# WebDBs: 사용자 중심의 웹 검색 엔진

정회원 김 홍 일*, 임 해 철**

# WebDBs: A User oriented Web Search Engine

Hong-il Kim*, Hae-chull Lim** *Regular Members*

요 약

본 연구에서는 SQL과 유사한 질의어를 사용하여 웹에 등록된 정보를 검색하는 시스템인 WebDBs(Web DataBase system)를 제안한다. 제안된 시스템에서는 웹에 산재되어 있는 HTML 문서로 부터 검색에 필요한 정보들을 자동으로 추출한다. 추출된 자료에 대하여 SQL 기반의 질의 처리가 가능하도록 하였다. 웹 데이타베이스 시스템에서는 대부분의 질의 수행 시간이 통신 회선을 통한 문서 획득에 소요된다. 따라서, 웹 검색의 경우 웹 지역성에 많이 의존한다는 점에 착안하여, 사용자 검색 결과를 캐쉬에 저장하고 유사한 응용에서 이를 재사용 하고자 한다. 이때 캐쉬에 저장된 정보들을 검색된 질의와 연관하여 저장함으로써 좀더 사용자 응용을 고려한 캐쉬 관리 기법을 제안하였다. 또한 위와 같은 개념에 입각한 웹 검색엔진을 구현하였다.

ABSTRACT

This paper propose WebDBs(Web Database system) which retrieves information registered in web using query language similar to SQL. This proposed system automatically extracts information which is needed to retrieve from HTML documents dispersed in web. Also, it has an ability to process SQL based query intended for the extracted information. Web database system takes the most of query processing time for capturing documents going through network line. And so, the information previously retrieved is reused in similar applications after stored in cache in perceiving that most of the web retrieval depends on web locality. In this case, we propose cache mechanism adapted to user applications by storing cached information associated with retrieved query. And, Web search engine is implemented based on these concepts.

## I. 서 론

인터넷이 국내·외적으로 빠른 속도로 보급되면서, 세계에서 가장 많은 정보를 포함하는 데이타베이스로 성장하고 있다. 이에 따라서 인터넷에 등록되는 자료가 증가되면서 당연하게 등장하는 문제점으로 사용자가 자신이 원하는 정보를 얻는 방법이 점점 더 복잡해졌다는 것이다.[1,2,3] 이러한 문제점을 해결하기 위하여, 전 세계적으로 각종 웹 검색 엔진(web search engine)이 보급되어 검색의 편이성을 제공하고 있다. 그러나 각 검색 엔진들이 가지는 검색 방법이 서로 다르기 때문에 사용자가 이를 충분하게 숙지하지 않을 경우 검색이 용이하지 않다는 문제점이 있다. 더욱이 각 검색 엔진들은 단지 해당 검색 엔진에 등록된 url에 대한 검색만을 제공하고, 등록된 자료들을 주기적으로 갱신하기 때문에 지속적으로 데이터가 축적되고 갱신되는 인터넷의 특성을 충분히 반영하지 못하고 있는 실정이다. 또한, 현존하는 검색 엔진의 종류와 수가 너무 많고, 사용자들은 이들 중 자신에게 친숙한 몇몇 검색 엔진만을 이용하는 경향이 있다. 따라서, 사용자가 원하는 정보를 원활하게 획득하기 위해서는 표준화된 검색 방법이 절실하게 요구된다.

본 연구에서는 데이터 검색시에 사용되는 SQL과 유사한 질의어를 사용하여 웹에 등록된 정보를 검색

* 대진대학교 컴퓨터공학과 (hikim@net.daejin.ac.kr),
논문번호 : 99064-0218, 접수일자 : 1999년 2월 18일
** 홍익대학교 컴퓨터공학과

하는 시스템인 WebDBs(Web DataBase system)를 제안하였다. 제안된 시스템에서는 웹에 산재되어 있는 HTML 문서들에 대하여 가상의 스키마를 구성하고, 인터넷 로봇을 이용하여 검색에 요구되는 인스턴스들을 수집한다. 수집된 자료에 대하여 SQL 기반의 질의어 처리가 가능하도록 하였다. 따라서 웹에서 정보를 검색하고자 할 때 좀더 정제된 검색 방법을 제공한다.

사용자가 데이타베이스에 질의를 하기 위해서는 전체 데이타베이스에 대한 스키마가 존재하여야 함에도 불구하고 실제로는 웹 전체 자료들에 대한 잘 정의된 스키마가 존재하지 않는다. 따라서 HTML 문서로부터 검색에 필요한 정보를 축출하여 웹에 대한 가상 스키마(virtual schema)를 제안한다.

사용자는 웹을 하나의 거대한 데이타베이스로 보고 HTML 문서를 기반으로 한 스키마를 참조하여 검색 질의어를 작성하며, 웹 데이타베이스 관리 시스템은 해당 질의에 대한 결과를 전달하는 형태로 질의 수행을 진행한다.

이때, 검색 대상이 되는 HTML문서는 전세계에 걸쳐 분산되어 있고, 해당 자료에 대한 질의어를 수행하기 위해서는 인터넷 로봇이 해당 자료를 수집한다. 따라서 질의 수행 속도가 현저하게 저하되는 문제점이 있다. 본 연구에서는 이러한 문제점을 고려하여, 자주 사용되는 검색 형태를 뷰(view)로 선언하고, 기존에 선언된 뷰를 이용한 검색을 실행하는 방안을 제안한다. WebDBs에서의 뷰는 일반적인 데이타베이스 시스템의 뷰와는 달리 뷰 선언문과 함께 해당 인스턴스들을 지역 데이타베이스에 함께 저장하도록 설계하였다. 따라서 뷰를 이용한 검색은 통신 회선을 통하여 자료를 추출하는 과정이 생략되므로 질의 수행에 현저한 성능 향상을 이룰 수 있었다. 또한 뷰와 인스턴스를 미리 추출하는데서 발생될 수 있는 데이터 일관성 문제를 완화시키기 위하여 뷰에 의해서 추출된 자료는 물론이고 일반적인 질의 수행에서 검색된 자료들도 주기적으로 갱신하여 준다. 그리고 위에서 저장된 데이터들을 웹 질의 지역성(web query locality)을[3] 고려한 캐쉬 관리 기법을 이용하여 관리한다.

위에서 제안한 WebDBs 시스템은 Windows 95/98 운영체제 환경에서 실행될 수 있는 "Visual Studio" 환경으로 구현되었다. 구현된 WebDBs는 전용 웹 브라우저를 탑재하여 사용자가 편리하게 검색 결과를 이용할 수 있게 하였으며, 기존의 상용 웹 브라우저와 동시에 실행할 수 있도록 설계하였다.

## Ⅱ. 관련 연구

현재 상용되는 대부분의 웹 검색 엔진들은 검색 서버에 url 정보를 저장하고 사용자의 검색 요구에 응답하는 인덱스 서버들이 주류를 이루며, 클라이언트 환경에서 운영되는 검색 로봇들이 상용되고 있다. 일부의 경우, 정보는 서버에 저장하고 사용자에게 이를 검색하는 편이성을 제공해 주는 클라이언트를 제공한다. 웹 데이타베이스는 사용자측에 검색에 필요한 정보를 저장하고 이에 대한 질의를 수행하는 형태로 검색을 수행하는 인터넷 검색의 새로운 영역이다. 기존의 웹 검색에 대하여 살펴보면 다음과 같다.

### 1. 웹과 검색 엔진

1992년 인터넷에 웹이 등장한 이래 많은 사용자들이 웹에 자료를 등록함에 따라 웹은 점차로 거대한 데이터 창고가 되었고, 이에 따라 웹에서 자료를 신속하고 효율적으로 검색하는 방법이 점점 문제가 되어왔다. 1993년 미국 콜로라도 대학에서 인덱스 서버를 처음 제안한 이후로 국내·외적으로 많은 검색 엔진들이 개발되어 사용되고 있다.[1] 그러나 기존의 인덱스 서버 방식의 검색 엔진은 다음과 같은 문제점을 가지고 있다.

- 서로 상이한 검색 방법을 제공하며, 사용 방법이 다르기 때문에 사용자들이 이를 이용하기가 불편하다.
- 거의 대부분의 검색 엔진들이 자신의 데이타베이스에 등록된 자료들에 대한 검색만을 제공하기 때문에 인터넷의 일부분에 대한 검색만이 가능하다.
- 대부분의 사용자들이 자신의 url에 대한 정보를 검색 엔진에 등록할 때 보유하고 있는 자료들 중 일부분(주로 중요한 내용만)에 대한 자료만을 등록하기 때문에 실제 보유하고 있는 자료에 대한 명세를 충분하게 반영하지 못한다.
- 검색 엔진들은 등록된 자료에 대하여 일정 기간을 주기로 하여 갱신하기 때문에 지속적으로 변화하는 정보를 정확하게 알려주지 못한다.

위와 같은 문제점은 검색 엔진이 다양한 사용자의 요구와는 관계없이 검색 엔진에 등록된 자료들을 유지하는 것에서 비롯된다. 검색의 또 다른 방법으로는 검색 로봇의 이용이 있다. 검색 로봇을 이용한 검색의 경우 사용자가 원하는 정보를 추출하는 알고리즘

을 탑재하고 있기 때문에 인덱스 서버 방식 보다 사용자의 검색 요구를 충족시켜 준다는 장점이 있다. 그러나 매번 검색마다 이에 합당한 검색 로봇을 설정하여야 하는 문제점이 있고, 기존에 개발된 검색 로봇들은 기능이 단순하거나, 업체 나름의 사용 방법들이 다양하기 때문에 다양한 사용자의 요구를 충분하게 만족시켜 주지 못한다는 문제점을 가지고 있다. 따라서 사용자의 다양한 요구를 만족시켜 줄 수 있으면서 검색의 일관성을 줄 수 있는 방법들에 대한 연구가 절실하다.

### 2. 웹 데이타베이스

1995년 이스라엘의 Shmueli는 웹 전체를 하나의 거대한 그래프로 보고 SQL과 유사한 질의를 할 수 있도록 구성된 W3QS를 제안하였다.[2] W3QS에서는 웹을 하나의 거대한 그래프로 보고 이를 로봇을 이용하여 탐색하는 형태를 가진다. 이때 로봇의 탐색 기준을 SQL과 유사한 질의어로 정의하는 특성을 가지고 있다. W3QS는 최초로 질의어를 이용한 웹 검색이지만, 그래프 환경에서의 검색이기 때문에, 웹 데이타베이스에 대한 스키마 구성이나 전체 시스템의 운영 방법이 기존 데이타베이스 시스템과는 차이가 있다. 1997년 캐나다의 Mendelzon과 Milo는 처음으로 HTML 문서에 대한 스키마를 구성하고, 이를 이용한 SQL 질의를 수행하는 WebSQL 시스템을 제안하였다.[3] WebSQL에서는 HTML문서로 부터 아래와 같은 형태의 튜플들로 구성되는 스키마를 유도하였다.

*tuple: Document [url, title, text, type, length, modif]*

*tuple: Anchor [base, href, label]*

여기서 Document 튜플의 url은 해당 HTML문서의 url 값이고 키로 사용되며, title은 <HEAD> 부분에 작성된 문서 제목, text는 HTML 문서 내용 원문, type은 문서 유형, length는 해당 문서의 크기, modif는 최종 갱신 날짜와 시간으로 정의하였다. Anchor 튜플에서 base는 해당 링크가 있었던 문서의 url, href는 연결될 문서에 대한 url, label은 해당 링크에 대한 설명부분으로 분류하였다. WebSQL은 전형적인 웹 데이타베이스 구조를 가진다. 그러나 제안된 테이블의 구조가 비교적 간단하게 구성되어 있기 때문에 질의어를 수행하고자 하는 경우, 위와 같은 단순한 구조의 튜플들로는 사용자가 요구하는 의미를 충분하게 전달하지 못한다. 또한 튜플 구조의 단순성은 자세한 검색 조건의 명시가 어렵기 때문에 질의 실행 성능도 현저하게 떨어지는 문제점을 가지고 있다.

## Ⅲ. 질의어와 웹 모델링

웹은 방대한 양의 정보를 저장하고 있는 분산 데이타베이스 환경으로 볼 수 있다. 이에 대한 데이타베이스 시스템을 구성할 경우 아래와 같은 사항들을 고려하여야 한다.

- 잘 정의된 스키마를 설계할 DBA가 없다
- 데이터가 널 값을 가질 수 있고 수시로 변화할 수 있다.
- 인터넷 로봇을 통하여 획득하는 자료들은 현재로서는 한정된 정보만을 자동으로 추출할 수 있다.

따라서 웹 전체에 대한 모델링은 현실적으로 불가능하다. 이에 반하여 HTML 문서는 표준적인 표현 규칙을 따르는 준 구조적(semi-structured) 문서이다. 본 연구에서 HTML 문서로부터 아래와 같은 테이블을 정의하였다.

### 1. 가상 스키마

HTML 문서는 크게 <head>와 <body>로 구분되며, 이를 분석하면 아래와 같은 튜플들을 자동으로 추출할 수 있다. 본 연구에서는 이렇게 추출되는 튜플들을 이용하여 아래와 같이 세가지 테이블을 구성한다.

#### 1.1 문서(document) 테이블

웹에 등록된 각 HTML 문서는 아래와 같은 튜플 형태로 저장된다. url, title, text, type 에 대해서는 2장에서 설명된 바와 동일하다. path는 현 문서까지 추출되기까지의 거쳐온 HTML 문서의 경로명(path name)으로, 중복된 검색이나 연관성을 확인하기 위하여 추가하여 튜플 document를 아래와 같이 정의한다.

*tuple: document [url, title, text, type, path]*

#### 1.2 앵커(Anchor) 테이블

앵커는 HTML에서 다른 문서의 연결로 사용되며 각 앵커에 대한 문자열 형태의 label 값을 부여할 수 있다. 웹의 HTML 문서는 사용자에게 그래픽 화면으로 제공된다. 따라서 일부 사이트의 경우 아래와 같이 앵커를 그래픽 아이콘 형태로 처리한 경우가 있다. 이들 앵커에는 레이블 부분 대신 이미지 정보의 "ALT" 필드에 해당 링크에 대한 설명 정보를 저장

한다. 본 연구에서 이들 자료를 검색 자료로 활용하기 위하여 함께 정의한다.

*tuple: anchor [base, href, label, alt]*

### 1.3 소제목(Heading) 테이블

HTML 문서의 <BODY> 부분에서 중요한 제목은 <H1> 또는 <H2> 와 같은 태그로 구성하는 경우가 많다. 일반적으로 많이 사용되는 AltaVista 검색 엔진[10]에서도 이들 정보 중 문자열을 150자 까지 검색의 자료로 활용하고 있다. 본 연구에서는 <H> 태그로 강조된 문자열을 텍스트 형태로 보관하여 검색의 참고 자료로 활용하고자 아래와 같이 튜플 Heading을 정의한다.

*tuple: heading [base, String]*

위에서 제안한 테이블들은 일반적인 데이타베이스 측면에서는 매우 단순한 구조이다. 그러나 본 연구에서 개발하고자 하는 시스템은 어떤 사이트를 찾는 특정 응용에 한정되기 때문에, 검색 엔진의 데이타베이스로서 사용이 가능하다.

위에서 제안된 튜플 구조를 토대로 하여, 웹에 등록된 모든 HTML 문서는 검색 알고리즘을 따르는 인터넷 로봇을 통하여 자동으로 추출될 수 있고, 제안된 튜플들로 구성된 테이블로 볼 수 있다. 여기서 일부 애트리뷰트들은 널(Null) 값을 가질 수도 있다. 상기와 같은 테이블들을 그림으로 구성하면, HTML 문서는 그림 1.과 같이 구성된 데이타베이스 정보로 변환하여 표시할 수 있다.

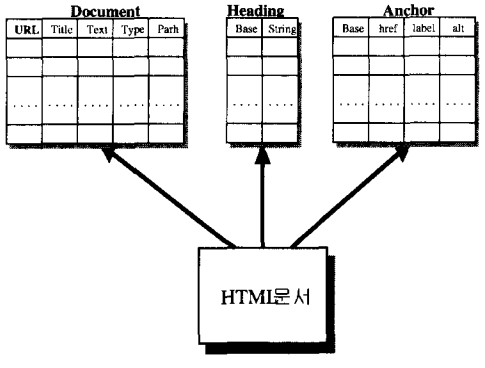


그림 1. HTML 문서 데이터 모델

그림 1.은 WebDBs에 저장된 모든 HTML 문서에 대한 전체 스키마 구성이다. 실제적으로 웹에서의 HTML 문서는 각 문서들이 링크로 연결된 그래프 구조이다. 따라서 이들간의 상호 연결은 document 테이블에 있는 url 값과 anchor, heading 테이블에 저장된 base 애트리뷰트의 url 값에 의하여 각 항목들 사이의 연결 구조로 추적할 수 있다. 위와 같은 표현은 HTML 문서의 그래프 연결 특성을 관계 데이타베이스의 테이블 형식의 표현으로 변환한 것으로 사용자 질의에 대한 그래프 추적 부분을 생략할 수 있는 장점이 있다.

## 2. WebDBs 질의어

웹 데이타베이스 질의어는 일반적인 데이타베이스 언어와는 다르게 적용되는 범위가 HTTP 프로토콜을 이용하여 접속이 가능한 문서에서 사용자가 찾고자 하는 정보를 검색하는 범위로 한정된다. 본 연구에서는 웹 데이타베이스 언어를 현재 보편적으로 사용되고있는 SQL 표준 질의어를 확장하여 응용하고자 한다. 따라서 아래와 같은 제한점과 요구사항이 추가된다.

### 2.1 질의어 설계

WebDBs 질의어는 표준 SQL 질의어와 마찬가지로 데이타베이스에 저장된 특정 정보를 검색한다. 이때 WebDBs 질의어의 경우 자료들이 추출된 도메인이 웹에 등록된 한정적 자료이고 별도의 DBA가 없이 비교적 단순한 문장들의 집합 형태로 이루어진 문서에 대한 검색을 수행하는 것이 일반 SQL 질의어와는 다른 형태를 가진다. 그리고 일반적인 데이타베이스의 정보와는 달리 하이퍼텍스트의 특성을 가지고 있기 때문에 검색 대상 자료들은 각 l링크들의 추적을 통하여 획득된다. 본 연구에서는 웹 데이타베이스에서 응용될 수 있는 질의 형태를 아래의 3가지로 구분하였다.

- 웹에 등록된 문서에 포함된 문자열을 찾는 질의
  (예: "소제목에 논문이란 단어가 나오는 문서를 찾아라")
- 숫자, 날짜, 크기 등과 같은 정보에 대한 비교 연산자를 포함하는 질의
  (예: "1998년 이후에 등록된 2K byte 이상의 문서를 찾아라")
- 문자열 검색과 숫자 등의 비교 연산을 동시에 사용하는 질의
  (예: "소제목에 논문이라는 단어를 포함하면서 1998년 이후에 등록된 문서를 찾아라")

위에서 분류한 질의를 원활하게 표현 할 수 있고,

사용자들에게 비교적 친숙한 SQL 질의 유형과 유사하게 하기 위하여 WebDBs 질의어를 아래와 같은 기본적인 구조로 정의하였다.

```
Query := SELECT    AttributeList
         FROM   TableList  [,BASE =  url ]
         [WHERE Condition]
        [ OptionsList ];
```

그림 2. WebDBs 질의어 문법의 기본 구조

그림 2.에서 제안한 질의어는 SQL유형의 SELECT-FROM-WHERE 형태를 취하며 아래의 5가지 특성을 가진다.

1) 모든 질의 결과는 릴레이션이다.
2) SELECT-FROM-WHERE 구조로 이루어진다.
3) WHERE절 이하는 애트리뷰트들의 서술로 이루어진다.
4) BASE는 검색 추적의 첫 주소를 명시하면 별도의 base url을 사용자가 제공하지 않는 경우에는 시스템 내부의 값을 base로 한다. 즉 어떤 경우에도 base url은 존재한다.
5) Option은 기타 검색의 부가 조건들을 기술하는데 사용된다.

위의 질의어에서 표준 SQL 언어와의 차이점은 FROM 절에 나타나는 BASE 부분과 질의어 후반에 첨가되는 option 부분이다. BASE부분은 사용자의 질의 수행시 검색을 하는 시작점을 의미하며, 생략된 경우에는 미리 정의된 검색 사이트를 시작점으로 선정하여 검색을 수행한다. 질의어 뒷부분에 기술되는 option은 해당 질의의 부가적인 특성을 나타내며, 검색 결과의 갱신 주기나, 연결 문서 추적의 정도 등을 나타낼 때 사용된다.

HTML 문서의 링크 연결은 다음 문서에 대한 분기를 의미하며, 해당 문서가 네트워크를 통하여 연결되어 있기 때문에 각 문서의 저장 위치가 각 문서에 대한 상호 연관성에 깊이 관여한다. 따라서, 각 링크의 연결 문서와 현재의 문서 사이의 지역적인 차이를 기준으로 하여, HTML 문서에 나타나는 각 링크를 다음의 세가지 유형으로 구분 할 수 있다.

- Inner : 해당 사이트 내에 존재하는 경우
- Local : 지역 도메인 내에 존재하는 경우
- Global : 다른 도메인 내에 존재하는 경우

이와 같은 구분은 인터넷 로봇이 정보를 추적할 때 사용자의 질의어를 분석하여 알맞은 수준으로의 문서 추적을 수행하기 위하여 사용되며 인터넷 로봇 추적 깊이는 사용자 질의에 명시된 질의수행 옵션에 따라서 정해진다.

### 2.2. WebDBs 질의어의 제한 사항

이상과 같이 설계된 WebDBs 질의는 웹 환경 아래에서 HTML 문서에 수록된 내용에 대한 검색을 실행하기 때문에, 질의 처리가 일반적인 데이타베이스와는 다른 특성을 가진다. 따라서 아래와 같은 제한 조건을 가진다.

1) HTML 1.0 또는 HTML 1.1 표준을 따르는 문서에 대한 검색만을 허용한다.
2) 이미지, 음성, 동영상 등 멀티미디어 정보에 대한 검색은 제외된다.
3) 웹에 관련된 업체의 표준이 아닌 표현 부분은 문장 전체를 텍스트로 해석하여 검색을 수행한다.
4) HTML 문서를 릴레이션으로 표현할 때 특정 애트리뷰트가 해당 값이 문서에 포함되지 않은 경우가 있다 이때 미리 정해진 기본 값을 배정한다.
5) 검색 결과에는 아래와 같은 사항을 반드시 포함하며, 사용자가 질의에 명시하지 않은 경우에도 결과에 포함한다.
   . 문서의 url 값
   . 해당 문서의 전체 원문 내용

1)항의 경우 새로운 표준이 발표될 때 이에 따르는 갱신을 수행한다. 2)항의 경우에는 현재 연구가 활발하게 진행중이나, 응용할 수 있는 단계가 아니므로 본 연구에서는 제외하였다, 3)항의 경우에는 업체마다 별도의 표준이 지속적으로 갱신되고 있기 때문에 이를 모두 수용할 수 없기 때문에 본 연구의 영역에서는 고려하지 않는다. 4)항에서 기본 값은 숫자와 같은 정보는 '0', 문자열의 경우엔 null string과 같은 기본 값을 배정한다. 5)항에서 문서에 대한 url값은 3.1절에서 해당 튜플의 키로 사용되므로 꼭 필요한 애트리뷰트이다. 문서 전체의 내용은 사용자에게 검색 결과를 참고로 제시할 때 사용된다.

## 3. 질의 예제

앞 절에서 예시한 바와 같이 모든 HTML 문서를 그림 1.과 같은 테이블로 구성된 데이타베이스로 변환되었을 때 각각의 응용에 대한 질의 예는 아래와

같다.

[예제 1]
"전자통신 연구소 사이트에 연결된 문서 중에 'etri'이라는 단어를 포함한 문서를 찾아라"

```
SELECT d.url, d.title, d.text, d.type
FROM Document d, Anchor a,
        BASE = "http://www.etri.re.kr/"
WHERE d.url = a.href AND d.text CONTAINS "etri"
```

여기서 BASE는 문서 검색의 기준이 되는 url을 나타내며 전자통신 연구소 사이트의 첫 주소는 http://www.etri.re.kr/ 이다.

[예제 2]
"전자통신 연구소 사이트에 연결된 문서 중에 'etri'이라는 단어 포함하고, 문서 중에 '프로토콜'이라는 문자를 포함하는 문서를 찾아라"

```
SELECT d.url
FROM Document d, Anchor a
WHERE d.text CONTAINS "프로토콜" AND " d.url
        IN
        (SELECT d.url, d.title, d.text, d.type
        FROM Document d, Anchor a,
                BASE = "http://www.etri.re.kr/"
        WHERE d.url = a.href AND d.text CON-
                TAINS "etri") ;
```

위의 예는 좀더 간단하게 표현이 가능하나 뷰를 이용하는 응용을 소개하기 위하여 질의어를 구성하였다.

[예제 3]
"뷰를 이용하여 예제 2를 실행하라."

```
CREATE VIEW etri AS
    SELECT d.url, d.title, d.text, d.type
    FROM Document d, Anchor a,
            BASE = "http://www.etri.re.kr/"
    WHERE d.url = a.href AND d.text
            CONTAINS "etri";
```

위와 같이 'etri'이라는 단어를 포함한 문서를 모두 찾은 뷰 'etri'을 먼저 선언한다.

```
SELECT d.url
FROM VIEW etri
WHERE etri.text CONTAINS "프로토콜" AND
        d.text CONTAINS "etri";
```

여기서 기존에 선언된 internet이라는 뷰를 이용하는 경우에는 추가적인 검색 조건을 명시하여 검색을 수행할 수 있다. 따라서 미리 선언된 뷰의 내용을 저장해 둔다면 뷰를 이용하는 질의에 대해서는 좀더 간편하게 질의를 처리할 수 있다.

## Ⅳ. WebDBs 시스템

WebDBs는 웹 전체를 데이타베이스로 간주하고 이에 대한 표준 질의를 수행 할 수 있는 인터넷 검색 엔진이다. 또한 상용되는 검색 엔진이 서버 중심의 인덱스 서버인 것에 비하여 WebDBs는 사용자측이 웹 인덱스를 수집하고 저장하는 사용자 중심의 웹 검색 엔진이다. 이때 검색 대상이 될 문서는 인터넷 로봇 생성 알고리즘에 의하여 결정되고 인터넷에 파견된 각 인터넷 로봇은 검색 조건에 맞는 HTML 문서를 질의 처리기에 전달한다. 인터넷 로봇을 통하여 획득된 HTML은 분석 단계를 거쳐서 튜플 단위로 지역 데이타베이스에 저장된다. 질의 처리는 현재 지역 데이타베이스에 저장된 자료를 중심으로 하여 검색 수행이 이루어지고, 새로운 정보를 인터넷 로봇을 통하여 전달받는다. 또한, 인터넷의 자료들은 수시로 그 내용이 변화한다는 점을 감안하여, 한번 저장된 정보들도 사용자의 조건 명세에 따라 정기적으로 자동으로 갱신한다. 여기서 지역 데이타베이스에는 사용자 질의 처리 결과와 뷰의 인스턴스 내용들이 저장되며, 이들은 4.1절에서 제시하는 구조를 가지는 캐쉬로 관리된다. WebDBs의 전체적인 기본 구성은 그림 3.과 같다.

### 1. 지역 데이타베이스 구조

WebDBs 시스템의 질의 수행 속도를 결정하는 것은 질의 처리에 필요한 데이터 인스턴스들을 획득하는 것이다. WebDBs는 뷰와 이전에 수행된 질의 결과를 캐쉬에 저장하여 이를 다음 질의 수행에 이용함으로써 통신 회선을 통한 인스턴스 수집에 따르는 질의 수행 시간을 줄인다 따라서 효율적인 지역 데이타베이스 관리 기법이 요구된다.

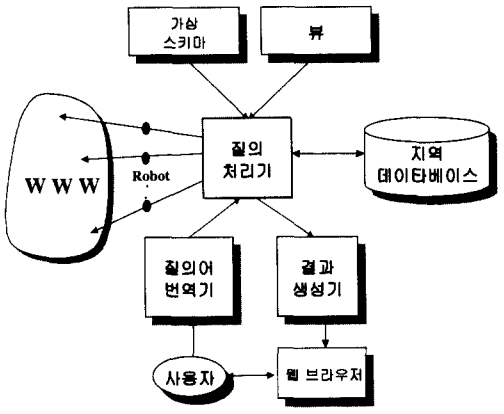


그림 3. WebDBs 시스템 구성

그림 1.의 테이블은 각 질의와 뷰 선언마다 별개로 생성된다. 따라서 문서 테이블은 질의마다 별개의 테이블로 구성되기 때문에 중복된 문서들이 발생될 수 있다. 본 연구에서 이와 같은 문제를 해결하기 위하여 전체의 문서 테이블을 통합한 통합 문서 테이블(Total Document Table : TDT)을 설계하였다. 통합 문서 테이블은 문서 화일에 대한 정보뿐만 아니라 캐쉬 관리에 필요한 정보를 함께 저장함으로써 문서 관리의 효율성을 증대하였다. 통합 문서 파일의 각 항목은 표 1.과 같이 정의된다.

표 1. 통합 문서 테이블(TDT)

| | field 명 | 내 용 |
|---|---|---|
| 문서 엔트리 | no | 문서의 순서 번호 |
| | url | 해당 문서 url |
| | title | 문서 제목 |
| | HTML_text | HTML 문서 화일명 |
| | type | 문서 형식 |
| | path | 해당 문서까지의 추적경로 |
| | base_url | 부모 문서의 url |
| 캐쉬 엔트리 | count | 해당 문서를 조회한 횟수 |
| | expire_date | 문서 만료시기 |
| | last_access | 문서의 최종 조회 일자 |
| | auto_access | 주기적 자동 갱신 여부 |
| | access_period | 자동 확인 주기 |
| | flag | 문서 삭제 여부 |

표 1.에서 문서 엔트리는 그림 1.의 문서 모델에 있는 내용과 동일하고 캐쉬 엔트리는 해당 문서의 캐쉬 재배치 때 이용되는 정보들이다. 표 1.의 각 튜플들은 검색 로봇이 추출해온 모든 문서마다 1:1로 생성된다. 질의어 테이블과 뷰 테이블에는 실제 문서 내용을 저장하는 것이 아니라 통합 문서 테이블에 저장된 대응하는 문서에 대한 포인터 값만을 가진다. 따라서 여러 개의 질의어나 뷰가 선언되어 있어도, 통합 문서 테이블에는 동일한 문서에 대해서는 1개의 엔트리만이 존재한다. 모든 질의와 뷰 선언은 대응되는 질의 테이블과 뷰 테이블에 등록되며 이에 대한 인스턴스 정보들은 각 질의 마다 별도의 테이블들로 개별적으로 등록된다.

## 2. WebDBs 로봇 추적 알고리즘

일반적인 웹 데이타베이스 시스템의 질의 수행 속도를 좌우하는 것은 검색에 필요한 인스턴스 정보를 얼마나 빨리 획득하느냐에 있다. WebDBs는 4.1절에서 제안한 통합 문서 테이블을 이용하여, 기존에 검색이 완료되어 지역 데이타베이스에 저장된 문서는 별도로 통신 회선을 통하여 획득하지 않고 이미 검색된 자료로부터 질의 수행에 필요한 인스턴스들을 최대한으로 획득하고자 한다. 따라서 4.1절에서 제시한 각종 테이블들의 관리와 로봇 추적 알고리즘은 서로 연관되어 구성되어야 한다.

웹은 전체적으로 HTML 문서들의 상호 연결로 구성된 그래프 구조이지만, 로봇에 의하여 검색을 시행하는 경우 기존에 방문했던 문서에 대해서는 중복된 방문을 하지 않는다. 따라서 로봇 방문 결과는 그림 4.와 같은 싱크 트리(sink tree) 형태를 취하게 된다. 그림 4.의 (a)는 원래의 HTML 문서의 그래프 구조이며, (b)는 모든 방문 가능한 문서들을 방문한 경우의 싱크 트리이며, (c)는 현 검색이 아닌 다른 질의 수행이나 뷰 선언에서 노드 E가 이전에 검색되어 지역 데이타베이스에 존재할 경우 해당 문서에서 파생되는 자식 노드들에 대해서는 기존에 지역 데이타베이스에 저장된 내용을 대치할 수 있다는 것을 나타낸다. 따라서, WebDBs 에서는 지역 데이타베이스에 저장된 문서가 존재할 경우 해당 문서에 대한 통신 회선을 통한 검색은 물론이고 해당 문서와 연결된 다른 문서들에 대한 검색을 수행하지 않더라도 필요한 인스턴스들을 모두 구할 수 있다는 장점이 있다. 이를 알고리즘으로 표현하면 [알고리즘 1]과 같이 작성된다. [알고리즘 1]에서 방문 예정인 url 에 대응되는 문서가 통합 문서 테이블에 존재하는 경우에 해당 문서와 연결된 모든 문서의 테이블 인스턴스 값들을 통신을 통하지 않고 획득하기 때문에 자료 획득 시간을 단축시킬 수 있다.

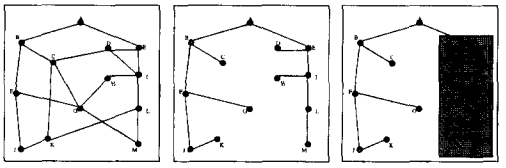

그림 4. 로봇 검색의 경우 방문되는 노드들의 관계

[알고리즘 1] 로봇 추적 알고리즘
입력 : 질의 Q
출력 : TDT 와 각 테이블
PROCEDURE
READ Q
url_queue = base_url
WHILE url_queue NOT NULL AND (검색 범위 만족)
url = GET(url_queue)
IF (url IN tdt) THEN
TDT(url).flag = true;
Q.flag = true;
local_queue = (TDT를 추적하여 검색 범위 이내의 모든 문서의 url)
WHILE local_queue NOT NULL {
(해당 url의 지역 데이타베이스 저장 내용을 copy)
TDT(url).flag = true; }
}
ELSE THEN
GET_HTML(url)
MAKE_TABLE(document, anchor, heading)
(TDT에 새로운 문서 항목 등록)
url_queue = (모든 anchor)
ENDIF
}
ENDPROCEDURE

### 3. 지역 데이타베이스 관리

WebDBs에서는 로봇이 웹을 추적하여 추출된 각 문서들을 그림 4.와 같은 싱크 트리 구조로 검색하고 이를 지역 데이타베이스에 저장한다. 지역 데이타베이스는 캐쉬 형태로 관리된다. 따라서 캐쉬 재배치가 요구되어 한 노드가 제거 대상으로 설정된 경우에 해당 노드의 자손 노드들도 함께 제거되어야 한다. 또한 이들 인스턴스를 이용하는 질의나 뷰는 질의 수행에 필요한 인스턴스들을 충분하게 확보하지 못하였으므로 차후에 해당 질의 결과를 이용하고자 할 때 재검색 하여야 한다. [알고리즘 2]는 이를 고려하여 지역 데이타베이스에 저장된 노드를 제거하는 알고리즘이다. [알고리즘 2]에서 Q는 표 1의 질의 테이블에 저장된 질의이며, 뷰에 대한 알고리즘은 자료구조가 같고 알고리즘이 같기 때문에 생략하였다.

[알고리즘 2] 노드 제거 알고리즘
입력 : 제거 대상 노드 (url)
출력 : 캐쉬 여유 공간과 와 테이블 갱신
PROCEDURE
leaf_url = (TDT에서 url로 파생되는 문서중 최종 자손)
IF tdt(url).flag=true THEN {
FOR (모든 Q.flag=true인 질의 Q에 대하여)
IF url IN Q.document THEN Q.flag=flase ;
ELSE THEN {
before_url = url;
FOR url_i = url TO leaf_url {
IF TDT(url_i).flag = true
THEN {leaf_url = before_url; EXIT} ;
before_url = url_i ;
}
(해당 url을 인스턴스로 하는 질의 Q를 찾는다) ;
Q.flag=false ;
}
TDT_UPDATE(url, leaf_url);
ENDPROCEDURE

위의 알고리즘에서 Q.flag가 참인 경우에는 해당 질의를 처리하는데 필요한 인스턴스들이 지역 데이타베이스에 모두 존재하는 것이고, 거짓인 경우에는 일부 인스턴스들이 제거되었음을 나타낸다. TDT_UPDATE 함수는 제거 대상 노드로부터 모든 자손 노드까지의 인스턴스를 실제로 지역 데이타베이스에서 삭제하고 통합 데이터 테이블을 갱신한다. 위의 알고리즘은 다소 복잡한 경향이 있으나, 자손 노드까지 동시에 제거하기 때문에 다량의 여유 공간을 확보할 수 있고, 따라서 노드 제거는 빈번하게 발생하지 않는다.

## V. WebDBs 시스템 구현 및 실험

WebDBs는 검색 엔진이기 때문에 현재 사용자들이 가장 많이 사용하고 있는 운영체제 환경인 Windows 95/98 환경에서 원활하게 운영 될 수 있도록 하였다. 이를 위하여 Visual Studio 개발 환경에서 시스템을

구현하였다.

### 5.1 WebDBs 시스템 구성

그림 5.는 WebDBs 시스템의 각 모듈별 연관성을 나타낸다. 사용자의 질의는 시스템 사용자의 데이타베이스에 대한 이해 정도가 서로 틀린 점을 감안하여 일반 질의어 입력기와 전문가 질의어 입력기를 별도로 설계하여 사용자 입력을 받아들인다. 일반 질의어 입력기의 경우 SQL 질의어에 대하여 초보적인 지식을 가지고 있다고 해도 쉽게 사용이 가능하도록 설계하였으며, 전문가의 질의 입력기는 일반적인 SQL 질의어 형식으로 입력이 가능하도록 하였다. 검색 결과를 사용자에게 보여주는 웹 브라우저로는, 현재 상용되고 있는 웹 브라우저 중 Internet Explorer 4.0 이상의 경우 WebDBs에 내장된 전용 웹 브라우저와 함께 사용이 가능하다. 질의 처리기는 그림 5.에서 볼 수 있듯이 시스템 전반적인 운영을 관장하는 질의 처리기 부분과 캐쉬 관리기, 로봇 생성기, 뷰 관리기, HTML 문서 분석기, 결과 생성기로 구성되었다.

사용자의 질의가 일정 주기에 자동으로 문서 내용의 갱신을 명시한 경우, 시스템이 이를 실행하는 동안 정기적으로 기존 자료들에 대한 갱신을 수행한다. 지역 데이타베이스로는 질의 수행시 필요한 자료를 일시적으로 저장하는 임시저장소와 TDT, 뷰 테이블, 질의어 테이블로 구성된다. 이들 자료들은 4장에서 제시한 알고리즘을 기반으로 생성되고 운영된다.

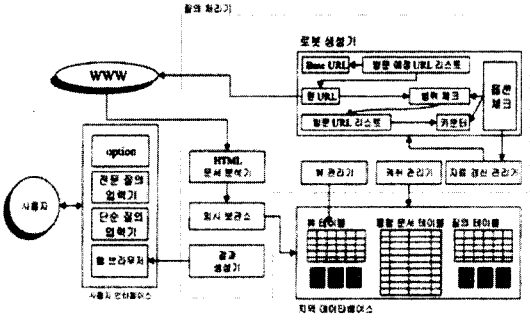


그림 5. WebDBs의 시스템 구성도

### 2. 질의 처리 실험

그림 6.은 [예제 1]의 질의를 단순 질의 입력 화면에서 입력한 경우이다. 단순 질의 입력 화면은 SQL 언어에 대한 사용자의 수준이 낮더라도 비교적 수월하게 필요한 검색어를 생성할 수 있음을 보여준다.

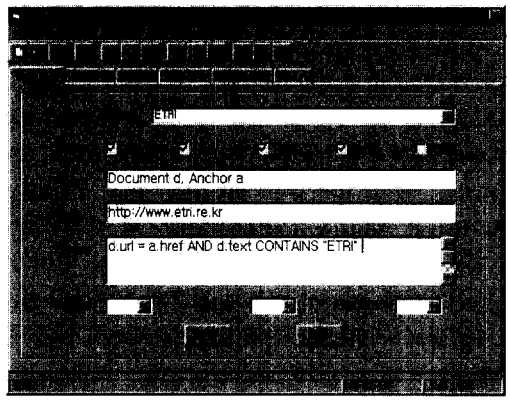


그림 6. WebDBs 일반 질의 입력 화면

질의 수행 결과는 그림 7.과 같이 전용 웹 브라우저를 통하여 볼 수 있다. 이때 사용자가 특정 문서에 대하여 좀더 자세한 내용을 보고싶은 경우에는, 해당 시스템의 기본 웹 브라우저를 이용하여 해당 문서를 관찰할 수 있게 하였다.

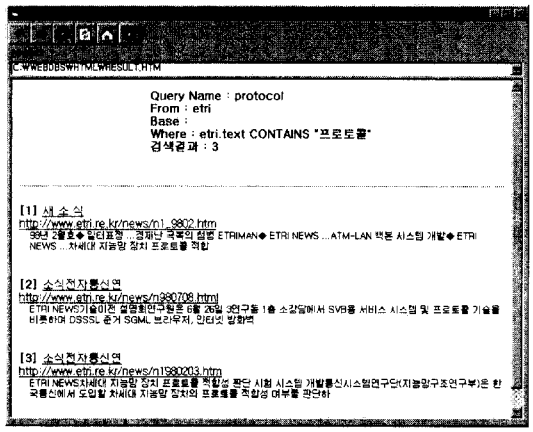


그림 7. 예제 3 질의의 수행 결과 화면

## VI. 평가 분석

본 연구에서 설계 구현한 웹 데이타베이스 시스템을 이용한 검색이 일반적인 인덱스 서버를 통한 검색과 차별화 될 수 있는 가장 큰 부분은 인덱스 서버 형식의 검색 엔진의 경우 해당 서버에 검색하고자 하는 URL 정보가 등록되어 있어야 한다. 그러나 웹 데이타베이스의 경우에는 이런 정보가 저장되어 있지 않더라도 검색이 가능하다는 점이다. 웹 데이타베이스 검색 엔진이 가지는 가장 큰 결점은 사용자가 질의를 제공한 후 그 결과를 획득하는데 소요되

는 질의 수행시간이 서버 중심의 검색 엔진에 비하여 상대적으로 느리다는 것이다. W3Qs와 WebSQL에서는 이에 대한 뚜렷한 해결책을 제시하지 않았다.

본 연구에서는 기존의 검색 정보를 질의를 중심으로 하여 캐쉬에 저장하는 방법을 제안하였다. 기존의 네트워크 캐쉥 방법들인 FPC(Fast Page Caching), HAC(Hybrid Adaptive Caching)[8]등은 웹 데이타베이스 질의와의 연관성을 고려하지 않은 캐쉬 관리 기법을 사용한다. 따라서 기존에 검색된 결과 인스턴스 중 일부분이 지역 데이타베이스에 존재하지 않을 수 있다.

일반적인 네트워크 캐쉬 관리 방법과 제안된 캐쉬 관리 방법을 WebDBs의 지역 데이타베이스 관리 방법으로 적용하였을 때 차이점은 다음과 같다. 웹 데이타베이스에서는 질의 처리를 수행하기 위하여 요구되는 자료들을 연속적으로 추적하여 필요한 튜플들을 자동으로 추출한다. 따라서 유한 자원인 캐쉬의 재배치가 질의의 성격에 따라 연속적으로 발생한다. 캐쉬 체제의 조회 시간 성능은 아래와 같이 표현 할 수 있다.

$$A = H + M \times P$$

여기서, A는 전체적인 조회 시간이고, H는 캐쉬에서 성공적으로 자료를 획득하였을 때의 시간, M은 캐쉬 실패율, P는 실패할 때의 부과시간이며 P는 아래와 같이 표현할 수 있다.

$$P = F + R + C$$

F는 자료를 획득하는데 소요되는 시간, R은 기존 자료와의 재배치 시간, C는 획득된 자료를 캐쉬 자료구조에 맞게 변환하는 시간이다. FPC나 HAC와 같은 네트워크 캐쉬 방법과 본 연구에서 제안한 캐쉬 방법은 획득된 정보의 재배치에서 서로 다른 특성을 가진다. 제안된 방법에서는 재배치 할 때 자식 노드들을 동시에 제거하기 때문에 여유공간을 기존의 방법에 비하여 좀더 여유 공간을 확보할 수 있다는 장점이 있다. 즉 캐쉬에 저장되지 않은 정보를 재배치 할 때 재배치에 소요되는 시간을 줄일 수 있다. 예를 들어 질의 수행을 위하여 문서를 획득하는데 n회의 캐쉬 실패가 발생할 경우 FPC나 HAC는 아래와 같이 실패 부과시간을 산출 할 수 있다.

$$P = n (F + R + C)$$

여기서 제안된 캐쉬 방법에서는 한번의 재배치에 여러개의 인스턴스들을 동시에 제거하므로 아래와 같은 수식으로 표현된다.

$$P = n (F + C) + R'$$

따라서 질의 지역성이 있는 웹 데이타베이스 캐쉬 환경에서는 제안한 방법이 자료의 재배치 측면에서 좀더 효율적이다. 위에서, 제안된 방법에서는 동시에 여러개의 인스턴스들을 캐쉬에서 제거하기 때문에 캐쉬의 성공률 H가 변화될 수 있으나, 캐쉬공간의 크기가 일정크기 이상일 경우에는 전체적인 성공 확률의 편차가 현저하게 적기 때문에 이에 대한 고려는 무시하였다.

## Ⅶ. 결 론

본 연구에서는 현재 가장 많이 사용되는 인터넷 서비스인 웹에서 사용자가 요구하는 정보를 SQL과 유사한 질의어를 이용하여 검색하는 방안을 제안하였다. 이를 위하여 웹에 등록된 HTML 문서의 주요 특성을 분석하여, 이들로 부터 유효한 검색 정보를 테이블 형태로 저장할 수 있는 튜플들을 정의하였다. 즉, 모든 HTML 문서는 document, anchor, heading의 세가지의 테이블에 저장되는 정보로 볼 수 있다. 따라서 일반적인 데이타베이스 시스템에서 보편적으로 사용되는 SQL과 유사한 질의어를 통하여 질의 수행이 가능하다. 본 연구에서는 위에서 제안한 데이타베이스 테이블을 충분하게 활용할 수 있는 웹 검색 전용 질의어를 설계하였다. 제안된 질의어는, 기존의 웹 데이타베이스 검색 엔진이 갖는 하이퍼텍스트 데이터 추적과 유사한 방식의 항해형 질의 형식을 배제함으로써 좀더 SQL 표준 질의어와 유사한 형태를 취한다.

기존에 연구된 웹 데이타베이스 시스템들은 질의가 수행되면서 필요한 인스턴스들을 네트워크를 통하여 수집하기 때문에, 서버 중심 검색 엔진에 비하여 상대적으로 질의 수행 시간이 현저하게 오래 걸린다는 문제점이 있다. 본 연구에서는 이러한 문제점을 보완하기 위하여 사용자가 자주 검색하는 응용들을 미리 뷰로 선언하는 방안을 제시하였다. 그리고 선언된 뷰는 연산자와 피연산자를 지역 데이타베이스에 저장하도록 하였다. 따라서 뷰를 이용하는 질의는 지역 데이타베이스로부터 자료를 검색하기 때문에 질의 수행 속도가 향상된다. 또한 자료에 대한 인스턴스들을 지역 데이타베이스에 저장하면서 문제가

되는 지역 데이타베이스 관리를 웹 지역성을 고려한 캐쉬 관리 기법을 도입하였다. 그리고 위와 같은 개념의 시스템인 WebDBs를 Visual Studio 환경에서 구현하였다.

차후의 연구 과제로는 제안된 시스템의 구현과 좀 더 빠른 질의 수행을 위한 최적화 과정, HTML 문서에 포함된 의미적인 분석이 추가로 연구되어야 한다.

## 참 고 문 헌


[1] Michael F. Schwartz, "Internet Resource Discovery at the Web", *University of Colorado, IEEE Computer,* pp. 25 ~ 35, 1993.

[2] David Konopnicki, Oded Shmueli, "W3QS: A Query System for the World Wide Web", *International Conf. on Very Large Data Bases(VLDB'95) Zurich,* pp. 54-65, 1995.

[3] Alberto O. Mendelzon, George A. Mihaila, and Tova Milo, "Querying the World Wide Web", *International Journal of Digital Library,* pp. 80-91, 1997

[4] Luis Gravano, "Merging Ranks from Heterogeneous Internet Source", *International Conf. of Very Large Data Bases(VLDB'97) Greece*, pp. 196-205, 1997.

[5] Paolo Atzeni, "To Weave the Web", *International Conf. of Very Large Data Bases(VLDB'97) Greece*, pp. 206-215, 1997.

[6] Altavista Discovery Home Page.http://discovery.altavista.com

[7] "HTTP-ng Architectual Model", *W3C Working Draft,* 1998. http : //www.w3c.org / TR / WD - HTTP - NG architecture-1998710

[8] Miguel, Castrp, "HAC: Hybird Adaptive Caching for Distributed Storage", http://www.pmg.lcs.mit.edu/papers/hac-97/published.html



김 홍 일(Hong-il Kim) 정회원

1986년 2월 : 홍익대학교 전자계산학과 졸업
1989년 2월 : 인하대학교 전자계산학과 석사
1991년 9월~현재 : 홍익대학교 전자계산학과 박사과정
1994년 3월~현재 : 대진대학교 컴퓨터공학과 전임강사

임 해 철(Hae-chull Lim) 정회원

1976년 : 서울대학교 계산통계학과 졸업
1978년 : 한국과학기술원 전산학과 석사
1988년 : 서울대학교 컴퓨터공학과 박사
1989년~1990년 : University of Florida(Post doc.)
1981년~현재 : 홍익대학교 컴퓨터공학과 교수