데이터베이스에서 한글 글자 단위 검색 기능 설계 및 구현

이호진* · 이중화*

*동의대학교 컴퓨터소프트웨어공학과

The design and implementation of the search function of hangeul characters in database.

Ho-jin Lee* · Jung-hwa Lee*

*Dongeui University, Dept. of Computer Software Engineering

E-mail : hjlee83@deu.ac.kr

요 약

본 논문에서는 한글의 글자 단위 검색 기능 구현에 있어서 한글의 빈도수 정보를 이용하여 보다 효율적으로 검색할 수 있는 방안을 제시한다. 또한 본 논문에서 제시하는 알고리즘을 적용하여 데이터베이스에서 한글 글자 단위 검색이 가능하도록 검색 기능을 구현하였다.

ABSTRACT

In this paper, we can find a new effective way in the search function of hangeul characters by using frequency of data. Also using a new algorithm suggested in this paper, we made the effective search function of hangeul characters in database.

키워드

한글의 글자 단위, 한글의 빈도수 정보, 검색

Ⅰ. 서 론

한글 검색 기능은 워드 프로세서, 데이터베이스 등의 많은 소프트웨어 등에서 필수적으로 사용하는 중요 기능 중 하나이다.

한글의 신속한 단어 검색을 위해서 해싱 기법, 단어 사용 빈도에 따른 계층적 구조, 리스트 구조, 트라이 구조 등과 같은 다양한 방법들이 제안되었다. [1][2][3]

그러나 모아쓰기를 하는 한글의 문자적 특성을 고려해 본다면 검색 기능에서도 글자를 기본 단위로 처리 하는 것이 필요하다.

따라서 본 논문에서는 효율적인 글자 단위 검색을 위해 글자단위 빈도수 정보를 사용하여 신속한 단어 검색이 이루어 질 수 있도록 하였으며, 본 논문에서 제안하는 빈도수 정보를 이용한 글자단위 검색 기능을 데이터베이스에서 사용할 수 있도록 기능을 구현하였다.

Ⅱ. 본 론

한글에서의 글자단위 검색이란 첫소리, 가운뎃소리, 끝소리와 같은 글자단위를 사용하여 완전한 글자마디를 찾는 방법이다.

그리고 글자단위의 빈도수란 글자마디 각각의 첫소리, 가운뎃소리, 끝소리가 사전 전체에서 사용되어진 횟수를 말한다. 이러한 글자단위의 빈도수를 검색에 활용하여 검색속도가 향상되도록 구현하였다.

2.1 글자단위 검색유형

한글에서의 글자마디는 첫소리글자-가운뎃소리글자와 첫소리글자-가운뎃소리글자-끝소리글자로 구성된 글자마디를 완전한 글자마디라고 한다.

글자를 기본단위로 하는 한글 검색에서 나타날 수 있

는 검색유형은 완전한 글자마디들로 구성된 문자열을 찾고자 하는 경우와 첫소리글자-가운뎃소리글자-끝소리글자 중 어느 특정 부분만을 포함하는 검색으로 분류된다.[1]

본 연구에서는 데이터베이스에서 한글 사전을 검색할 경우에 완전한 글자마디가 한글 사전에 존재하는 경우만을 고려하여 검색 유형을 만들어 보았다. 또한 찾고자 하는 글자마디를 효율적으로 검색할 수 있도록 글자마디수를 제한할 수 있는 기능을 구현하였다.

다음은 글자단위 검색 유형을 정리해 본 것이다. [1]

① 첫소리-가운뎃소리-끝소리가 존재하고 첫소리만 입력하는 경우

/ㄱ-*-*/ -- 가, 각, 간, 갈 등

② 첫소리-가운뎃소리-끝소리가 존재하고 끝소리만 입력하는 경우

/*-*-ㄴ/ -- 간, 견, 건, 관 등

③ 첫소리-가운뎃소리-끝소리가 존재하고 첫소리와 끝소리만 입력하는 경우

/ㄱ-*-ㅇ/ - 강, 공, 갱 등

④ 첫소리-가운뎃소리-끝소리가 존재하고 가운뎃소리만 입력하는 경우

/*-ㅏ-*/ -- 가, 각, 간, 갈 등

⑤ 첫소리-가운뎃소리가 존재하고 첫소리만 입력하는 경우

/ㄹ-*/ -- 라, 래, 랴, 러 등

⑥ 첫소리-가운뎃소리가 존재하고 가운뎃소리만 입력하는 경우

/*-ㅑ/ -- 냐, 랴, 야

⑦ 첫소리-가운뎃소리-끝소리가 존재하고 첫소리와 가운뎃소리를 입력하는 경우

/가-*/ -- 가, 각, 간, 갈 등

⑧ 첫소리-가운뎃소리-끝소리가 존재하고 가운뎃소리와 끝소리를 입력하는 경우

/*-ㅏㄴ/ -- 간, 깐, 난, 단 등

⑨ 완성된 글자마디+글자단위(①~⑧) 검색

/고구*-ㅏ/ -- 고구마
(완성된 글자마디 + 유형⑥)

/*-ㅏㄱ기/ -- 각기, 악기, 학기
(완성된 글자마디 + 유형⑧)

⑩ 글자마디 수 제한 기능

입력되는 글자마디 또는 글자단위의 마지막에 숫자를 입력하여 찾고자하는 글자마디의 개수를 제한하였다. 숫자를 입력하지 않는 경우에는 포함하는 모든 글자를 찾게 된다.

/*-사-*2/ -- 화생, 화석, 화서 등
(2개의 글자마디만 검색)

/*-사-*/ -- 확, 확견, 확실히, 환골탈태 등
(글자마디의 수에 관계없이 모두 검색)

2-2 글자단위 검색을 위한 빈도수 정보

한글의 글자단위 빈도수 정보를 이용하기 위해서는 글자마디를 분리할 수 있는 KS X 1001 조합형, 유니코드 조합형, 유니코드 완성형으로 이루어진 한글사전을 사용하여야 한다.

본 연구에서는 KS X 1001 완성형을 글자마디로 분리할 수 있는 유니코드 완성형의 한글사전으로 변환하였다.

유니코드 완성형은 한글의 11,172 글자마디를 모두 포함하고 있으며, 그림 1와 같은 계산식에 의해 첫소리, 가운뎃소리, 끝소리 글자의 가나다순 인덱스를 찾을 수 있다.[4]

```
index = 부호값 - 0xAC00;

끝소리글자 인덱스 = index % 끝소리글자수

가운뎃소리글자인덱스=((index - 끝소리글자인덱스) / 끝소리글자수) % 가운뎃소리글자수

첫소리글자인덱스=((index - 끝소리글자인덱스) / 끝소리글자수) / 가운뎃소리글자수
```

그림 1. 유니코드 완성형의 경우 글자 검색

유니코드 완성형으로 이루어진 한글사전에서 첫소리, 가운뎃소리, 끝소리 글자를 그림 2와 같은 계산식에 의해 글자단위별 빈도수를 찾을 수 있다.

```
for(i=0;;i+=2)
{
 if(str[i] == 단어수){
 break;
 }
 for(k=0;k<첫소리글자수; k++){
 if(k == 첫소리)sorimadi[i/2].first[k]++;
 }
 for(k=0;k<가운뎃소리글자수;k++){
 if(k == 가운뎃소리)sorimadi[i/2].second[k]++;
 }
 for(k=0;k<끝소리글자수;k++){
 if(k==끝소리)sorimadi[i/2].last[k]++;
 }
}
```

그림 2. 글자단위 빈도수 측정 방법

유니코드 완성형에서의 글자단위 빈도수 정보를 글자단위 검색을 할 경우에 우선순위별로 살펴보면,

* 첫 번째 소리마디에서의 검색 순위

첫소리글자 -- ㅉ(245회) -> ㅃ(247회) -> ㅆ(279회) -> ㄸ(409회) -> ㄹ(426회).....

가운뎃소리글자 -- ㅐ(5회) -> ㅖ(64회) -> ㅙ(116회) -> ㅟ(266회) -> ㅓ(297회).....

끝소리글자 – ㄺ(1회) -> ㅋ(1회) -> ㅥ(2회) -> ㅄ(8회) -> ㄳ(10회).....

위와 같이 빈도수가 낮을수록 검색이 먼저 이루어지며, 빈도수가 같을 경우에는 순차적 검색이 이루어지도록 하였다.

2.3 빈도수정보를 이용한 글자단위 검색

빈도수를 적용한 검색 방법은 입력한 글자단위 중에서 가장 적은 검색횟수를 가지는 글자단위부터 검색하게 된다. 아래는 49,754개의 단어로 이루어진 한글사전에서 '학교' 를 검색하였을 때의 검색 순서이다.

① 빈도수를 적용하였을 때

ㅛ(1278개) -> ㅎ(87개) -> ㄱ(14개) -> ㄱ(1개) -> ㅏ(1개) -> 공백(1개)
= 총 검색횟수는 51135회

② 빈도수를 적용하지 않았을 때

ㅎ(3854개) -> ㅏ(753개) -> ㄱ(68개) -> ㄱ(9개) -> ㅛ(1개) -> 공백(1개)
= 총 검색횟수는 54239회

'학교'의 검색 횟수를 그림 3과 같은 그래프로 살펴보면 빈도수를 적용하였을 때의 검색 성능이 향상되었음을 알 수 있다.

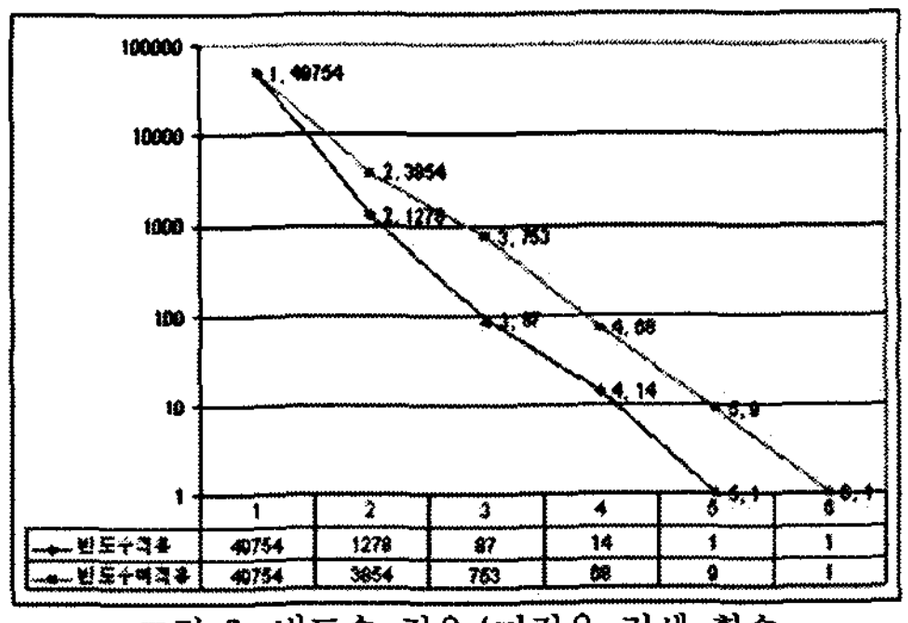

그림 3. 빈도수 적용/미적용 검색 횟수

빈도수를 적용한 경우의 검색횟수를 O(N)이라 할 때,

① N=Y 일 경우, N == O(N) 이다.
② N≠Y 일 경우, N > O(N) 이다.
N - 빈도수를 적용하지 않을 경우
Y - 빈도수를 적용할 경우

즉, 빈도수를 적용하지 않을 경우에는 빈도수를 적용한 경우와 글자단위의 검색 순서가 같은 경우에는 검색횟수가 같으나, 검색 순서가 일치하지 않을 경우에는 검색횟수가 많아지므로 검색 효율이 좋지 않다는 것을 알 수 있다.

2.4 데이터베이스에서 글자단위 검색 기능 구현

본 논문에서는 한글 글자단위 검색 기능의 사용 가능성을 검증해 보기 위해서 데이터베이스에서 한글 글자단위 검색 기능을 사용할 수 있도록 검색 기능을 구현해 보았다.

본 연구에서는 연구용 DBMS로 널리 사용되고 있는 포스트그레스 DBMS를 사용한다.

포스트그레스 DBMS에서 한글 글자단위 검색은 사용자 정의 연산자를 통해 구현이 가능한데, 사용자 정의 연산자를 만들기 위해서는 먼저 연산자를 위한 함수를 만들고 create function을 통해 함수를 정의하고 create operator를 사용하여 연산자를 정의하면 된다.

포스트그레스에서 사용자 정의 연산자로 사용할 기호는 ~ ! @ # % ^ & ' ? 가 사용된다. 두 자 이상의 연산자인 경우, | $: + - * / < > = 와 같은 31문자 이외의 문자도 사용할 수 있다.

그림 4는 사용자 정의 함수와 연산자를 만드는 구문이다. leftarg는 좌측항의 자료형, rightarg는 우측항의 자료형을 지정하고 procedure는 함수이름을 지정하며 나머지 명령어는 선택 사항이다. [5]

```
Create function func_name(type1, type N)
 Returns [setof] return_type
 AS 'sql-queries'
 Language 'c';

Create operator name
{
 procedure = func_name
 [, Leftarg = type 1][Rightarg = type2]
 [, Commutator = com_op]
 [, Negator = neg_op]
 [, Restrict = res_proc]
 [, Join = join_proc]
 [, Hashes]
 [, Sort = left_sort1_op]
 [, Sort2 = right_sort_op]
}
```

그림 4. 사용자 정의 함수/연산자의 구문

한글 글자 단위 검색을 위해서는 *, -, = 와 같은 3가지 연산자를 정의하고, 아래와 같은 형태로 검색할 수 있다.

첫소리-가운뎃소리-끝소리가 존재하고 가운뎃소리만 입력하여 3개의 단어로 된 글자를 찾는 경우

```
select *
from dic
where ((first = *) - (second = ㄱ) - (last = *))
        and number = 3;
```

위와 같은 형태로 하여 아래 표 1에 대한 글자단위 검색을 구현하였다.

표 1. 상가 테이블(shop_list)

shop_name	shop_address
나루터	서면
다모임	가야
알프스	덕천
로데오	만덕
또레세	남포동
•	•
•	•
•	•

* 'ㄷ'으로 시작하는 상호명을 찾는 경우

```
select *
from shop_list
where shop_name=((first='ㄷ') - (second = *) -
 (last = *)) and shop_address='가야';
```

라는 질의를 실행하였을 때 아래 표 2 와 같은 검색결과로부터, 데이터베이스에서 글자단위 검색을 활용하여 아래 표 2와 같이 'ㄷ'으로 시작하는 상호명과 주소가 '가야' 인 검색결과를 얻을 수 있다.

표 2. 'ㄷ'으로 시작하는 상호명 테이블

shop_name	shop_address
다모임	가야
돼지사냥	가야
동의일번가	가야
돈까스존	가야
동아리	가야

위와 같이, 데이터베이스에서 한글 글자단위 검색을 위한 사용자 정의 연산자를 구현함으로써, 다양한 검색 방법을 사용할 수 있다.

III. 결 론

이상에서 데이터베이스의 한글 글자단위 검색기능을 구현하는데 있어서, 필요한 입력 패턴들을 만들고, 글자단위 빈도수를 적용 하였을 때의 검색효율성과 데이터베이스에서 한글 글자단위 입력을 위한 사용자 정의 연산자에 대해서 살펴보았다.

데이터베이스에서 한글 글자단위를 기본으로 처리하고자 하는 경우에는 편리하게 사용될 수 있을 것이다.

참고문헌

[1] 이중화외 2인, 한글 글자 단위 인덱스를 위한 검색 유형 정의 및 한글 부호계와의 연관성에 관한 연구, 한국해양정보통신학회논문지, 제11권6호, pp.1083-1088, 2007

[2] 윤근수, 한국어 특성을 이용한 인덱싱 기법 연구, 연구논문집, 제27권 제1호 pp.145~162, 2000

[3] 김철수 외 1인, 이중 트라이를 이용한 한국어 단어 검색, 전북대학교 논문집, 제38호, pp.277~283, 1994

[4] 김경석, 컴퓨터 속의 한글 이야기 -둘째 보따리-, 부산대학교, pp. 32~33, 46~49, 1999

[5] Ishii Tatsuo, Postgresql, 영진닷컴, pp. 182~185, 2001