오디세우스/IR: 정보 검색 기능과 밀결합된 고성능 객체 관계형 DBMS

(Odysseus/IR: a High-Performance ORDBMS Tightly-Coupled with IR Features)

황 규 영 † 이 민 재 †† 이 재 길 ††† 김 민 수 †††† 한 욱 신 †††††
(Kyu-Young Whang) (Min-Jae Lee) (Jae-Gil Lee) (Min-Soo Kim) (Wook-Shin Han)

요 약 상용 ORDBMS 개발사들은 자신의 DBMS에 사용자 정의 타입과 사용자 정의 함수를 추가하는 확장 방법을 제공하고 있다. 이러한 확장은 상위 레벨 인터페이스를 사용하여 이루어진다. 이러한 기법을 소결합(*loose-coupling*)이라 부른다. 소결합의 장점은 구현하기 쉽다는 것이나, 높은 성능이 요구되는 대용량 데이타베이스에서 새로운 데이타 타입과 연산을 추가하기에는 적합하지 않다. 본 논문에서는, 이러한 요구 조건을 충족하기 위해 밀결합(*tight-coupling*)이라는 개념을 사용하는 것을 제안한다. 밀결합에서 새로운 데이타 타입과 연산은 DBMS의 엔진 내부에 통합된다. 따라서, 새로운 데이타 타입과 연산이 높은 성능으로 일관성 있게 제공된다. 이 밀결합 아키텍처는 정보 검색 기능과 공간 데이타베이스 기능을 한국과학기술원/첨단정보기술연구센터에서 개발 중인 객체 관계형 DBMS 오디세우스/IR에 통합하기 위해 사용되고 있다. 본 논문에서는, 오디세우스/IR을 소개하고 오디세우스/IR과 밀결합된 정보 검색 기능(미국 특허 등록)을 설명한다. 다음으로, 오디세우스/IR을 사용한 단일 시스템(non-parallel) 설정에서 2,000만건의 웹 페이지를 관리할 수 있는 웹 검색 엔진을 보인다.

키워드 : 정보 검색, 객체 관계형 DBMS, 밀결합

Abstract Conventional ORDBMS vendors provide extension mechanisms for adding user-defined types and functions to their own DBMSs. Here, the extension mechanisms are implemented using a high-level interface. We call this technique *loose-coupling*. The advantage of loose-coupling is that it is easy to implement. However, it is not preferable for implementing new data types and operations in large databases when high performance is required. In this paper, we propose to use the notion of *tight-coupling* to satisfy this requirement. In tight-coupling, new data types and operations are integrated into the core of the DBMS engine. Thus, they are supported in a consistent manner with high performance. This tight-coupling architecture is being used to incorporate information retrieval(IR) features and spatial database features into the Odysseus/IR ORDBMS that has been under development at KAIST/AITrc. In this paper, we introduce Odysseus/IR and explain its tightly-coupled IR features (U.S. patented). We then demonstrate a web search engine that is capable of managing 20 million web pages in a non-parallel configuration using Odysseus/IR.

Key words : Information Retrieval, Object-Relational DBMS, Tight-Coupling

· 본 연구는 첨단정보기술연구센터를 통하여 한국과학재단으로부터 지원을 받았음
† 종신회원 : 한국과학기술원 전산학과 교수/첨단정보기술연구센터 소장
kywhang@mozart.kaist.ac.kr
†† 비 회 원 : 한국과학기술원 전산학과/첨단정보기술연구센터
mjlee@mozart.kaist.ac.kr
††† 정 회 원 : 한국과학기술원 전산학과/첨단정보기술연구센터
jglee@mozart.kaist.ac.kr
†††† 학생회원 : 한국과학기술원 전산학과/첨단정보기술연구센터
mskim@mozart.kaist.ac.kr
††††† 종신회원 : 한국과학기술원 전산학과/첨단정보기술연구센터
wshan@mozart.kaist.ac.kr
논문접수 : 2004년 10월 14일
심사완료 : 2005년 3월 9일

1. 서 론

최근 들어 인터넷에 존재하는 데이타의 양이 3~4년에 10배의 비율로 급격하게 증가하고 있으며[1], 정보 검색, 공간 데이타베이스, 데이타 마이닝, 그리고 데이타 스트림과 같이 새로운 데이타 타입을 다루는 응용들이 늘어나고 있다[2,3]. 이에 따라, 데이타베이스 관리시스템(DBMS)은 이러한 새로운 응용들을 지원하도록 발전하고 있다.

주요 상용 DBMS 개발사들은 자신의 DBMS에 새로운 데이타 타입과 이에 대한 연산을 추가하는 방법을 제공한다. 대표적인 예로는 Oracle의 Cartridge[4]와 IBM DB2의 Extender[5]가 있다. 이들 방법에서 새로운 데이타 타입은 사용자 정의 타입(user-defined type)을 통해 정의되고, 이들에 대한 연산은 사용자 정의 함수(user-defined function)를 통해 정의된다. 사용자 정의 타입과 사용자 정의 함수는 DBMS에서 제공하는 상위 레벨 인터페이스를 통해 구현된다. 이러한 방식을 소결합(*loose-coupling*)이라 부른다.

소결합 아키텍처는 DBMS가 제공하는 상위 레벨 인터페이스만을 사용하기 때문에 다음과 같은 단점을 가진다. 첫째, 추가된 데이타 타입에 대한 연산이 데이타베이스 엔진 외부에서 수행되기 때문에 데이타베이스 엔진과의 통신 오버헤드가 발생한다. 둘째, 제공되는 상위 레벨 인터페이스가 100% 일반적이지 않기 때문에 추가할 수 있는 데이타 타입과 연산에 제한이 있다. 셋째, 상위 레벨 인터페이스로는 DBMS 엔진의 하위 레벨 기능들을 추가된 데이타 타입에 완전히 활용할 수 없기 때문에 이들에 대한 세부적인 동시성 제어와 파손 회복을 수행할 수 없다. 따라서, 대용량 데이타베이스에서 높은 성능이 요구될 때, 새로운 데이타 타입과 연산을 소결합 아키텍처로 구현하는 것은 바람직하지 않다.

본 논문에서는 이러한 단점들을 해결하기 위해서 밀결합(*tight-coupling*) 아키텍처[6]를 사용하는 것을 제안한다. 밀결합 아키텍처에서 새로운 데이타 타입과 이에 대한 연산은 DBMS 엔진 내부에 구현된다. 밀결합 아키텍처는 소결합 아키텍처에 비해 다음과 같은 장점을 가진다. 데이타베이스 엔진과의 통신 오버헤드가 발생하지 않고, 추가할 수 있는 데이타 타입과 연산에 제한이 없으며, 세부적인 동시성 제어와 파손 회복을 수행할 수 있다.

이러한 밀결합 아키텍처는 한국과학기술원/첨단정보기술연구센터에서 14년간 개발 중인 오디세우스/IR 객체관계형 DBMS에 정보 검색(information retrieval: IR) 기능을 통합하기 위해 사용되고 있다.[1] 밀결합 아키텍처를 통해 오디세우스/IR은 효율적인 정보 검색 질의 처리와 빠른 동적 수정 기능을 제공하며 세밀한 동시성 제어 및 파손 회복을 지원한다. 본 논문에서는 먼저 오디세우스/IR과 오디세우스/IR에 밀결합된 정보 검색 기능을 설명한다. 그리고, 밀결합된 정보 검색 기능의 우수성을 보이기 위해 오디세우스/IR로 구현된 웹 검색 엔진을 보인다. 구축된 데모 시스템(단일 시스템 설정)은 약 2,000만건의 웹 문서들을 빠르게 검색하고 관리한다.[2]

본 논문은 다음과 같이 구성된다. 제 2 절에서는 오디세우스/IR의 기능과 구조에 대해 소개하고, 제 3 절에서는 밀결합된 정보 검색 기능의 특징에 대해 소개한다. 제 4 절에서는 데모 시스템을 소개한다.

2. 오디세우스/IR 소개

오디세우스/IR은 객체 관계형 데이타베이스 관리시스템(ORDBMS)으로, 대용량 멀티미디어, 정보 검색, GIS, OLAP, 데이타 마이닝등의 최신 응용에 적합한 DBMS이다. 오디세우스/IR은 다음과 같은 특징들을 가진다.

- SQL3 표준 질의어
- 대용량 데이타베이스 지원
  - 최대 32 ZBytes(1021)의 테이블(클래스) 지원
  - 최대 8 EBytes(1018)의 레코드(객체) 지원
- 빠른 벌크 로드 및 벌크 삭제
- 동시성 제어 및 파손 회복 (미세(fine) 혹은 거친(coarse) 단위)
- 웹과의 연동 기능 제공
- 밀결합된 정보 검색 기능
  - 정보 검색 기능의 밀결합을 위해 확장된 SQL 기반 질의어
  - 빠른 동적 수정 기능
    - 정보 검색 데이타의 변경을 위한 시스템 정지가 필요 없음
  - 정보 검색 데이타에 대한 미세 혹은 거친 단위의 동시성 제어 및 파손 회복
- 밀결합된 공간 검색 기능 (개발 중)
  - 공간 검색 기능의 밀결합을 위해 확장된 SQL 기반 질의어
  - Multi-Level Grid File(MLGF)[7,8] 공간 색인을 사용
  - 공간 데이타에 대한 미세 혹은 거친 단위의 동시성 제어 및 파손 회복

그림 1은 오디세우스/IR의 아키텍처를 보인다. 이는 크게 저장시스템인 오디세우스/COSMOS와 질의 처리

1) 오디세우스/IORDBMS는 약 45만 라인의 C/C++ 코드로 구성되어 있다.

2) 병렬 설정으로 100개의 리눅스 머신에 데이타파티션함으로써 20억건까지 웹 문서들을 저장할 수 있다. 다섯 개의 리눅스 머신사용한 프로토타입이 개발되었고 성공적으로 테스트 되었다.

기인 오디세우스/OOSQL로 구성된다. 오디세우스/COSMOS는 데이타베이스에 객체들을 저장하고 관리하는 서브시스템으로 디스크 관리자(Disk Manager), 소형 객체 관리자(Small Object Manager), 대형 객체 관리자(Large Object Manager), 색인 관리자(Index Manager), 커서 관리자(Cursor Manager), 공간 DB 엔진(Spatial DB Engine), 텍스트 IR 검색 엔진(TextIR Search Engine), 파손회복 관리자(Recovery Manager), 그리고 트랜잭션 관리자(Transaction Manager)로 구성된다. 디스크 관리자는 디스크의 제어를 수행하고, 소형 객체 관리자와 대형 객체 관리자는 각각 한 페이지 크기 이하의 객체와 8 EBytes 크기 이하의 객체를 관리한다. 색인 관리자는 B+-tree, Multilevel Grid File(MLGF), 텍스트 IR 색인을 관리하고, 커서 관리자는 순차 스캔과 색인 스캔 연산을 수행한다. 공간 DB 엔진과 텍스트 IR 검색 엔진은 각각 공간 데이타와 텍스트 데이타에 대한 연산을 저장 시스템 수준에서 수행하고, 파손회복 관리자와 트랜잭션 관리자는 각각 파손 회복과 동시성 제어를 수행한다. 오디세우스/OOSQL은 SQL 질의 처리 기능을 수행하는 서브시스템으로 질의 분석기(Query Analyzer), 질의 계획 생성기(Query Plan Generator)와 최적화기(Optimizer), 그리고 질의 계획 수행기(Query Plan Executor)로 구성된다. 질의 분석기는 주어진 SQL 질의를 분석하며 질의 계획 생성기와 최적화기는 분석된 질의를 질의 계획으로 만들고 만들어진 질의 계획을 최적화 한다. 질의 계획 수행기는 최적화된 질의 계획을 오디세우스/COSMOS를 통해 수행하여 질의 결과를 가져온다.

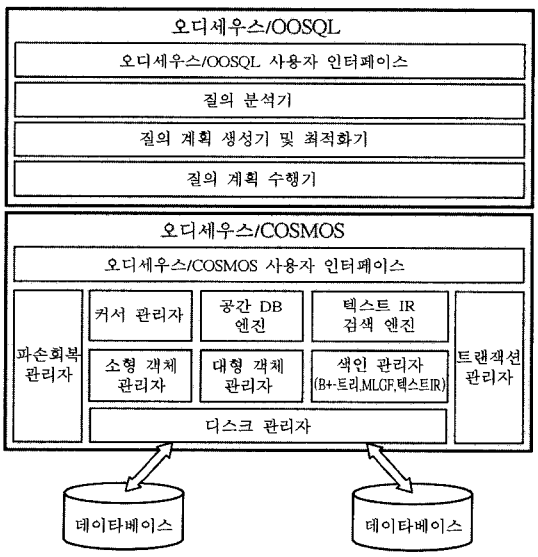

그림 1 오디세우스/IR의 아키텍처

오디세우스/IR에는 정보 검색 기능과 공간 검색 기능이 밀결합되어 있다. 정보 검색 기능의 밀결합은 구현이 완료되었으며, 공간 검색 기능의 밀결합은 구현이 진행중이다. 다음 절에서는 오디세우스/IR에 밀결합된 정보 검색 기능에 대해 자세히 설명한다.

3. 정보 검색 기능의 밀결합

오디세우스/IR은 정보 검색을 위해 웹 문서와 같은 텍스트 데이타를 저장하고, 저장된 텍스트 데이타를 키워드 조건에 따라 검색하는 기능을 제공한다. 텍스트 데이타의 저장을 위해 텍스트 타입이 지원되며, 키워드 조건에 따른 검색을 위해 텍스트 IR 색인이 지원된다. 사용자는 텍스트 타입과 텍스트 IR 색인을 비텍스트 타입과 이에 대한 색인과 동등한 수준에서 스키마 정의에 사용할 수 있다. 그림 2는 텍스트 타입, 텍스트 IR 색인, 정수 타입, 그리고 B^+-tree 색인을 사용하여 정의된 스키마를 따르는 데이타 레코드의 물리적 구조를 보인다. 그림에서 보듯이, 텍스트 타입은 정수 타입과 같은 수준에서, 텍스트 IR 색인은 B+-tree 색인과 같은 수준에서 각각 사용되었다.

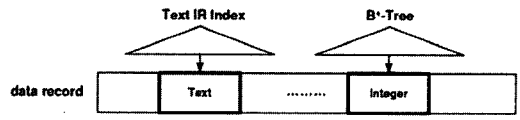

그림 2 텍스트 타입과 텍스트 IR 색인이 포함된 데이타 레코드의 구조

밀결합 아키텍처를 통해, 오디세우스/IR은 텍스트 타입의 데이타와 텍스트 IR 색인을 갱신할 때 높은 수준의 일관성을 유지하며, 동시에 향상된 질의 처리 성능을 제공한다. 특히, 오디세우스/IR은 1) 복수 키워드 질의와 2) 키워드 및 속성 조건이 동시에 명시된 질의를 빠르게 처리할 수 있다. 이러한 질의는 각각 **IR 색인 조인**(*IR index join*)과 **속성 임베딩**(*attribute embedding*) 기법을 사용하여 수행된다. 제 3.1 절에서 텍스트 IR 색인의 갱신 기능을 설명하고, 제 3.2 절에서 질의 처리 기능을 설명한다.

3.1 텍스트 IR 색인의 빠른 동적 수정

빠른 동적 수정은 오디세우스/IR의 정보 검색 기능 중에서 가장 중요한 특징이다. 전통적인 텍스트 IR 색인은 복잡한 구조를 가지기 때문에 이에 대한 동적 수정은 어렵거나 매우 비효율적인 연산으로 인식되어 왔다. 따라서, 수정이 빈번히 일어나는 환경에서는 전통적인 텍스트 IR 색인을 사용하기 힘들었다. 하지만, 오디세우스/IR에서는 **대형 객체**(*large object*)[9]와 **서브인덱스**

(*subindex*)를 사용한 역 색인 구조를 활용함으로써 동적 수정이 빠르게 처리된다. 아래에서 이 구조(미국 특허 등록[6])를 자세히 설명한다.

그림 3은 오디세우스/IR이 사용하는 텍스트 IR 색인의 구조를 나타낸다. 그림 3은 정보검색을 위해 널리 사용되는 전통적인 역 색인 구조[10]와 비슷하나, 1) 포스팅 리스트를 저장하기 위해 대형 객체를 사용한다는 점과, 2) 각각의 포스팅 리스트 별로 포스팅을 색인하기 위해 서브인덱스를 사용한다는 점이 다르다. 여기에서, 포스팅 리스트를 대형 객체로서 저장하고 Biliris[9]가 Exodus를 위해 제안한 대형 객체 트리(large object tree)를 사용함으로써 포스팅 리스트의 저장 공간을 관리한다. 이 방법의 장점은 새로운 포스팅을 포스팅 리스트에 추가하거나 포스팅을 포스팅 리스트에서 삭제하는 것이 용이하다는 점이다. 서브인덱스는 포스팅 리스트를 저장하는 각각의 대형 객체에 생성되는 B^+-tree 이다. 서브인덱스의 구조는 일반적인 B^+-tree의 구조와 동일하다. 이러한 B^+-tree에서 키는 문서 식별자이며, 리프(leaf) 노드는 포스팅 리스트 내에서 특정 문서 식별자를 가지는 포스팅이 저장된 오프셋(offset)을 가리킨다. 서브인덱스는 포스팅 리스트 내에서 주어진 문서 식별자를 가지는 특정 포스팅을 찾는데 사용된다. 서브인덱스를 사용하여, 새로운 포스팅이 삽입되거나 기존의 포스팅이 삭제 혹은 변경될 위치를 빠르게 찾아낼 수 있다.

대형 객체와 서브인덱스 없이는 포스팅의 삽입 및 삭제는 매우 비용이 큰 연산이다. 포스팅 리스트에서 포스팅이 삽입 혹은 삭제된 곳의 뒤에 위치한 전체 포스팅들은 앞 혹은 뒤로 이동되어야 한다. 이 연산은 많은 디스크 I/O를 유발하며, 따라서 시스템의 성능을 저하시킨다. 서브인덱스 없이는 포스팅을 삽입 혹은 삭제할 곳을 찾기 위해 전체 포스팅이 읽혀져야 한다. 이 연산도 많은 디스크 I/O를 유발한다. 따라서, 빠른 동적 수정을 지원하기 위해서는 대형 객체와 서브인덱스가 필수 불가결하다.

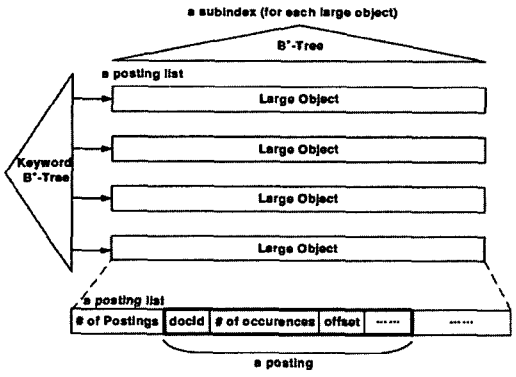

그림 3 대형 객체와 서브인덱스를 사용하는 텍스트 IR 색인의 구조

텍스트 IR 색인은 동적 수정의 성능을 향상시킬 뿐만 아니라, 질의 처리의 성능도 향상시킨다[6]. 복수 키워드 질의를 처리하기 위해 포스팅 리스트를 병합-조인(merge-join) 할 때, 실제로 조인될 포스팅 리스트의 부분만을 가려내기 위해 서브인덱스가 사용된다. 이 방법은 질의 처리 성능을 크게 향상시킨다. 이 방법에 대해서는 다음 절에서 자세히 설명한다. 따라서, 본 텍스트 IR 색인은 빈번한 동적 수정과 빠른 질의 처리에 모두 유용함을 알 수 있다.

3.2 빠른 질의 처리

오디세우스/IR은 단일 키워드 질의 뿐만 아니라 1) 복수 키워드 질의와 2) 키워드 및 속성 조건이 동시에 명시된 질의도 빠르게 처리한다. 이를 위해 각각 IR 색인 조인 기법과 속성 임베딩 기법을 제안한다[3].

IR 색인 조인(*IR index join*)은 복수 키워드 질의를 빠르게 처리하기 위한 기법이다. 복수 키워드(즉, m 개의 키워드)가 질의에 주어질 때, 해당 포스팅 리스트들은 m-웨이 병합-조인된다. 서브인덱스를 사용하여, 포스팅 리스트에서 병합될 필요가 있는 부분들을 정확하게 찾아낼 수 있다. 포스팅 리스트의 필요 없는 부분이 읽히지 않으므로 성능이 향상된다. 이러한 최적화 기법을 **포스팅 스키핑**(*posting skipping*)이라 부른다.

그림 4는 IR 색인 조인 알고리즘을 나타낸다. (설명의 편의를 위해 두 개의 포스팅 리스트를 조인한다고 가정한다.) 두 포스팅 리스트에서 같은 문서 식별자를 가지는 포스팅의 쌍을 찾는 것이 목적이므로, 현재 커서가 위치한 포스팅에 저장된 문서 식별자를 비교한다. 이때, 문서 식별자가 작은 쪽의 포스팅 리스트에서의 커서의 위치를 서브 인덱스를 사용하여 문서 식별자가 같아지는 위치로 이동시킨다. 그림 5는 '**system**'과 '**database**'의 두 키워드들을 함께 가진 문서를 찾기 위해 두 포스팅 리스트들을 병합하는 과정을 보인다. 회색으로 색칠된 부분은 문서 식별자들이 서로 같아 병합되어야 하는 부분이다. 포스팅 리스트는 문서 식별자의 값으로 정렬되어 유지된다. 이는 데이타베이스에 문서를 저장하는 순서대로 문서 식별자를 부여함으로써 쉽게 해결된다. 서브인덱스를 활용하여, 회색으로 색칠된 부분을 쉽게 찾을 수 있고 이 부분만을 읽을 수 있다. 예를 들어, 'system' 키워드의 포스팅 리스트에서 그림 4의 라인 5에 의해 문서 식별자가 *doc10*인 포스팅부터 곧바로 읽을 수 있고, 'database' 키워드의 포스팅 리스트에서 그림 4의 라인 7에 의해 문서 식별자가 *doc74*인 포스팅부터 곧바로 읽을 수 있다. Guo 등[11]과 Halverson 등[12]은 최근에 XML 데이타베이스에서의 복수 키워드

```
알고리즘  IR 색인 조인
입력: 포스팅 리스트 1, 포스팅 리스트 2
출력: 질의 결과 (문서 식별자가 동일한 포스팅의 쌍)
01  포스팅 리스트 1에서 현재 커서가 위치한 포스팅을 curPosting1이라고 한다;
02  포스팅 리스트 2에서 현재 커서가 위치한 포스팅을 curPosting2라고 한다;
03  while (curPosting1.docId != curPosting2.docId) do   /* docId는 문서 식별자임 */
04      if (curPosting1.docId < curPosting2.docId) then
05          포스팅 리스트 1에서 curPosting2.docId와 문서 식별자가 같은 (혹은 큰)
            포스팅 전까지 서브인덱스를 사용하여 스킵한다;
06      else if (curPosting1.docId > curPosting2.docId) then
07          포스팅 리스트 2에서 curPosting1.docId와 문서 식별자가 같은 (혹은 큰)
            포스팅 전까지 서브인덱스를 사용하여 스킵한다;
08  curPosting1과 curPosting2의 쌍을 결과로 반환한다;
```

그림 4 IR 색인 조인 알고리즘

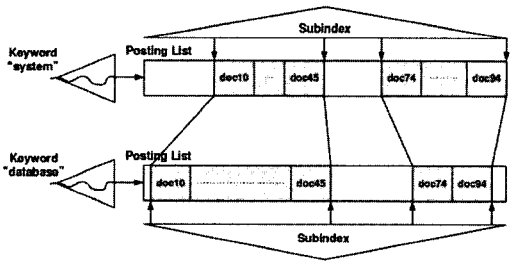

그림 5 IR 색인 조인을 사용하여 복수 키워드 질의를 처리하는 예

질의를 빠르게 처리하기 위해 비슷한 기법을 사용한다.

속성 임베딩(*attribute embedding*)은 키워드 및 속성 조건이 동시에 명시된 질의를 빠르게 처리하기 위한 기법이다. 이 기법은 문서의 속성 값이 임베드(embed)된 하나의 포스팅 리스트를 읽음으로써 키워드 및 속성 조건을 동시에 검사한다. 이 기법은 텍스트 IR 색인을 액세스하는 것만으로도 속성 값을 얻을 수 있으므로 메인 데이타베이스에 저장된 데이타 레코드를 액세스할 필요성을 없애준다. 데이타 레코드를 액세스하는 것은 방대한 저장 공간에 대한 랜덤 액세스를 유발하므로 매우 비용이 큰 연산이다. 따라서, 속성 임베딩은 성능을 크게 향상시킨다. 그림 6은 'system' 키워드를 가지면서 2004년에 작성된 문서를 찾는 과정을 보인다. 'system' 키워드를 위한 포스팅 리스트에 year 속성의 값이 임베드되어 있으므로, 질의 처리기는 포스팅 리스트를 읽으면서 각 문서가 2004년에 작성되었는지 검사할 수 있다. 사용자는 데이타베이스 스키마를 정의할 때 포스팅 리스트에 임베드되어야 하는 속성을 명시할 수 있고, 질의 처리기는 질의를 처리할 때 임베드된 속성의 값을 자동적으로 사용한다.

포스팅에 임베드된 속성 값은 데이타 레코드의 속성 값이 갱신되면 자동으로 갱신된다. 데이타베이스 카탈로그에는 속성에 대한 정보로서 임베드된 속성인지의 여부가 저장된다. 레코드의 속성 값이 갱신되면 그 속성이 임베드된 속성인지를 데이타베이스 카탈로그를 검사하여 알아낸다. 만약 임베드된 속성이면, 갱신된 레코드의 문서 식별자를 알아내어 그 문서 식별자를 키로 하는 모든 포스팅에서 임베드된 속성 값을 갱신한다. 이러한 갱신 연산을 빠르게 하기 위해, 각 문서 식별자 별로 그

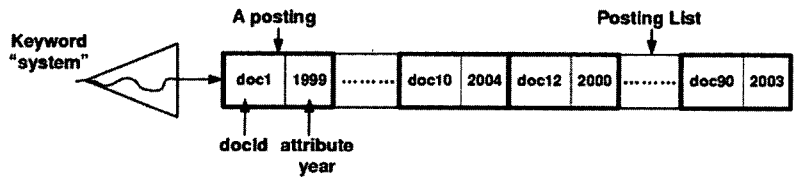

그림 6 속성 임베딩을 사용하여 키워드 및 속성 조건이 동시에 명시된 질의를 처리하는 예

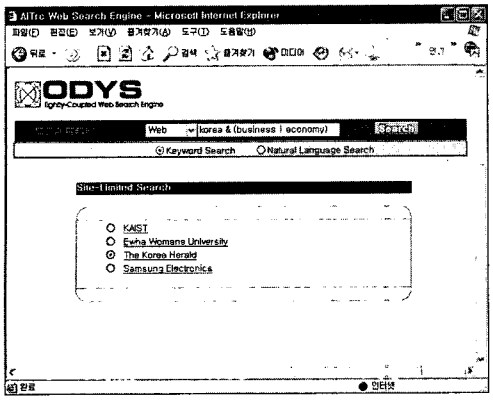

그림 7 질의 입력 페이지

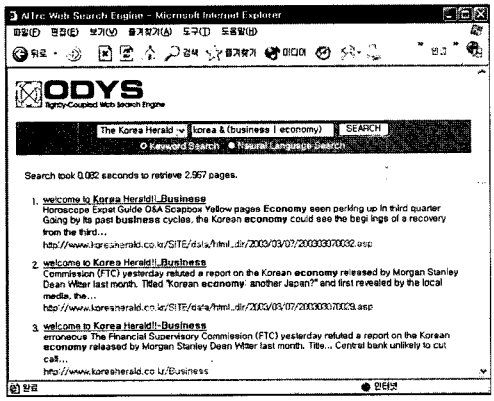

그림 8 질의 결과 페이지

문서 식별자를 키로 하는 포스팅이 포함된 포스팅 리스트들에 대한 포인터를 유지하고 있다.

4. 데모 시스템

오디세우스/IR에 밀결합된 정보 검색 기능의 우수성을 입증하기 위해, 오디세우스/IR을 사용하여 구현된 웹 검색 엔진(ODYS)을 보인다. ODYS는 400MHz CPU를 장착한 Sun Enterprise 3500 서버에서 동작하고, 550 GBytes의 Sun T3+ 디스크 어레이를 사용하여 2,000만 건의 웹 문서들을 저장한다. 웹-DBMS 게이트웨이(오디세우스/Web)를 통하여 웹 검색 인터페이스를 제공한다.

그림 7은 검색을 위해 질의 조건을 입력하는 웹 페이지를 보인다. 질의 조건으로는 복수 키워드와 임의의 불리언 연산자를 줄 수 있다. 질의 조건을 줄 때, 검색 범위를 특정 사이트로 제한할 수 있다. 이 기능을 **사이트 제한 검색**(*site-limited search*)[13]이라 부른다. 오디세우스/IR은 사이트 제한 검색을 위해 두 가지 처리 방법을 제공한다. 첫 번째 방법은 사이트 식별자를 해당 포스팅 리스트에 임베드함으로써 사이트 제한 검색 질의를 키워드 및 속성 조건이 동시에 명시된 질의로 처리하는 것이다. 즉, 포스팅 리스트를 읽으면서 특정 사이트의 웹 문서인지를 판별하기 위해 속성 임베딩 기법을 사용한다. 두 번째 방법은 사이트 식별자를 텍스트 타입으로 선언하고 텍스트 IR 색인을 구축함으로써 사이트 제한 검색 질의를 복수 키워드 질의로 처리하는 것이다. 여기에서, 사이트 식별자의 포스팅 리스트는 특정 사이트로부터 수집된 웹 문서의 문서 식별자를 저장하고 있다. 즉, 질의 키워드의 포스팅 리스트와 주어진 사이트 식별자의 포스팅 리스트를 m-웨이 병합-조인하기 위해 IR 색인 조인 방법을 사용한다. 사이트 식별자의 포스팅 리스트의 크기는 질의 키워드의 포스팅 리스트의 크기보다 매우 작으므로, 포스팅 리스트를 m-웨이 병합-조인할 때, 포스팅 스키핑 기법에 의해 많은 포스팅이 건너뛰어 진다. 따라서, 두 번째 방법의 성능이 크게 향상된다. 이 두 방법 간의 선택은 시스템 디자이너가 수동으로 혹은 질의 최적화기가 자동으로 할 수 있다. 본 데모에서는 사이트 제한 검색을 처리하기 위해 두 번째 방법을 채택한다.

그림 8은 검색 결과를 나타내는 웹 페이지를 보인다. 그림 8에서 ODYS는 Korea Herald 사이트에 대한 사이트 제한 검색의 결과로 2,967건의 결과를 82 ms 만에 얻었다. 이 결과는 밀결합된 IR 색인을 활용하는 질의 처리의 효율성을 입증하는 것이다.

참 고 문 헌

[1] Sandhya, S. M., Enterprise Storage: Answer to Backup & Recovery Blues, Cyber India Online Ltd., 2001.

[2] Banerjee, S., Krishnamurthy, V., and Murthy, R., All Your Data: The Oracle Extensibility Architecture, Oracle White Paper, Oracle Corp., Oracle Parkway, California, 1999.

[3] Whang, K., "Tight-Coupling: A Way of Building High-Performance Application Specific Engines," Presented at the panel session of *Int'l Conf. on Database Systems for Advanced Applications (DASFAA)*, Japan, Mar. 2003, available on-line from http://db-www.aist-nara.ac.jp/dasfaa2003/file/-Prof_Kyu-Young_Whang_5.pdf.

[4] Oracle, Oracle9i Data Cartridge Develope's Guide, 2002.

[5] IBM, DB2 UDB Text Extender Administration and Programming, 2003.

[6] Whang, K., Park, B., Han, W., and Lee, Y., "An Inverted Index Storage Structure Using Subindexes and Large Objects for Tight Coupling of Information Retrieval with Database Management Systems," U.S. Patent No. 6,349,308, Feb. 19, 2002, Appl. No.

09/250,487, Feb. 15, 1999.
[7] Lee, J., Lee, Y., Whang, K., and Song, I., "A Region Splitting Strategy for Physical Database Design of Multidimensional File Organizations," In *Proc. the 23rd Int'l Conf. on Very Large Data Bases*, pp. 416~425, 1997.
[8] Whang, K. and Krishnamurthy, R., Multilevel Grid Files, IBM Research Report RC 11516, 1985.
[9] Biliris, A., "The Performance Three Database Storage Structures for Managing Large Objects," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, pp. 276~285, 1992.
[10] Faloutsos, C., "Access Methods for Text," *ACM Computing Surveys*, Vol. 17, No. 1, pp. 49~74, Mar. 1985.
[11] Guo, L., Shao, F., Botev, C., and Shanmugasundaram, J., "XRANK: Ranked Keyword Search over XML Documents," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, pp. 16~27, 2003.
[12] Halverson, A., Burger, J., Galanis, L., Kini, A., Krishnamurthy, R., Rao, A. N., Tian, F., Viglas, S., Wang, Y., Naughton, J. F., and DeWitt, D. J., "Mixed Mode XML Query Processing," In *Proc. the 29th Int'l Conf. on Very Large Data Bases*, pp. 225~236, 2003.
[13] 이재길, 이민재, 김민수, 황규영, "오디세우스 객체관계형 DBMS를 사용한 사이트 제한 검색의 구현", 한국정보과학회 봄 학술발표논문집(A), pp. 752~754, 2003년 4월.

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. Editor-in-Chief: The VLDB Journal, 2003년~현재. Editor: IEEE Transactions on Knowledge and Data Engineering, 2002년~현재. Editor: The VLDB Journal, 1990년~2003년. Editor: Distributed and Parallel Databases: An International Journal, 1991년~1995년. Editor: International Journal of Geographical Information Science, 1994년~현재. Associate Editor: The IEEE Data Engineering Bulletin, 1990년~1993년. 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

이 민 재

2004년 12월~현재 (주)네오위즈 연구원 2004년 3월~2004년 11월 박사후연구원, 첨단정보기술연구센터, KAIST. 2004년 2월 한국과학기술원 전자전산학과 전산학전공 박사. 1997년 2월 한국과학기술원 전자전산학과 전산학전공 석사. 1995년 2월 한국과학기술원 전자전산학과 전산학전공 학사. 관심분야는 지리 정보 시스템, 객체 관계형 데이타베이스 시스템

이 재 길

2005년 3월~현재 한국과학기술원 BK21 박사후연구원. 1999년 3월~2005년 2월 한국과학기술원 전자전산학과 전산학전공 박사. 1997년 3월~1999년 2월 한국과학기술원 전산학과 석사. 1993년 3월~1997년 2월 한국과학기술원 전산학과 학사. 2001년 7월~2001년 8월 Visiting Scholar, 미국 HP Labs. 관심 분야는 객체 관계형 데이타베이스 시스템, 정보 검색, 질의 최적화, XML 데이타베이스, 데이타베이스 보안

김 민 수

2000년 3월~현재 한국과학기술원 전자전산학과 전산학전공 박사. 1998년 3월~2000년 2월 한국과학기술원 전산학과 석사. 1994년 3월~1998년 2월 한국과학기술원 전산학과 학사. 관심 분야는 객체 관계형 데이타베이스 시스템, 정보 검색, XML 데이타베이스

한 욱 신

2003년 3월~현재 조교수, 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