데이터베이스 관리 시스템에 기반한 웹검색엔진의 구현†

강병주, 이지동*, 최기선
한국과학기술원 전산학과
*한국오라클

Web Search Engine based on Database Management System

Byung-Ju Kang, Ji-Dong Lee*, Key-Sun Choi
Dept. of Computer Science, KAIST
*Oracle Korea

요 약

웹검색엔진은 색인되는 웹문서가 많아질수록 시스템 확장성(scalability)이라든지, 데이터베이스 유지 관리의 용이성, 데이터의 안전성 문제, 등의 많은 문제가 웹검색엔진에 부담으로 주어지게 된다. 반면에 인트라넷(intranet)용 검색엔진의 경우는 확장성보다는 검색엔진 자체의 개발의 용이성이 더욱 중요하다. Oracle ConText™ 는 오라클 社의 RDBMS 인 Oracle7™의 정보검색 확장 옵션으로 텍스트를 Oracle7 의 기본 데이터 타입으로 사용될 수 있게 한다. Oracle7+ConText 는 대용량의 문서 베이스와 개발의 용이성을 동시에 보장할 수 있는 매우 훌륭한 웹검색엔진 개발 도구이다. 우리는 이를 검증하기 위하여 Oracle7+ConText 에 기반한 WEBSECT(Web Search Engine with ConText)라는 웹검색엔진을 개발하였다. 본 논문은 WEBSECT 의 개발과 시험 운영을 통해 데이터베이스에 기반한 웹검색엔진의 우수한 확장성과 텍스트 애플리케이션 개발의 용이성 등을 소개한다.

1. 서론

웹검색엔진(Web search engine)은 World Wide Web 을 가능하게 하는 가장 중요한 기술 중의 하나이다. 하루가 다르게 WWW 에서 접근 가능한 정보의 양은 폭발적으로 증가하고 있고 이제 효과적인 검색 도구 없이 WWW 이라는 광활한 바다를 항해하는 것은 거의 불가능하다.

웹검색엔진 기술은 인터넷(internet)과 인트라넷(intranet) 모두에서 그 유용성을 입증하고 있다. 웹검색엔진은 검색 대상이 인터넷인가 또는 인트라넷인가에 따라 웹검색엔진에 요구되는 기능에 차이가 있다.

인터넷을 대상으로 하는 웹검색엔진은 많은 웹문서를 다룰 수 있는 능력이 요구된다. 참고로 대표적인 웹검색엔진인 알타비스타(AltaVista)의 경우는 약 15,000,000 건의 웹문서의 색인을 유지하고 있다 [AltaVista]. 색인하는 문서의 수가 많아질수록 시스템 확장성(scalability)이라든지, 데이터의 안전성 문제, 데이터베이스 유지 관리의 용이성, 등 많은 문제가 웹검색엔진의 부담으로 주어지게 된다. 반면에 인트라넷을 대상으로 하는 웹검색엔진은 확장성보다는 검색엔진 자체의 개발의 용이성이 가장 큰 문제가 된다. 많은 비용을 들이지 않고서도 간단하게 검색엔진을 구현할 수 있어야 하고 사용자의 특수한 요구를 쉽게 만족시킬 수 있어야 한다.

Oracle ConText™ 는 오라클 社의 RDBMS(관계형 데이터베이스 관리 시스템)인 Oracle7™의 정보검색 확장 옵션으로 텍스트를 Oracle7 의 기본 데이터 타입으로 사용될 수 있게 한다. Oracle7+ConText 는 대용량의 문서 베이스와 개발의 용이성을 동시에 보장할 수 있는 매우 훌륭한 웹검색엔진 개발 도구이다. Oracle7+ConText 는 인터넷과 인트라넷 모두에서 요구되는 웹검색엔진의 기능 요구 사항을 동시에 만족시킬 수 있다. Oracle7+ConText 는 웹검색엔진과 같은 복잡한 텍스트 애프리케이션의 개발을 간단한 PL/SQL 애플리케이션 작성으로 단순화하였다.

Oracle7 은 대용량의 데이터 관리에서 이미 그

† 본 논문은 삼성-KAIST 멀티미디어 연구 부문 협력 사업의 일환으로 실시되고 있는 멀티미디어 미래 기술 연구 과제 중 '멀티미디어 엔진에 관한 기반 기술 연구'의 일환으로 이루어졌다.

능력을 인정받은 관계형 데이터베이스 관리 시스템으로 대규모 웹검색엔진의 확장성(scalability)을 확보할 수 있고 또한 Oracle7 의 다양한 기능을 사용한다면 질적인 면이나 기능면에서 기존의 웹검색엔진들 보다 뛰어난 웹검색엔진의 구현이 가능할 것이다. 그리고 웹검색엔진의 사용자 인터페이스인 웹애플리케이션도 간단하게 PL/SQL 로 작성이 가능하기 때문에 많은 노력 없이 검색엔진의 개발이 가능하다.

우리는 Oracle7 과 Oracle ConText 의 웹검색엔진에의 적합성을 시험하기 위하여 실제로 인터넷 상의 웹문서를 대상으로 한 WEBSECT(WEB Search Engine with ConText)라는 웹검색엔진을 구현하였다. 우리는 본 논문에서 WEBSECT 의 개발 경험을 소개하고 RDBMS 에 기반한 웹검색엔진의 우수성을 입증하고자 한다.

2. 웹검색엔진 (Web Search Engine)

웹검색엔진은 World Wide Web 상의 웹오브젝트(Web object)의 내용(content)을 색인(indexing)하여 가지고 있다가 웹오브젝트의 내용에 기반한 질의가 들어 왔을 때 정보가 있을 법한 URL 과 관련 정보를 제공해주는 정보검색시스템이다. 색인의 대상이 되는 웹오브젝트는 주로 HTML 문서이고 HTML 문서의 자연언어로 쓰여진 텍스트 부분이 구체적인 색인의 대상이다.

일반적인 웹검색엔진의 시스템 구성 (그림 2)은 웹로봇, 웹로봇이 웹 공간을 돌아다니면서 URL 과 관련 정보들을 모아와서 구축된 URL 데이터베이스, 이 URL 데이터베이스를 색인하여 만들어진 색인 데이터베이스(index database), 색인기(indexer), 사용자의 질의를 받아서 파싱(parsing)하고 질의를 문서 색인과 비교하여 일치되는 URL 을 제공하는 검색엔진(search engine), 등으로 이루어 진다.

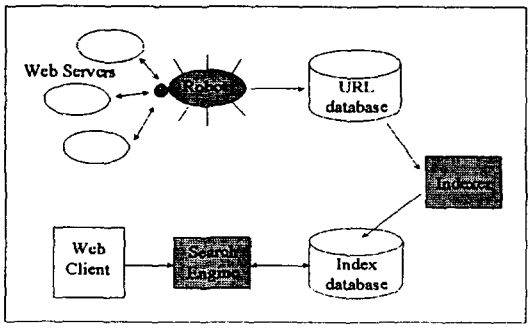

그림 2. 웹검색엔진의 구조

WEBSECT 에서는 색인기(indexer)와 검색엔진(search engine)은 ConText 를 사용하면 되고 웹로봇과 사용자 인터페이스인 웹애플리케이션만 구현하고 전체적인 시스템 통합만 하면 된다.

웹검색엔진을 구현하는 큰 단계를 보면 먼저 웹로봇의 구현, URL 과 관련 정보를 수집하여 URL 데이터베이스를 구축, ConText 를 사용하여 URL 데이터베이스 색인, 웹애플리케이션 개발, 순으로 진행된다.

3. 웹로봇(Web Robot)

웹로봇은 자동으로 World Wide Web 을 돌아 다니면서 웹검색엔진이 필요로 하는 정보를 모으는 일을 하는 소프트웨어 에이전트(software agent)이다. WWW 에 있는 웹오브젝트(Web object)에 관한 정보를 사람이 일일이 모으는 것이 불가능하므로 이 작업을 대리자(Agent)가 있어 대신 해줄 수 있다면 매우 편리할 것이다. WEBSECT 의 웹로봇은 현재 HTML 문서에 관한 정보만을 수집한다. 웹검색엔진이 완성된 후에도 갱신된 문서, 새로 추가된 문서, 삭제된 문서가 있는지 계속해서 WWW 을 탐색해야 할 필요가 있다.

WEBSECT 의 웹로봇은 자동으로 URL 데이터베이스에 수집한 URL 과 관련 정보를 삽입한다. WEBSECT 의 웹로봇은 Java[JDK]로 구현되었고 Oracle 데이터베이스를 액세스하기 위해서는 Java 와 Oracle 의 인터페이스 방법이 필요하다.

3.1 Database Access from Java

Java/Oracle 인터페이스 방법에는 Oracle WebServer Toolkit™을 사용하는 방법과 Oracle JDBC 를 사용하는 방법이 있다. JDBC 를 사용하는 방법이 보다 편리할 수 있지만 여기서는 Oracle WebServer Toolkit™을 사용하는 방법을 간단하게 설명한다.

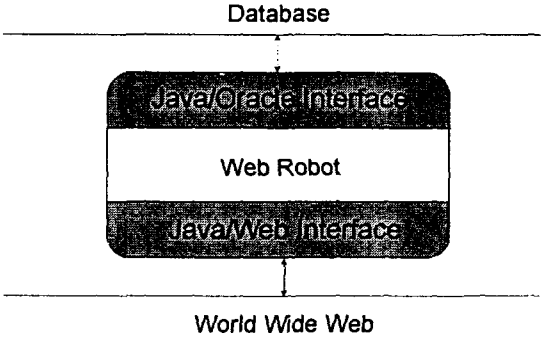

그림 3.1. 웹로봇의 구조

Oracle WebServer 에 포함되어 있는 Java Interpreter™와 Java Web Toolkit™을 사용하면 Java 에서 Oracle 데이터베이스를 액세스하는 문제는 쉽게 해결할 수 있다. Java Interpreter 는 Oracle

WebServer 의 일부로서 Java 를 번역(interpret)하여 실행하고 HTML 형태의 결과를 웹페이지의 형태로 클라이언트 브라우저에 보낸다. Java Web Toolkit 은 데이터베이스를 액세스할 수 있게 하고 Java Interpreter 를 사용하여 웹페이지를 동적으로 생성하게 하는 Java 클래스들을 제공한다.

Java Web Toolkit 을 사용하기 위해서는 Java 코드에 다음의 package 를 include 해야 한다.

- *oracle.html.** contains the objects for dynamic HTML generation.
- *oracle.rdbms.** contains the objects for database access.
- *oracle.plsql.** contains the objects for PL/SQL access.

Java Interpreter 는 PL/SQL package 를 실행함으로써 Oracle7 Server 와 인터페이스한다. Java 애플리케이션이 실행해야 할 각 package 는 *package wrapper* 를 가져야만 한다. package wrapper 는 PL/SQL package 에 포함된 프로시져(procedure)와 함수(function)를 호출하는 method 를 포함하는 Java 클래스이다. 사용자가 작성한 PL/SQL package 에 대한 package wrapper 는 *pl2java* utility 를 사용하여 생성할 수 있다 [OWSUG].

간단히 요약하면 Java 코드에서 데이터베이스 액세스를 위한 PL/SQL 프로시져를 실행해야 하는데 이를 위해 PL/SQL package 의 Java wrapper class 를 생성하여 사용한다는 것이다

3.2 목표 웹 공간 (Target Web Space)

웹로봇의 탐색 목표(target)는 한글 HTML 문서이다. 하지만 한글 HTML 문서를 찾기 위해 WWW 전체를 뒤지는 것은 매우 비효율적이다. 한글 문서는 국내 웹서버에만 있다고 가정해도 큰 무리가 없다. 이렇게 가정하면 탐색 공간을 상당히 줄일 수 있다. 그러면 국내 웹서버를 어떻게 알 수 있는가 하는 것이 문제이다. 우리가 원하는 것은 지리적으로 국내에 있는 웹서버를 탐색 공간으로 하고 싶다는 것이다. 이 문제는 그렇게 간단하지 않다. 인터넷에서의 서버의 위치는 IP 주소에 의해 계층적으로 결정되는데 이 IP 주소는 서버의 지리적인 위치와 정확히 일치하지 않는다. 따라서 다음과 같은 휴리스틱을 사용하여 국내 웹서버를 탐색한다.

- kr 도메인에 있는 웹서버는 국내에 있다. (영역 A)
- kr 도메인에 있는 HTML 문서에 있는 1 차 링크는 국내 웹서버일 가능성이 있다. (영역 B)
- kr 도메인에 있는 웹서버에로의 링크를 가지고 있는 HTML 문서는 국내 웹서버이다. (영역 C)

WEBSECT 에서는 이러한 3 가지 휴리스틱을 사용하여 어떤 웹페이지가 국내 웹서버의 HTML 문서인지 아닌지를 판단하는 알고리즘을 구현하였다. WEBSECT 가 색인 대상으로 하는 웹 부분공간 (Web subspace)은 영역 A 와 영역 C 의 합집합(union)으로 표현될 수 있다 (그림 3.2).

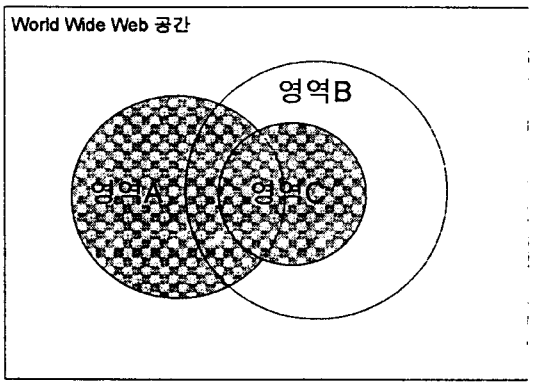

그림 3.2. WEBSECT 의 목표 웹 공간

4. URL 데이터베이스

웹검색엔진이 목적으로 하는 서비스를 제공하기 위해서 어떠한 정보들을 모을 것인가를 결정하는 것이 중요하다. WEBSECT 는 다음의 정보들을 URL 데이터베이스에 유지한다. 이들 정보는 웹로봇에 의해 자동적으로 URL 테이블에 삽입된다.

- **URL** - Uniform Resource Locator
- 제목 - 태그 <TITLE>과 </TITLE> 사이에 오는 문자열
- 문서내용요약 - 전체 내용에 대한 요약을 문서의 처음 5 줄로 대신 한다
- 문서크기(바이트 단위) - HTML 문서의 크기
- 마지막 갱신 날짜 - 나중에 문서의 갱신 여부를 확인하기 위해

가장 먼저 해야 할 일은 URL 테이블 스키마 (schema)를 생성하는 것이다. 위와 같은 목적으로 사용될 5 개의 필드(field)와 primary key 로 사용될 문서 id 필드가 정의되어야 한다. URL 필드가 텍스트 데이터 타입으로 선언된다. 실제 텍스트가 들어가는 필드지만 대신 URL 을 입력하면 나중에 색인시 해당 URL 이 가리키는 실제 웹문서를 가져와서 처리하게 된다. 따라서 보통의 텍스트 테이블에서는 텍스트 필드에 실제 텍스트 문서가 들어가지만 URL 테이블에서는 실제 웹문서 대신에 웹문서를 가리키는 주소가 들어가게 된다.

문서의 내용요약은 문서의 처음 5 줄 (공백문자를 삭제한 후)로 대신한다. 마지막 갱신 날짜는

나중에 색인 갱신 시 문서의 내용 변경 여부를 판단하기 위해 사용된다. 여기서 주목할 점은 색인어(indices)가 빠져 있다는 것이다. 키워드 추출 및 색인 구축은 WEBSECT의 로봇이 하지 않고 Oracle ConText가 URL 데이터베이스의 URL을 다시 방문하여 하게 된다.

5. 색인(Indexing)

일단 URL 데이터베이스가 만들어졌으면 다음은 색인을 구축할 차례다. 색인은 Oracle ConText를 사용한다. ConText는 URL 테이블을 처음부터 읽어가면서 각 레코드의 URL이 가르키는 HTML 문서를 가져와서 색인한다. URL 데이터베이스에는 URL만 가지고 있지 실제 HTML 문서는 가지고 있지 않기 때문에 색인할 때 다시 URL에서 HTML 문서를 가져오기 위해 해당 웹서버에의 네트워크 접속이 필요하다. 이 때문에 색인 시간이 매우 길어질 수 있고 색인에 실패할 경우도 발생한다. 따라서 네트워크 속도가 느린 사이트에서는 ConText의 URL 색인 옵션에서 URL 접속 타임 아웃(time out)을 좀 넉넉하게 잡아 주는 것이 바람직하다. 그래도 접속 당시 해당 웹서버에 대한 트래픽이 폭주하고 있다거나, 잠시 서비스를 중단하고 있는 경우, 등으로 인해 색인에 실패하는 URL들이 있게 된다. 이런 경우는 나중에 다시 실패한 URL에 대해서만 재색인(re-indexing)을 하여야 한다.

ConText는 한글 문서의 경우 형태소 분석을 통해 명사만을 추출하여 색인으로 취한다. 복합명사는 단순명사로 분리하여 색인하는 것을 원칙으로 하고 복합명사 자체도 색인으로 취한다. 예를 들어 '정보검색시스템'의 경우 추출된 색인은 '정보', '검색', '시스템', '정보검색시스템', 의 4개이다.

ConText는 색인 자료 구조로 정보검색에서 가장 많이 쓰이는 역색인(inverted index)을 사용한다. 역색인이란 각 색인별로 포스팅리스트(posting list)를 가지는 방식이다. 포스팅리스트란 색인어를 포함하는 문서 id들의 리스트이다. 그리고 ConText에서는 포스팅리스트에 문서 id와 함께 색인의 문서 내 위치정보까지 가지고 있다. 위치정보는 구(phrase) 검색, 근접(proximity) 검색, 등에 사용된다. 이러한 역색인 정보는 내부적으로 RDBMS 테이블과 Oracle 인덱스(index)의 형태로 저장된다.

6. 질의(Query)와 검색(Retrieval)

URL 데이터베이스와 URL 색인이 구축되었으면 SQL*Plus 에서 SQL 언어로 바로 텍스트 검색이 가능하다. 즉, 다음과 같은 질의가 가능하다.

```
SELECT * FROM websect
WHERE CONTAINS (url, '오라클') > 0;
```

여기서 websect는 URL 테이블의 이름이고, url은 URL 주소가 저장되어 있는 URL 테이블의 column 이름이다. 그리고 CONTAINS는 ConText Option이 있어야만 Oracle7이 이해할 수 있는 PL/SQL 함수로 WHERE 절은 url에 '오라클' 이라는 검색어가 출현하는 레코드를 찾으라는 의미이다. '> 0'를 주는 이유는 CONTAINS 함수가 0부터 100사이의 값을 출력하는 데 이 값이 0보다 커야 한다는 의미이다. 데이터베이스의 각 레코드에 대한 CONTAINS 함수 적용 결과 적합성 점수가 계산되게 되는데 0부터 100사이의 값을 가지며 검색어가 한 개라도 포함되어 있으면 0보다 큰 값을 가지게 된다. 이 점수는 결과를 순위화(ranking)하는데 사용될 수 있다.

위의 예는 One-step 질의라고 부르는데 실제 우리가 검색엔진을 개발하는 데 필요한 것은 Two-step 질의다. One-step 질의에서는 임시 결과 테이블이 내부적으로만 사용되나 Two-step 질의에서는 임시 결과 테이블을 사용자가 조작할 수 있다. Two-step 질의에서는 검색이 2 단계에 걸쳐 일어나게 되는데 1단계에서는 텍스트 검색 결과가 임시 결과 테이블로 만들어지고 2단계에서 임시 결과 테이블과 URL 테이블을 조인(join)하여 원하는 결과를 얻게 된다. 이때 결과를 적합성 점수에 따라 내림차순으로 정렬하면 순위화(ranking)가 된다. Two-step 질의에서는 텍스트 검색은 CONTAINS 저장 프로시져(procedure)로 구현된다. 따라서 검색엔진을 구성하는 PL/SQL 프로그램은 사용자의 질의를 받아서 문법에 맞게 질의을 재구성하고 텍스트 검색을 위해 CONTAINS 저장 프로시져를 호출하고 임시 결과 테이블과 URL 테이블을 조작하여 검색결과를 사용자에게 출력하는 형태로 구현된다.

사용자의 질의식(query statement)은 CONTAINS 함수나 저장 프로시져에 인수(argument)로 주어지는데 위의 One-step 질의 예제에서 '오라클'은 질의식이 한 개의 검색어로 이루어진 경우이고 '오라클' 대신에 복잡한 질의식이 올 수 있다. 다양한 검색 옵션을 사용하기 위한 질의식 작성방법은 [CTXDG]를 참조할 수 있고 ConText에서 사용 가능한 검색 옵션의 일부를 열거하면 아래와 같다

- Full Boolean Logic - AND, OR, NOT
- Score-based Arithematic - ACCUMULATE, MINUS
- Phrase Search
- Proximity Search - NEAR
- Wildcards Search
- Term Weighting, Thresholding
- Thesaurus-based Search

근접검색(proximity search)과 구검색(phrase search)은 색인의 위치정보를 사용하게 된다. 근접검색은 두 개의 검색어가 서로 가깝게 문서 내에

서 출현할 때 보다 높은 적합성 점수를 주는 방법으로 매우 유용한 검색 옵션이다. ConText 의 구검색 옵션은 복합명사 검색에 유용하게 쓰일 수 있다. 질의 '정보 검색'으로 검색하면 "정보검색", "정보 검색", 모두 검색된다. 반대로 질의가 복합명사 '정보검색'일 때도 "정보검색", "정보 검색", 모두 검색된다.

7. 웹 애플리케이션(Web Application)

URL 데이터베이스와 URL 색인의 구축이 완료되었으면 이제 사용자들과의 입.출력을 담당하는 웹 애플리케이션을 만들 차례이다. 웹애플리케이션은 사용자의 질의를 받아서 ConText 에 텍스트 질의를 하고 ConText 로부터 검색 결과를 받아 웹페이지 형태로 출력하는 PL/SQL 프로시져이다.

7.1 사용자 인터페이스(User Interface)

사용자 인터페이스는 검색 전문가와 검색 초보자 모두를 염두에 두고 설계되어야만 한다. 검색 초보자가 복잡한 검색식을 만들지 않고서도 최대한 원하는 검색 목적을 달성할 수 있도록 배려 해야만 한다. 초보자는 정보 검색에서 사용하는 질의 연산자에 익숙하지 않으므로 전문적인 질의 연산자를 사용하지 않고 질의를 만들 수 있도록 하는 것이 바람직 하다.

단순 검색(simple query) 인터페이스의 경우 대부분의 웹검색엔진은 검색어를 공백문자를 사이에 두고 나열하도록 하고 있다. 검색어 나열식 인터페이스는 검색엔진이 벡터 공간 모델에 기반한 경우이거나 아니면 검색엔진이 내부적으로 다시 질의 검색어 리스트를 가공하여 검색어들 사이에 적합한 연산자를 추가하는 식으로 문법에 맞는 질의식(query expression)으로 바꾸게 된다. 이렇게 사용자 검색어 리스트를 어떻게 해석하느냐는 웹검색엔진마다 다를 수 있다. 이러한 경우 질의가 사용자의 의도와 다르게 해석되어 사용자에게 혼란을 줄 가능성도 있다.

WEBSECT 의 단순 검색(simple query) 인터페이스에서는 3 개 까지의 검색어를 입력할 수 있고 각 검색어 사이의 검색 연산자는 NEAR, AND, OR 연산자 가운데 하나를 선택할 수 있다. NEAR 연산자가 디폴트인데 NEAR 연산자는 근접 찾기(proximity serach)를 하기 위한 연산자로 두 검색어가 문서 중에서 서로 가깝게 나타날수록 적합성 점수가 높아진다.

검색 전문가를 위해서 복잡한 검색식을 바로 입력할 수 있도록 하는 인터페이스도 제공하여야 한다. 이때 제공되는 검색 기능과 검색 연산자의 사용법을 알려주는 도움말을 필히 제공하여야 한다. WEBSECT 의 고등 검색(advanced search) 인터페이스에서는 ConText 에서 제공하는 모든 검색 옵션을 사용할 수 있다.

WEBSECT 에서는 URL 관련 정보를 테이블의 정형 데이터로 가지고 있기 때문에 관계형데이터베이스가 기본적으로 제공하고 있는 기능들을 사용하여 다양한 형태로 출력이 가능하다. WEBSECT 의 현재 사용자 인터페이스에는 구현되어 있지 않지만 URL 정보들을 데이터베이스에 저장하고 있기 때문에 다음의 기능들을 쉽게 추가 할 수 있다.

- 검색 대상 공간 선택: .co.kr, .re.kr, .ac.kr, 등
- 검색 결과 정렬: 점수, 문서크기, 마지막 갱신 날짜, 등
- 검색 결과 수의 제한: 점수, 개수, 마지막 갱신 날짜, 등

또한 ConText 에서는 시소러스 기반의 검색도 가능하다. 완성된 시소러스를 제공하지는 않기 때문에 직접 필요한 시소러스를 만들어 사용하여야 한다. 예를 들면 동의어 시소러스가 구축되어 있으면 동의어 확장을 검색 옵션으로 제공할 수 있다. 이렇게 다양한 검색 옵션을 손쉽게 구현할 수 있다는 점이 ConText 를 사용한 웹검색엔진의 큰 장점이라고 할 수 있다. 그 외에도 ConText 에서 제공하는 강력한 기타 검색 옵션이 많이 있다 [CTXDG].

7.2 Oracle Access from Web

Oracle WebServer™ 2.1 의 PL/SQL Web Toolkit 과 일종의 카트리지 역할을 하는 PL/SQL Agent™ 를 사용하면 Oracle 데이터베이스를 액세스하는 완전히 동적인 웹페이지를 쉽게 만들 수 있다.

Oracle WebServer™ 는 Oracle Web Listener™, Oracle Web Agent™, Oracle WebServer Developer's Toolkit™, 그리고 Oracle7 server 로 구성되어 있다 (그림 7.2). Web Listener 는 HTTP server 에 해당하고 Web Agent 는 CGI 프로그램과 PL/SQL 유틸리티들의 집합으로 Web Listener 로부터 요청을 받아 Oracle7 Server 에 저장되어 있는 PL/SQL 저장 프로시저(stored procedure)를 실행시키는 역할을 수행한다.

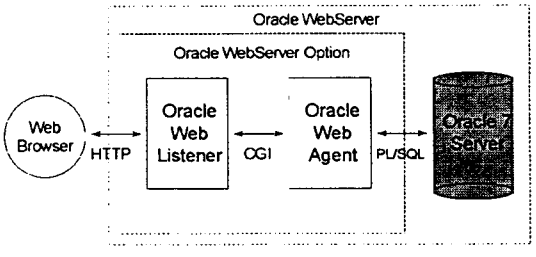

그림 7.2. Oracle WebServer

동적인 웹페이지를 만드는 간단한 예를 보자. 먼저 Web Developer's Toolkit 을 사용하여 웹페이지를 생

성하는 다음의 PL/SQL 프로시저를 만들고 데이터베이스에 저장한다.

```
CREATE OR REPLACE PROCEDURE hello (name
VARCHAR2) IS
BEGIN
  htp.title ( 'WebServer Example' );
  htp.print ( 'Hello ' || name );
END;
/
```

그런 후 웹브라우저(Web browser)로부터 위의 hello 프로시저를 실행하여 동적인 웹페이지를 생성한다.

```
http://servername:port/dcdname/owa/hell
o?name=Byung-Ju
```

라는 URL을 열면 'Hello Byung-Ju' 가 출력된다.

자세한 사용법에 대해서는 WebServer 개발자 가이드 [OWSUG]를 참조한다. 이렇게 Oracle WebServer를 사용하면 웹검색엔진의 사용자 인터페이스를 담당하는 웹애플리케이션을 간단하게 구현할 수 있다.

7.3 성능 문제

웹애플리케이션은 실제 사용자와의 인터페이스를 담당하는 부분으로 웹검색엔진에서 가장 신경을 써야 할 부분 중의 하나이다. 사용자가 실제로 웹검색엔진을 사용하면서 가장 민감하게 느끼는 부분은 질의 후 결과를 전송 받기 시작할 때 까지 걸리는 시간이다. 이 시간이 길어지면 웹검색엔진의 효용성은 크게 위축될 것이다. 질의반응시간은 검색엔진의 성능, 네트워크 상태, 등에도 영향을 받지만 웹애플리케이션의 설계에서도 심각한 영향을 받을 수 있다.

검색 결과를 전송받을때 한번에 모두를 전송받는 것은 비효율적이다. 만약 검색결과가 몇 천건이라고 가정하면 이 결과를 전송 받으려면 상당한 시간이 걸릴 것이다. 검색 결과는 적합성 순위화(relevancy ranking)가 되어 가장 적합도가 높은 순으로 정렬된다. 그리고 사용자는 맨 앞의 몇 개의 문서에만 관심이 있는 것이 보통이다. WEBSECT 에서는 이점을 고려하여 결과를 10 개 단위로 전송한다. 추가로 요청이 있을 때만 다음 10 개 결과를 전송한다.

웹검색엔진은 동시에 많은 사용자들을 서비스해야만 한다. ConText 에서는 검색 결과를 임시 결과 테이블에 저장하고 이 임시 결과 테이블과 URL 테이블을 조인(join)하여 최종 결과를 출력한다. 임시 결과 테이블을 모든 사용자가 공유할 것인지 아니면 사용자 당 고유의 임시 결과 테이블을 생성할 것인지를 결정해야만 한다 (ConText 에서 선택할 수 있는 옵션이다). 관계형 데이타베이스 시스템에서 테이블에 데이터를 쓰고 지우는 작업은 시간이 많이 걸리는 처리이다. 임시 결과 테이블을 공유한다면 다른 사용자가 테이블 사용을 끝낼 때 까지 기다려야 하고 다시 쓰기 전에 임시 결과 테이블에서 모든 데이터를 삭제해야만 한다. 반대로 임시 테이블을 따로 생성한다면 테이블 생성하는 데 시간이 걸리고 많은 디스크 공간을 소모한다. 하지만 검색 결과를 계속 유지하므로 다음 페이지(다음 10 개 결과)를 요청했을 때 다시 검색할 필요가 없다.

위의 임시 결과 테이블을 사용하는 경우는 모두 물리적으로 디스크에 테이블을 생성하기 때문에 결과 테이블을 처리하기 위해서는 많은 디스크 액세스가 필요하다. ConText 에서는 In-Memory query 라는 기능이 있는 데 이는 결과 테이블을 메모리 상에 생성해서 검색 시간을 대폭 줄여준다. 따라서 In-Memory query 를 사용하고 임시 결과 테이블을 공유하면 최상의 성능을 얻을 수 있다. 물론 이때는 검색 결과를 유지하지 못하기 때문에 추가 페이지 요청이 있을 경우 다시 검색하여 결과를 생성하여야만 한다.

사용 가능한 자원의 한계 그리고 사용자의 검색반응시간을 적절히 유지하기 위해 최대 동시 사용자에 제한을 둘 필요가 있다. ConText 에서는 병렬 질의(parallel query) 을 사용할 수 있다. 특히 프로세서가 2 개 이상 있을 경우 많은 성능 개선을 기대할 수 있다. 웹검색엔진에서는 동적인(dynamic) 웹페이지의 사용이 필수적인데 동적인 웹페이지는 CGI-bin 프로그램을 실행하여야 하기 때문에 시간이 많이 걸린다. 따라서 될 수 있으면 정적인(static) 웹페이지로 처리할 수 있는 것은 그렇게 해야만 한다. 그 외 색인 최적화, Oracle 데이터베이스 튜닝, 등을 통해 성능 개선을 도모할 수 있다.

8. 전체 시스템 구조 (Overall System Architecture)

WEBSECT 의 전체적인 시스템 구성 (그림 8)은 웹브라우저로부터 사용자의 질의를 받는 Web Listener, Web Listener 가 질의을 실행하기 위해 호출하는 PL/SQL Agent, PL/SQL Agent 로부터 요청받은 PL/SQL 저장 프로시져를 실행하는 Oracle7 Server. 그리고 마지막으로 Oralce7 Server 로부터 요청받은 텍스트 질의를 수행하고 URL 색인 데이터베이스를 검색하는 ConText Server 로 구성된다.

Web Listener 는 HTTP 서버에 해당한다. 웹브라우져로부터 사용자의 텍스트 질의를 받아 이를 PL/SQL Agent 에 넘긴다. PL/SQL Agent 는 데이터베이스에 저장되어 있는 PL/SQL 저장 프로시져의 수행을 Oracle7 에 요청하는 역할을 수행하는 CGI-bin 프로그램이다. PL/SQL Agent 는 데이터베이스와의 접속(connection)도 담당한다. 여기서 PL/SQL 저장 프로시져는 사용자의 질의를 수행하고 검색 결과를 HTML 문서 형식으로 출력하는 프로그램이다.

Oracle7 서버는 요청된 저장 프로시져를 수행한다. 이때 Oracle7 서버는 텍스트 질의는 스스로 해결할 수 없으므로 텍스트 질의를 ConText 서버에 넘기게 된다. ConText 서버는 질의를 수행하고 검색 결과를 임시 결과 테이블에 출력한다. Oracle7 서버는 임시 결과 테이블과 URL 테이블을 조인(join)하여 최종 검색 결과를 HTML 문서의 형식으로 만든 다음 PL./SQL Agent 에 보낸다. PL/SQL Agent 는 Web Listener, Web Listener 는 Web Client 에 결과를 보낸다.

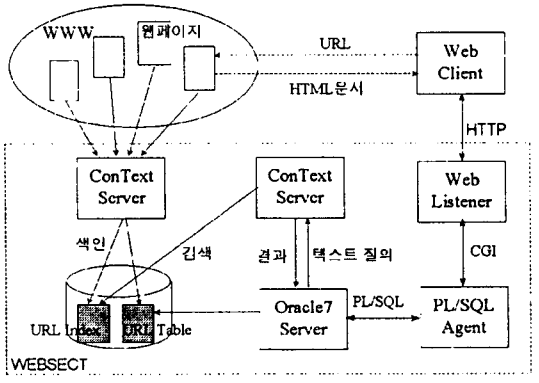

그림 8. WEBSECT 전체 시스템 구조

9. 결론

WEBSECT 의 웹애플리케이션을 구성하는 PL/SQL 코드 라인 수는 채 500 라인이 안된다. 실제로 웹애플리케이션 부분을 개발하는 데는 많은 노력이 들지 않았고 대부분의 시간은 Oracle7, ConText, WebServer 를 설치하고 웹애프리케이션이 제대로 동작할 수 있는 환경을 만드는데 소요되었다. 따라서 이미 이러한 환경이 셋업되어 있다면 인트라넷용 웹검색엔진은 간단하게 개발, 설치될 수 있다. URL 데이터베이스를 구축하는데도 인트라넷 상에 많은 웹문서가 있지 않다면 그다지 많은 시간을 요하지는 않을 것이다.

WEBSECT 가 인터넷을 대상으로한 웹검색엔진으로서도 경쟁력이 있는지는 아직 결론을 내릴 수 있는 단계는 아니다. 수십만 건 이상의 웹문서의 색인을 가지고 많은 동시 사용자를 대상으로 서비스가 본격적으로 개시되어야지만 WEBSECT 의 성능을 객관적으로 시험해 볼 수 있을 것이다. 하지만 Oracle7 의 대용량 데이터 관리 능력을 볼 때 색인되는 웹문서의 수가 늘어날수록 경쟁력은 강화되리라고 생각된다. 현재 10 만개 이상의 URL 과 관련 정보를 수집하였고 그 중 3 만 건 정도가 색인되어 있다. 앞으로도 계속 URL 수집을 해서 일차적으로 50 만 건 이상을 목표로 하고 있다.

URL 을 모으는 일은 URL 데이터베이스와 색인을 유지하고 갱신하는 것에 비하면 쉬운 일이다. 계속적인 색인의 갱신을 위하여 URL 모니터링 로봇이 필요하다. URL 모니터링 에이젼트는 24 시간 365 일 WWW 을 돌아 다니면서 URL 데이터베이스에 있는 URL 들을 정기적으로 방문하여 내용이 갱신되었는지 확인하고 새로운 URL 이 발견되면 추가하고 더 이상 존재하지 않는 URL 은 삭제하는 일을 하게 된다. 이렇게 문서의 추가 삭제가 빈번하게 일어날수록 WEBSECT 의 경쟁력은 강화될 것이다.

그리고 웹검색엔진에서 무엇보다도 사용자에게 가장 민감한 부분은 질의를 서버미트(submit)한 후 웹부라우져(Web browser)가 결과를 전송 받기 시작하기까지 걸리는 반응 시간(response time)이다. 데이터베이스에 기반한 검색엔진이 정보검색 전용으로 개발된 검색엔진에 비해 처리 속도가 떨어질 수 밖에 없다는 지적이 있어왔다. 하지만 데이터베이스 기술의 진보로 그리고 하드웨어 기술의 발전으로 이러한 속도 성능도 곧 정보검색 전용의 검색엔진과 비슷한 수준으로 향상될 수 있을 것이라고 생각된다.

WEBSECT 의 성능에 영향을 미치는 요인에는 크게 운영시스템 튜닝, Oracle 데이터베이스 튜닝, Oracle ConText 튜닝, 웹애플리케이션 설계, 네트워크 상태, 등이 있을 수 있다. 따라서 다른 독립형 웹검색엔진에 비해 추가로 고려되어야 할 부분이 데이터베이스와 ConText 튜닝인데 [CTXAG] 이 부분은 반드시 Oracle DBA 와 상의하여 최대의 성능이 나올 수 있도록 세심한 튜닝이 필요하다.

앞으로 Oracle 에 이어 많은 외국 DB 전문 업체들이 정보검색을 통합한 데이터베이스 시스템을 내놓을 것으로 예상되고 있다. 따라서 국내에서도 데이터베이스에 기반한 정보검색에 많은 관심이 요구된다.

참고 문헌

[AltaVista] The AltaVista.
http://www.altavista.digital.com/

[OWAS] "Oracle Web Application Server Data Sheet". Oracle Corp., Redwood Shores, CA, 1997.

[OWSUG] "Oracle WebServer 2.0 User's Guide". Oracle Corp., Redwood Shores, CA, 1996.

[CTXAG] "Oracle ConText Option 2.0 Administrator's Guide". Oracle Corp., Redwood Shores. CA, 1996.

[CTXDG] "Oracle ConText Option 2.0 Developer's Guide". Oracle Corp., Redwood Shores. CA, 1996.

[JDK] Java Development Kit 1.1.1. Sun Microsystems Inc., Mountain View, CA, 1996.