경로에 대해 경로 식별자를 할당하는 것을 특징으로 한다.

상기 다중 경로 인덱스의 레코드 구조는 헤더정보, 인덱스 키값, 서로 다른 경로의 개수, 해당 경로의 경로 식별자, 경로에 해당하는 객체 식별자의 개수 및 경로에 해당하는 객체의 객체 식별자들의 배열을 포함하는 것을 특징으로 한다.

다중 경로 인덱스에 대한 검색 및 갱신 방법은, 검색 경로에 대한 경로 익스텐트를 모두 구하고 경로 익스텐트내의 각각의 단일 경로에 대한 경로 식별자를 찾는 단계; 키값으로 상기 데이타베이스의 중간 노드와 최종 단말 노드를 찾는 단계; 상기 키값과 상기 경로 식별자의 값이 일치하는 객체 식별자만을 검색하는 단계;를 추가로 포함한다.

본 발명에 따른 다중 경로 인덱스를 사용하기 위한 조건은,

XML DTD 내의 엘리먼트(element)와 속성(attribute)의 이름은 해당 DTD 내에서는 유일해야 하며, 다른 DTD에 동일한 이름을 가진 것이 있더라도 서로 다른 것으로 한다. 또한 서로 다른 XML DTD에 존재하는 동일한 이름의 엘리먼트인 경우에는, 엘리먼트에 대응하여 생성되는 객체 데이터베이스 시스템에서의 클래스도 서로 다른 것으로 가정한다. 그리고 제안하는 다중 경로 인덱스 방법은 DTD 수준에서 순환(cycle)이 존재하는 경로에 대해서는 인덱스를 설정할 수 없다. 또한 동일한 이름의 속성이 다수의 엘리먼트에 대해 사용될 경우에는 그것들의 데이터 타입이 모두 동일한 것으로 가정한다.

다중 경로인덱스 방법은 XML에서 ID 속성값을 나타내는 IDREF를 통해 다른 XML 객체를 가리키는 경우나 동일한 XML 객체 내의 다른 객체를 가리키는 경우들에 대해서는 질의 경로를 사용할 수 없는 것으로 한다. IDREF의 경우 실제 그것이 참조할 대상이 어떤 타입의 객체가 될 지 알 수 없으므로 객체 데이타베이스 시스템의 스키마 상에서 참조관계를 그래프로 표현할 수 없게 되어 경로 인덱스를 사용할 수가 없기 때문이다.

모든 객체에 대해 상부객체는 하나만 존재한다. 즉 동일한 객체가 여러개의 객체에 의해 공유되는 경우는 없다. 그러나 하나의 상부객체에 대해 다수의 하부객체가 존재할 수 있다. 또한 하부객체는 상부객체에 대한 객체식별자(OID)를 유지한다. 그러나 상부객체에 대한 참조는 시스템 내부에만 유지되며 응용 프로그램 수준에서는 드러나지 않는다. 앞서 가정한 바와 같이 IDREF에 의한 참조관계를 객체 데이타베이스 시스템 수준에서의 참조관계로 지원하지 않는 점을 고려할 때, XML DTD에 해당하는 객체 데이타베이스의 스키마는 DAG형태가 될 것이므로 하나의 XML 객체를 나타내는 객체 데이타베이스의 객체들은 트리 형태로 나타내어 질 수 있다. 이 때 XML 객체를 나타내는 객체 데이타베이스에서 객체들의 최상위 객체를 '루트 객체'로 정의한다. 그리고 해당 트리의 최상위 객체는 DTD 타입의 이름을 가지는 객체 데이타베시스 시스템 클래스의 객체가 된다. 객체 데이타베시스 시스템으로 변환된 모든 XML의 엘리먼트에 대해서는 각각의 엘리먼트(혹은 클래스)에 대한 익스텐트 객체가 생성되고 이 객체는 해당 클래스의 모든 인스턴스들에 대한 포인터를 유지하고 있어 질의를 처리할 수 있게 한다. XML 데이터를 저장하는데 있어 일반 데이터와의 차이점은 루트 객체로부터 출발하는 부분 참조 경로(partial path instanciation)는 존재할 수 있으나, 루트가 아닌 객체로부터 경로가 생성되는 경우는 없는 것으로 한다.

예를 들어 '1'이 루트 객체를 나타낼 때, 1-2-3-4-5 또는 1-2-3 형태의 참조 경로는 존재할 수 있으나 1-2-3 없이 3-4-5의 형태는 없다는 점이다. 왜냐하면 루트를 제외한 모든 객체는 상부객체 - 하부객체의 참조 관계를 가져야 하기 때문이다.

용어 정의

객체 질의어에서 사용되는 기존의 경로식은 하나의 경로만을 나타내므로 단일 경로식이라 할 때, 기존 경로식 표현에 '*'문자를 결합하여 다음과 같은 형태로 나타낸 경로식을 완화 경로식(relaxed path expression)이라 한다.

A.B.C.D 와 A.B.C.D.D1과 같은 기존의 경로식은 단순 경로식이고, A.*.D 또는 A.B.*.D.D1과 같은 경로식은 완화 경로식이며, 네가지 모두 확장 경로식 이다. A.*.D와 A.B.*.D.D1의 전경로는 각각 A와 A.B이고, 후경로는 각각 D와 D.D1이 된다.

XML의 DTD 엘리먼트 또는 객체 데이타베시스 시스템의 스키마에서 나타난 클래스의 이름에 의한 경로식을 '스키마 경로'라 하고, 그것이 질의에 사용되었을 때는 '질의 경로'라 정의한다. 스키마 경로에 따라 데이터베이스에 저장된 일련의 객체들은 '경로 인스턴스(path instance)' 또는 '객체 경로(object path)'로 정의한다.

경로 식별자를 할당하는 알고리즘에 있어서

Pi - 인덱스 경로.

Di - 인덱스의 방향, 즉 순방향(forward, 경로의 방향과 일치) 또는 역방향(backward, 경로의 역방향).

KAi - 키가 되는 속성의 이름.

Hi - 경로의 시작 클래스로서 순방향 인덱스의 경우에는 후경로의 가장 앞쪽 엘리먼트(head(post_path(P))), 역방향의 경우에는 전경로의 가장 뒤쪽 엘리먼트(tail(post_path(P)))에 해당한다.

Ti - 경로의 끝 크랠스로서 순방향 인덱스의 경우에는 전경로의 가장 뒤쪽 엘리먼트(tail(post_path(P))), 역방향의 경우에는 후경로의 가장 앞쪽 엘리먼트(head(post_path(P)))에 해당한다.

Ei - 인덱스 i를 통해 검색 가능한 모든 경로 = $\varepsilon(P)$.

BTi - 인덱스 데이터를 저장하고 있는 B+-트리 객체의 객체 식별자.

각 단일 경로별 정보 : $\phi \in \varepsilon(P)$ 에서,

sp - 경로식,

$\eta(sp)$ - 경로에 해당하는 경로 인스턴스의 개수,

PID - 경로 식별자 이다.

이하 도면을 참조하여 상세히 설명한다. 그러나, 이들 도면은 예시적인 목적일 뿐 본 발명이 이에 한정되는 것은 아니다.

도1은 종래 XML DTD(document type definition)의 일 예를 보인 것이다. 도2는 도1의 XML DTD의 데이터를 객체 지향 데이타베이스에 저장하기 위한 객체지향 데이타베이스의 스키마의 일 예를 보인 것이며, 도2에서 네모칸과 이탤릭체 문자열은 각각 클래스와 클래스의 속성의 일예를 보인 것이다.

도3은 본 발명에 따른 다중 경로 인덱스를 구성하는 방법의 흐름도 이다.

다중 경로 인덱스를 구성하는 방법, 주어진 확장 경로식에 해당하는 개개의 단일 경로를 찾아내고, 각각에 대해 경로식별자를 할당하는 단계(S100), 상기 단일 경로별로 데이터베이스 내의 실제 데이터를 검색하여 인덱스 정보수집를 수집하는 단계(S200), 상기 경로 식별자 및 수집된 상기 인덱스 정보를 다중 경로 인덱스에 저장하는 단계(S300);로 되어 있다.

도4는 본 발명에 따른 다중 경로 인덱스 방법에서 경로식별자를 할당하는 흐름도 이다.

경로식별자를 할당하는 방법은, 경로 식별자를 할당하는 알고리즘에 완화경로식을 입력을 하는 제1단계(S110), 상기 경로 식별자를 할당하는 알고리즘에서 경로검색(search-path)함수를 호출하여 상기 완화경로식에서 전경로의 가장 뒤쪽 엘리먼트와 후경로의 가장 앞쪽 엘리먼트 사이의 모든 경로를 탐색하는 제2단계(S120), 상기 제2단계(S120)에서 탐색된 모든 경로와 해당 하부 엘리먼트를 합쳐서 경로 익스텐트 집합에 삽입하는 제3단계(S130), 상기 익스텐트가 구해지고 나면 찾아진 각각의 단일 경로에 대해 경로 식별자를 할당하는 제4단계(S140)로 되어 있다.

도5는 본 발명에 따른 경로식별자를 할당하는 알고리즘이다.

XML DTD와 인덱스를 설정할 확장 경로식 P가 주어졌을 때, 주어진 XML DTD를 만족하면서 P에 속하는 모든 단일 경로식을 해당 경로식에 대한 `경로 익스텐트'라 정의하고 $\varepsilon(P)$ 로 나타낸다. 도5의 알고리즘은 주어진 확장 경로식에 대한 경로 익스텐트를 구하고, 경로 식별자를 할당하는 알고리즘을 나타낸 것이다. XML DTD를 객체 데이터베이스 스키마로 변환하였을 경우 질의문에서 경로식으로 표현될 수 있는 곳은 원래의 XML DTD에서 상부 엘리먼트와 하부 엘리먼트 또는, 엘리먼트와 속성 관계를 이용하여 경로 익스텐트를 찾아낼 수가 있다.

도5의 알고리즘에서 사용되는 head(P)는 경로 P의 가장 앞쪽 엘리먼트를 나타내며, tail(P)는 경로 P의 가장 뒤쪽 엘리먼트 혹은 속성을 나타낸다. 그리고 엘리먼트 E에 대한 child(E)는 E를 구성하는 하부엘리먼트 또는 E의 속성들의 집합을 나타내는 것으로 한다.

도5의 알고리즘은 주어진 경로식이 '*'를 포함하지 않는 단일 경로식인 경우에는 주어진 경로식만으로 경로 익스텐트를 삼는다. 만일 완화 경로식이 주어진 경우에는 '전경로의 가장 뒤쪽 엘리먼트(tail(pre_path(P)))'부터 '후경로의 가장 앞쪽 엘리먼트 혹은 속성(head(post_path(P)))' 사이의 모든 경로를 찾기 위해 '경로검색(search_path)' 함수를 호출한다.

경로검색(search_path) 함수는, 인자로 넘겨 받은 엘리먼트(e)의 하부 엘리먼트(c) 중에서 후경로의 가장 앞쪽 엘리먼트(head(post_path(P)))와 동일한 것이 있는지를 검색한다. 동일한 엘리먼트가 있으면, 검색과정에서 스택에 저장된 탐색 경로 정보와 해당 하부 엘리먼트(혹은 속성)를 합쳐서 경로 익스텐트 집합에 삽입한다. 만일 하부 엘리번트(c)에 하부 엘리먼트가 존재하면, 경로검색(search_path) 함수를 재귀 호출(recursive call)하여 탐색을 계속한다.

이러한 탐색 과정은 하부 엘리먼트가 없는 단말 엘리먼트(leaf element)나 속성을 방문할 때까지 계속된다. 익스텐트가 구해지고 나면 찾아진 각각의 단일 경로에 대해 경로 식별자를 할당한다.

예를 들어 설명하면 도1의 XML DTD에 대해 도5의 알고리즘을 이용하여 확장 경로식 book.chapter.*.title의 경로익스텐트에 속하는 단일 경로에 경로 식별자를 할당하면 다음과 같다.

book.chapter.title - 경로식별자(PID) : 1

book.chapter.section.title - 경로식별자(PID) : 2

book.chapter.section.subsection.title - 경로식별자(PID) : 3

도6는 본 발명에 따른 다중경로 인덱스에서 단말 노드의 레코드 구조도 이다.

다중 경로 인덱스의 인덱스 정보는 단말 레코드의 구조를 변경시킨 B+-트리를 이용하여 저장한다. 다중 경로 인덱스에서 단말 노드의 레코드 구조의 각 부분이 설명하면,

record header --- 해당 레코드의 크기 등을 기록한 헤더 정보,

key value --- 인덱스 키값,

paths --- 서로 다른 경로의 개수,

PID --- 해당 경로의 경로 식별자,

OIDs --- 경로에 해당하는 객체 식별자의 개수,

OIDs --- 경로에 해당하는 객체의 객체 식별자들의 배열, 이다.

여기서 경로 식별자는 상기 경로 식별자 할당 알고리즘에 의해 주어진다. 다중 경로 인덱스의 중간 노드 구조는 일반적인 B+-트리의 중간 노드 구조와 동일하다.

도7은 본 발명에 따른 XML문서를 객체 데이타베이스에 저장한 예시도 이다.

도7은 도1에서 제시된 XML DTD 스키마에 따르는 XML 문서들을 객체 데이타베이스에 저장한 것으로 데이터베이스의 일부만을 나타낸 것이다. 도7에서 원은 객체(object)를 의미하며 원 내의 숫자는 해당 객체의 객체 식별자를 의미한다. 원 옆의 숫자는 mdate의 값이다. 이 데이터베이스에 대해 경로 book.chapter.*.mdate에 대한 다중 경로 인덱스를 i라 할 때, i는 다음과 같이 구현된다.

Pi - book.chapter.*.mdate

Di - backward

KAi - mdate

Hi - book

Ti - mdate

Ei - book.chapter.body.mdate

book.chapter.subsection.body.mdate

book.chapter.section.subsection.body.mdate

경로별 정보

book.chapter.body.mdate - PID : 1, ㄱ:2

book.chapter.subsection.body.mdate - PID : 2, ㄱ:2

book.chapter.section.subsection.body.mdate - PID : 3, ㄱ:7

도7을 참고로 인덱스 내의 단말레코드는 나타내면 다음과 같다.(record header 정보는 생략)

"1999:11:03",3,(1,1,o2),(2,1,o1),(3,1,o1))

"2000:01:01",3,(1,1,o1),(2,1,o2),(3,4,o1,o2,o2))

"2000:02:01",1,(3,2,o1,o2))

다중 경로인덱스의 검색 및 갱신 방법,

다중 경로 인덱스에 대한 검색 과정은 일반 B+-트리와 동일하되 단말 레코드의 검색만을 달리한다. 즉 B+-트리와 동일한 방식으로 인덱스 트리를 탐색하되 최종적으로 단말 노드에서만 주어진 질의 경로를 만족하는 객체 식별자만을 선택하면 된다. 따라서 원하는 검색 경로에 대한 경로 익스텐트를 모두 구하고 경로 익스텐트 내의 각각의 단일 경로에 대한 경로 식별자를 찾아낸다. 그리고 나면 주어진 키값으로 중간 노드와 최종 단말 노드를 찾은 다음, 키값과 경로 식별

자의 값이 일치하는 객체 식별자만을 검색하면 된다.

예를 들어 설명하면, 도6에서 보는 바와 같이

select b from b in Books where b.chapter.section.*.mdate = "1999:11:03";의 경로인 book.chapter.section.*.mdate의 경로 익스턴트는

book.chater.section.body.mdate와

book.chater.section.subsection.body.mdate이고 해당되는 경로 식별자 (PID)는 2와 3이 된다. 따라서 "1999:11:03"으로 인덱스를 검색한 다음, 단말 노드 중에서 키값이 일치하고 PID가 2 또는 3이 되는 객체 식별자를 검색하면 o1을 찾을 수 있다.

select b from b in Books where b.chapter.*.mdate = "2000:02:01";의 경로인 b.chapter.*.mdate의 경로 익스턴트는

book.chater.section.subsection.body.mdate이고 해당되는 경로 식별자(PID)는 3이 된다. 따라서 "2000:02:01"으로 인덱스를 검색한 다음, 단말 노드 중에서 키값이 일치하고 PID가 3이 되는 객체 식별자를 검색하면 o1과 02를 찾을 수 있다.

다중 경로 인덱스는 중첩 인덱스와 같이 지정된 경로에 대해 키값과 경로의 끝에 해당하는 객체의 객체 식별자를 유지하는 방식이다. 따라서 객체들간의 참조 관계가 삽입, 삭제 또는 변경되는 경우에는 중첩 인덱스와 동일하게 삽입 또는 삭제가 일어난 객체 경로의 양 끝을 탐색하여 경로에 대한 변경 정보를 구하고 그것을 인덱스에 반영하여야 한다.

실험실시예

임의의 확장 경로식에 대해 다중 경로 인덱스를 사용한 인덱스 방법과 여러 개의 단일 경로 인덱스를 사용한 방식과의 성능을 비교하였다. 단일 경로에 대한 기존의 인덱스 방법은 여러 가지가 있지만 기존 인덱스 방법 중에서 중첩 인덱스가 검색 성능이 가장 우수한 것으로 알려져 있기 때문에, 본 실험에서는 단일 경로에 대한 인덱스 기법으로 중첩 인덱스를 사용하였으며 인덱스들의 성능을 정량적으로 분석하였다.

도8은 본 발명에 따른 실험 실시예에서 사용된 경로의 구조도로서, 경로식을 객체 데이타베이스(OODBMS)의 스키마 형태로 나타낸 것이다. 각 경로의 중간에 존재할 수 있는 클래스(엘리먼트)는 생략한 것으로 경로의 시작 클래스인 C_t와 경로의 끝을 나타내는 클래스 C_p을 나타낸 것이다. 여기서 N은 서로 다른 경로의 개수를 나타낸 것으로 도8에 나타난 경로의 끝 클래스의 개수와 동일하다.

표1은 실험에 사용된 인자들을 나타낸다. 중첩 인덱스의 경우에는 각 경로에 대해 인덱스를 하나씩 설치하게 되므로 경로의 개수 만큼의 인덱스를 사용하게 된다. 반면 다중 경로 인덱스의 경우에는 경로의 개수에 상관없이 하나의 인덱스만을 사용한다.

실험 인자	값
N	1 ~ 10
f	1, 4, 8, 16, 32, 64
c	10^4, 10^5, 10^6, 10^7
kl	4, 8, 16, 32, 64
kr	1, 10, 100
dir	F(forward), B(backward)

경로당 팬-아웃(f)은 클래스C_t로부터 클래스C_p에 이르는 임의의 단일 경로에 대해 한 객체로부터 연결되는 클래스C_t의 인스턴스의 개수(C_t에 속하는 객체의 개수)를 나타낸다. 따라서 만일 N=8이고 f=10이면 C_t의 한 객체에 대해 8개의 서로 다른 단일 경로가 존재하고 각 단일 경로에 대해 10개씩의 객체가 연결되므로 모두 80개의 객체가 연결되어 있음을 의미한다. kl은 키의 크기, kr은 중복된 키의 개수를 나타낸다.

실험은 크게 포인트 질의와 영역 질의로 구분하여 수행하였다. 포인트 질의는 주어진 키값에 정확히 일치하는(exact match)키값만을 검색하는 것이고, 영역 질의는 일정 범위의 키값에 해당하는 객체를 검색하는 것이다. 성능 비교는 각 인덱스 기법에 대해 주어진 질의를 처리하기 위한 질의 처리 비용을 기준으로 하였고, 질의 처리 비용은 질의를 처리하는 과정에서 접근해야 하는 예상 인덱스 페이지의 개수로 정의 하였다.

도9는 본 발명에 따른 실험 실시예에 있어서 경로의 개수에 대한 영역 질의의 성능비를 도시한 그래프이다.

도9는 포인트 질의에 대한 다중 경로 인덱스의 질의 처리 비용을 중첩 인덱스의 질의 처리 비용으로 나눈 것이다. 그림에서 'F-*'와 'B-*'는 각각 전방향 인덱스에 대한 성능 비교와 역방향 인덱스에 대한 성능 비교를 나타낸다. 전방향 인덱스란 인덱스의 키값이C_t의 인스턴스에 위치하는 것을 의미한다. 따라서 이 경우에는 인덱스는 검색하고자 하는 키값을 갖는C_t의 인스턴스에 연결된C_p의 인스턴스를 알려 준다. 반대로 역방향 인덱스는 키값이 C_t의 인스턴스에 위치하며 검색하고자 하는 키값을 갖는C_p의 인스턴스에 연결된C_t의 인스턴스를 검색해 준다. F(또는 B)의 뒤에 연결된 1/10/100은 kr의 값을 나타낸다. 그리고 $f=8, c=10^6, kl=16$ 으로 설정하였다. f, c, kl의 값이 다른 경우에도 성능비는 도8과 유사한 결과를 나타내었다.

도9에서 모든 경우에 대해 경로의 개수가 많아질수록 다중 경로 인덱스의 검색 성능이 중첩 인덱스의 성능에 비해 월등히 나아짐을 알 수 있다. 중첩 인덱스의 경우에는 각 경로에 대해 인덱스가 하나씩 존재하므로 주어진 질의를 처리하기 위해서는 모든 인덱스를 검색해야 하기 때문이다.

포인트 질의는 특히 인덱스의 높이에 영향을 받게 되는데, 단일 경로 인덱스를 사용하는 경우에는 여러 개의 인덱스를 방문하므로 질의 처리 비용은 인덱스 개수에 선형적으로 비례하게 된다. 반면 다중 경로 인덱스는 하나의 인덱스를 사

용하므로 경로의 개수가 늘어나더라도 인덱스의 높이는 크게 차이가 나지 않기 때문에(보통 3 ~ 5) 경로의 개수에 따른 포인트 질의 처리 비용에는 별반 차이가 없다. 따라서 경로의 개수가 늘어날수록 다중 경로 인덱스의 검색성능이 월등히 우수하게 된다.

도10은 본 발명에 따른 실험 실시예에 있어서 경로의 개수에 대한 포인트 질의의 성능비를 도시한 그래프이다.

도10은 도9와 동일한 방식으로 실험을 수행하되, 영역 질의에 대한 성능비를 나타낸 것이다. 도9와 동일하게 $f=8, c=10^6, kl=16$으로 설정하였으며, f, c, kl의 값을 다르게 한 경우에도 성능비는 그림10과 유사한 결과를 나타내었다.

도10은 영역 질의에 대해서 경로의 개수가 많아질수록 다중 경로 인덱스가 중첩 인덱스보다 상대적으로 성능이 우수함을 보여준다. 영역 질의의 경우에는 인덱스의 단말 노드(또는 페이지)의 개수가 성능에 주요한 영향을 미치게 되는데 다중 경로 인덱스와 중첩 인덱스 모두 경로의 개수가 늘어남에 따라 단말 노드의 개수가 늘어나지만 상대적으로 중첩 인덱스의 단말 노드 개수가 많이 늘어난다. 왜냐하면 다중 경로 인덱스의 경우에는 경로의 개수가 증가하면 동일한 키 값에 포함되는 경로식별자(PID)와 객체 식별자의 개수만이 늘어나기 때문에, 단말 레코드의 크기가 조금씩 증가하는 것에 반해 중첩 인덱스의 경우에는 단일 경로 인덱스의 개수가 늘어나므로 모든 단일 경로 인덱스의 단말 노드 개수를 합한 것은 다중 경로 인덱스의 경우보다 급격하게 늘어나게 되기 때문이다.

이 점은 또한 B-1의 경우에 다중 경로 인덱스가 경로의 개수에 상관없이 중첩 인덱스보다 성능이 5%정도 떨어지는 현상을 설명해 준다. 즉, B-1은 역방향 인덱스를 사용하고 키의 값이 중복되지 않는 경우로, 경로의 개수가 늘어나더라도 계속해서 새로운 키값이 추가되되므로 중첩 인덱스와 마찬가지로 다중 경로 인덱스의 경우에도 단말노드의 개수가 늘어나게 되어 상대적인 성능이 향상되지 않게 된다. 또한 부가적으로 사용하는 경로식별자(PID)와 같은 필드의 영향으로 단말 레코드의 크기가 늘어나 오히려 중첩 인덱스보다 많은 수의 단말 노드를 사용하게 되므로 그 성능이 중첩 인덱스보다 다소 떨어지게 되는 것이다.

발명의 효과

이상과 같이 본 발명에 의하면,

다중 경로 인덱스는 확장 경로식으로 표현되는 질의에 대한 검색 성능이 향상되며, 다수의 경로에 대한 인덱스를 통합적으로 제공함으로써 인덱스 관리가 용이 하다. 또한 확장 경로뿐만 아니라 개개의 경로에 대한 검색 기능을 제공할 수 있으며, 경로 식별자를 이용하는 방식에서 기존의 단일 경로 인덱스 기법에 쉽게 응용해서 사용할 수 있는 효과가 있으며, B+-트리를 크게 수정하지 않고 사용할 수 있어 실용성이 우수하다는 효과가 있다.

(57) 청구의 범위

청구항 1.

주어진 확장 경로식에 해당하는 개개의 단일 경로를 찾아내고 각각에 대해 경로식별자를 할당하는 단계;

상기 단일 경로별로 데이터베이스 내의 실제 데이터를 검색하여 인덱스 정보수집하는 단계;

상기 경로 식별자 및 수집된 상기 인덱스 정보를 다중 경로 인덱스에 저장하는 단계;을 포함하는 XML 데이터의 효과적인 검색을 위한 다중 경로 인덱스 방법.

청구항 2.

제1항에 있어서,

경로식별자를 할당하는 방법은,

경로 식별자를 할당하는 알고리즘에 완화경로식을 입력을 하는 제1단계;

상기 경로 식별자를 할당하는 알고리즘에서 경로검색(search-path)함수를 호출하여 상기 완화경로식에서 전경로의 가장 뒤쪽 엘리먼트와 후경로의 가장 앞쪽 엘리먼트 사이의 모든 경로를 탐색하는 제2단계;

상기 제2단계에서 탐색된 모든 경로와 해당 하부 엘리먼트를 합쳐서 경로 익스텐트 집합에 삽입하는 제3단계;

상기 익스텐트가 구해지고 나면 찾아진 각각의 단일 경로에 대해 경로 식별자를 할당하는 제4단계;를 포함하는 것을 특징으로 하는 XML 데이터의 효과적인 검색을 위한 다중 경로 인덱스 방법.

청구항 3.

제2항에 있어서

상기 경로 식별자를 할당하는 알고리즘은,

완화 경로식을 입력받으며, 완화 경로식의 전경로의 가장 뒤쪽 엘리먼트와 후경로의 가장 앞쪽 엘리먼트 사이의 모든 경로를 상기 데이타베이스에서 검색하며, 검색된 각각의 단일 경로에 대해 경로 식별자를 할당하는 것을 특징으로 하는 XML 데이터의 효과적인 검색을 위한 다중 경로 인덱스 방법.

청구항 4.

제1항 내지 제3항 중 어느 한 항에 있어서,

상기 다중 경로 인덱스의 레코드 구조는 헤더정보, 인덱스 키값, 서로 다른 경로의 개수, 해당 경로의 경로 식별자, 경로에 해당하는 객체 식별자의 개수 및 경로에 해당하는 객체의 객체 식별자들의 배열을 포함하는 것을 특징으로 하는 XML 데이터의 효과적인 검색을 위한 다중 경로 인덱스 방법.

청구항 5.

제1항 또는 제3항 중 어느 한 항에 있어서,

다중 경로 인덱스에 대한 검색 및 갱신 방법은,

검색 경로에 대한 경로 익스텐트를 모두 구하고 경로 익스텐트내의 각각의 단일 경로에 대한 경로 식별자를 찾는 단계;

키값으로 상기 데이타베이스의 중간 노드와 최종 단말 노드를 찾는 단계;

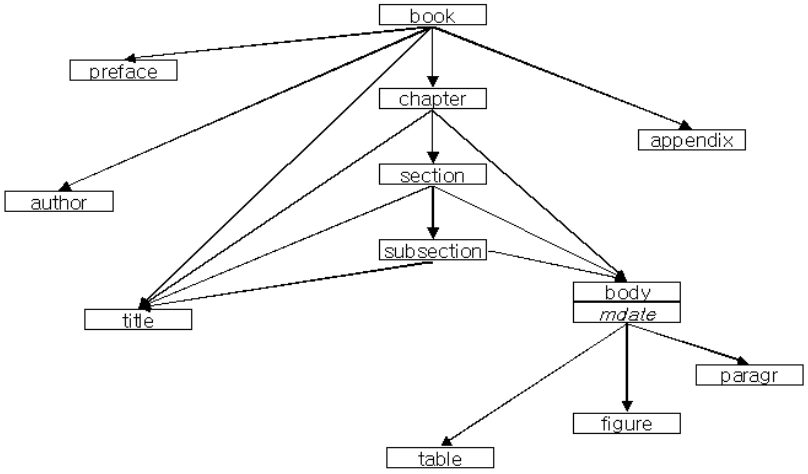
상기 키값과 상기 경로 식별자의 값이 일치하는 객체 식별자만을 검색하는 단계;를 추가로 포함하는 XML 데이터의 효과적인 검색을 위한 다중 경로 인덱스 방법.

도면

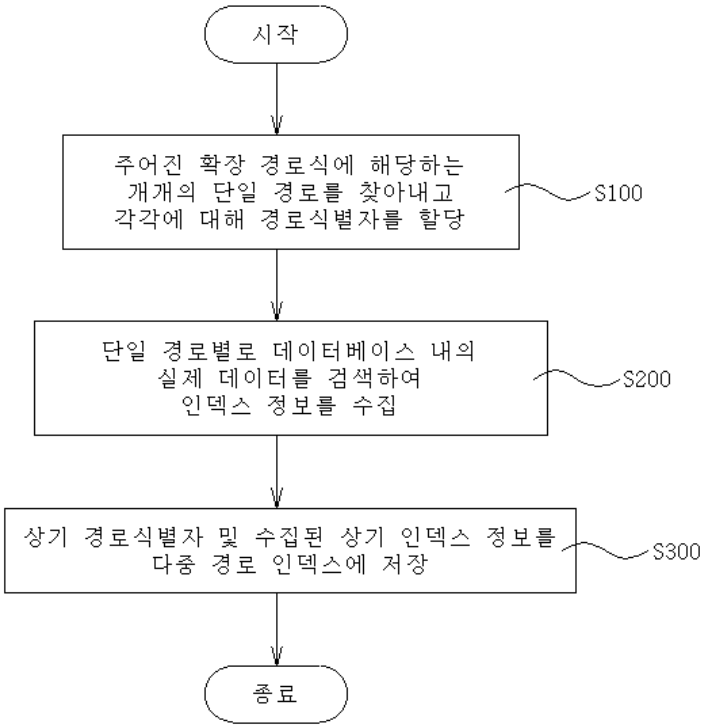
도면 1

```
<!DOCTYPE book [
<!ELEMENT book        (title,author+,preface,chapter+,appendix)>
<!ELEMENT chapter     (title,author*,body*,section+)>
<!ELEMENT section     (title,body*,subsection+)>
<!ELEMENT subsection  (title,body+)>
<!ELEMENT body        (paragr|figure|table)>
<!ATTLIST body        mdate  #CDATA>
...
```

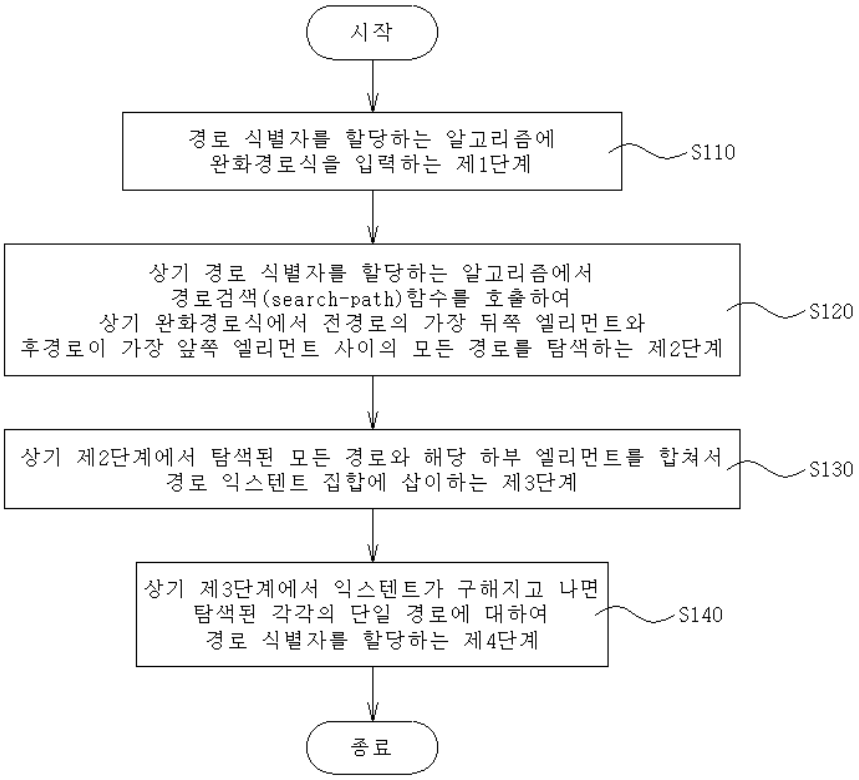
도면 2



도면 3


시작
주어진 확장 경로식에 해당하는
개개의 단일 경로를 찾아내고
각각에 대해 경로식별자를 할당
S100
단일 경로별로 데이터베이스 내의
실제 데이터를 검색하여
인덱스 정보를 수집
S200
상기 경로식별자 및 수집된 상기 인덱스 정보를
다중 경로 인덱스에 저장
S300
종료

도면 4


시작
경로 식별자를 할당하는 알고리즘에
완화경로식을 입력하는 제1단계
S110
상기 경로 식별자를 할당하는 알고리즘에서
경로검색(search-path)함수를 호출하여
상기 완화경로식에서 전경로의 가장 뒤쪽 엘리먼트와
후경로이 가장 앞쪽 엘리먼트 사이의 모든 경로를 탐색하는 제2단계
S120
상기 제2단계에서 탐색된 모든 경로와 해당 하부 엘리먼트를 합쳐서
경로 익스텐트 집합에 삽이하는 제3단계
S130
상기 제3단계에서 익스텐트가 구해지고 나면
탐색된 각각의 단일 경로에 대하여
경로 식별자를 할당하는 제4단계
S140
종료

도면 5

Algorithm 1 경로 식별자를 할당하는 알고리즘

Input: XML DTD, p : extended path expression
Output: $\varepsilon(P)$

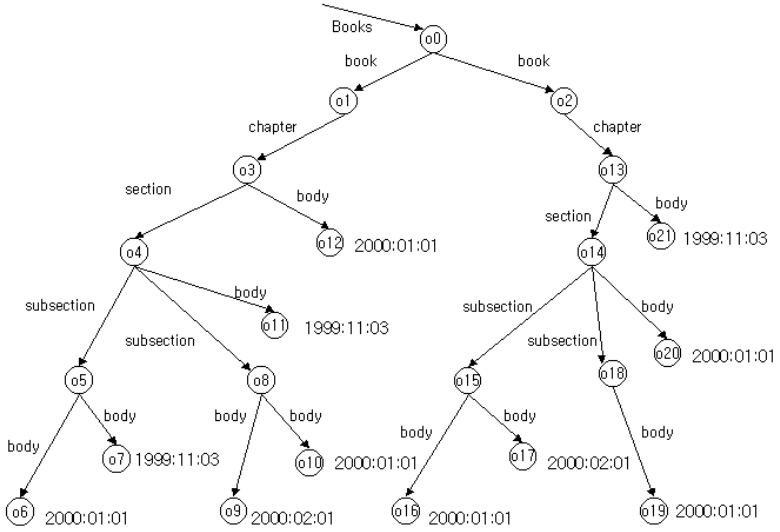
if p is a single path expression **then**
  $\varepsilon(P) \leftarrow p$
  return
endif
$\varepsilon(p) \leftarrow \emptyset$
$P_f \leftarrow pre\_path(p)$
$P_r \leftarrow post\_path(p)$
check validity of path P_f and P_r
$from \leftarrow tail(P_f)$
$to \leftarrow head(P_r)$
initialize stack
search_path($from$)
assign a serial number for each single path in $\varepsilon(P)$

procedure search path(e)
if is leaf element **then**
  return
end if
push e into stack
for all $c \in child(e)$ **do**
  if $c = to$ **then**
    $\varepsilon(p) \leftarrow \varepsilon(p) \cup \{\text{concatenate(path in stack, } c)\}$
  else if c is element **then**
    search path (c)
  end if
end for
pop stack
end procedure

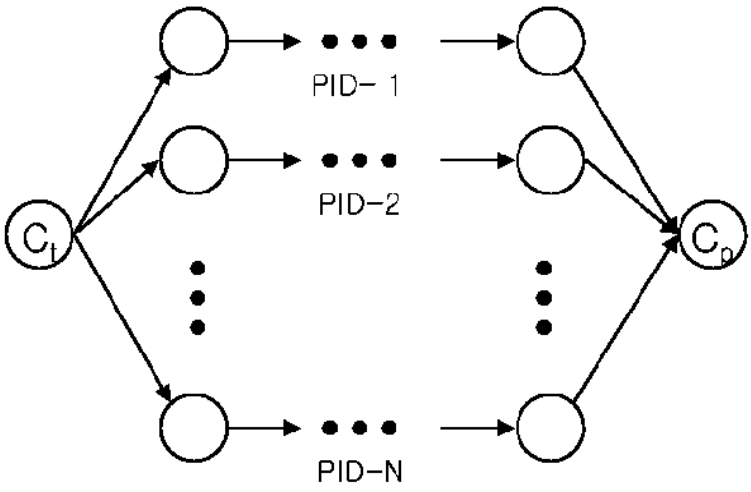
도면 6

Header	Key value	# paths	(PID1, # OIDs, OIDs)	...	(PIDn,# OIDs, OIDs)

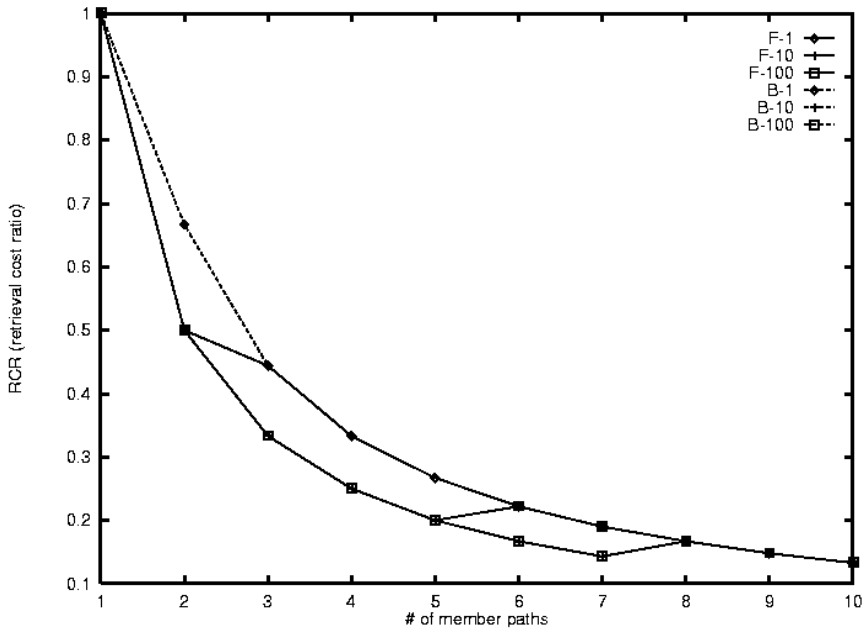
도면 7


Books
o0
book
book
o1
o2
chapter
chapter
o3
o13
section
body
section
body
o4
o12
2000:01:01
o14
o21
1999:11:03
subsection
body
subsection
body
subsection
o11
1999:11:03
o5
o8
o15
o18
o20
2000:01:01
body
body
body
body
body
body
body
o7
1999:11:03
o10
2000:01:01
o17
2000:02:01
o6
2000:01:01
o9
2000:02:01
o16
2000:01:01
o19
2000:01:01

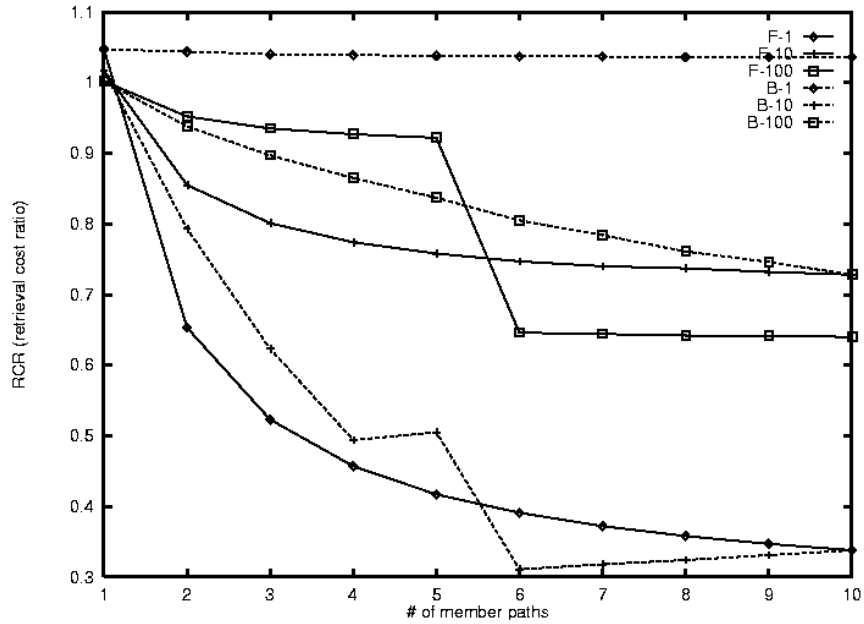
도면 8


PID- 1
PID-2
PID-N
C_i
C_o

도면 9


F-1
F-10
F-100
B-1
B-10
B-100
RCR (retrieval cost ratio)
# of member paths
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
1
2
3
4
5
6
7
8
9
10

도면 10


F-1
F-10
F-100
B-1
B-10
B-100
RCR (retrieval cost ratio)
# of member paths
1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
1
2
3
4
5
6
7
8
9
10