시소러스 데이터의 저장과 검색을 위한 효율적인 저장 시스템의 설계 및 구현

김점숙*, 안동언, 정성종
전북대학교 컴퓨터공학과
sun@calhp1.chonbuk.ac.kr, duan@moak.chonbuk.ac.kr

Design and Implementation of Efficient Storage System for Storing and Searching Thesaurus Data

Jum-Suk Kim*, Dong-Un An, Sung-Chung Jong
Dept. of Computer Engineering, Chonbuk University

요 약

본 논문에서는 시소러스를 효율적으로 구축하고 사용할 수 있는 시소러스 저장 구조를 구현하였다. 정보 검색 시스템은 사용자의 질의어를 용어들과 용어들 사이의 관계 집합으로 구성된 일종의 용어 사전인 시소러스를 이용하여 문헌에 대한 색인과 검색을 정확하고 통제된 용어 형태로 바꾸어 색인과 검색 작업의 효율을 높인다. 데이터베이스에 저장된 시소러스 구조에서 시소러스 검색을 할 때 속도가 감소하는데 이를 해시함수를 이용한 리스트 구조를 이용함으로써 전체적인 시소러스 검색 속도의 증가를 기대할 수 있다. 또한 현재 데이터베이스 형태의 시소러스를 다른 곳에 이식하려면 데이터베이스 시스템이 있어야 한다. 따라서 메모리에 올릴 수 있는 구조를 가지면 시소러스 보급에 기여 할 수 있다. 본 논문에서 제안한 데이터베이스에 저장된 시소러스 구조와 해시함수를 이용한 리스트 구조를 비교, 분석하고 보다 더 효율적인 시소러스의 역할 및 구조 형태에 대해 제안한다.

1. 서론

정보 검색 시스템은 사용자가 요구하는 정보를 효과적으로 검색하는 것을 목적으로 하고 있다. 요즘 인터넷으로 인한 정보의 양이 증가함에 따라 데이터베이스로부터 사용자 요구와 의미적으로 정확히 일치하는 정보만을 찾아 제공하는 시스템을 필요한다. 이러한 정보 검색 시스템에서 전문가가 아닌 일반 사용자가 원하는 정보에 대한 정확한 색인어를 질의어로 입력하는 것은 어려운 일이다. 따라서 모든 용어들의 의미 관계를 표현하는 시소러스가 검색 과정에 이용된다. 이러한 정보 검색 시스템이 가지고 있는 문서와 사용자의 질의를 검색하여 주는 정보화 시대에 꼭 필요한 것이다. 그러나, 최근에 사용자 질의에 표현된 정확한 용어(term)를 찾아주는 것은 검색속도에 의해서 좌우될 것이다.[1][2]

기존의 시소러스는 구축할 때 클라이언트/서버 구조에서 다수의 전문가가 구축을 한다. 이때 시소러스 저장을 데이터베이스를 사용한다. 사용자의 요구가 늘어날수록 서버에 부담이 늘어났고 그에 따라 좀더 효율적인 구조를 필요로 하게 되었다. 사용자의 질의에 도움을 주는 지식베이스인 시소러스를 데이터베이스에 저장된 시소러스 구조를 시소러스 검색에서 검색 효율을 높여 보고자 해시를 이용한 리스트 구조로 구축하고자 했다. 실제 구현을 통해 데이터베이스에 저장된 시소러스 구조에서 시소러스 작업을 수행 할 때보다 해시를 이용한 리스트 구조를 이용하여 구축한 시소러스가 전체적인 검색속도의 향상을 얻을 수 있음을 알았다. 또 하나의 목적은 시소러스의 보급이다. 현재 데이터베이스 형태의 시소러스는 다른 곳에 이식하려면 데이터베이스의 시스템이 꼭 필요하다. 따라서 메모리에 올릴 수 있는 해시함수를 이용한 리스트 구조를 구축하게 되었다. 그리하여 내부적인 구조를 보이지 않게 함으로써 시소러스 보급에 일조 할 수 있다. 많은 데이터를 벌크로딩(Importing)하여 저장해서도 효율적임을 확인했다. 본 논문의 구성은 다음과 같다. 2 장에서는 시소러스 관련연구에 대해 설명한다. 3 장에서는 데이터베이스에 저장된 시소러스 구조에서 시소러스를 구축하고 구현한 것을 보여준다. 4 장에서는 해시를 이용한 리스트 구조를 구축, 구현을 설명한다. 5 장에서는 데이터베이스에 저장된 시소러스 구조와 해시를 이용한 리스트 구조를 성능 평가 결과를 제시한다. 마지막으로 6 장에서는 결론과 향후계획에 대해 서술한다.

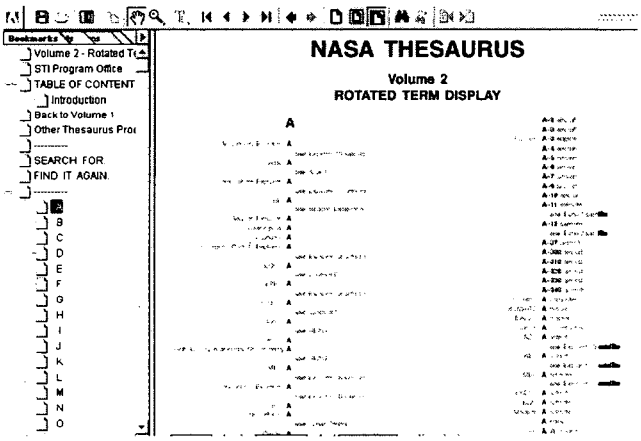
2. 관련연구

일반적으로 구축된 시소러스는 개념들 사이의 의미 관계를

상위어(BT: Broader Term), 하위어(NT: Narrower Term) 그리고 관련어(RT: Related Term) 관계로 표현하며, BT/NT 관계에 의해 개념계층이 형성된다.[1][4]대표적으로 "NASA Thesaurus", "INSPEC Thesaurus" 그리고 "TEST" 등이 있다.[1][3] 본 절에서는 INSPEC 시소러스와 NASA 시소러스에 대해 살펴본다.

INSPEC은 물리학, 전기, 전자공학, 컴퓨터 그리고 통신 분야에 있어 가장 권위 있는 정보 데이터베이스로서 4,000여종 이상의 기술 저널과 2,000여종의 회의록, 단행본, 학위 논문 그리고 보고서에 대한 색인 및 초록 정보를 제공하고 있다. 이 데이터베이스를 관리할 수 있는 INSPEC 시소러스는 약 8,000개의 도메인 개념을 포함하고 최대 깊이는 6레벨이며 RT 관계와 NT/BT 관계를 가지는 개념들로 존재한다.

NASA 시소러스는 미국 항공 우주국 (NASA: National Aeronautics and Space Administration)의 STI(Scientific & Technical Information) 문서 데이터베이스를 효과적으로 검색하기 위해 활용되는 도메인 지식이다. 이 계층화된 시소러스는 약 9,000개의 개념들로 구성되어 있다. 여기서, 개념들 사이의 의미 관계는 종속 관계(GS: Generic Structure), 연관어 관계(RT: Related Term) 그리고 동의어 관계(UF: User For)로 표현한다.[1][3]



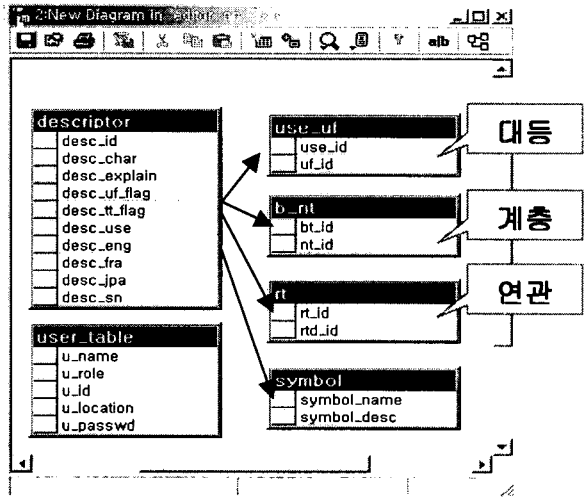
[그림 1] "NASA Thesaurus" 시소러스 구조

[그림 1]과 같이 "NASA Thesaurus"는 계층구조를 보여주고 있으며 데이터베이스에 저장된 구조를 가지고 있는 시소러스이다. 마찬가지로 "INSPEC Thesaurus"와 "TEST"도 데이터베이스에 저장된 구조를 가지고 있다.

3. 데이터베이스에 저장된 시소러스 구조의 시소러스 설계

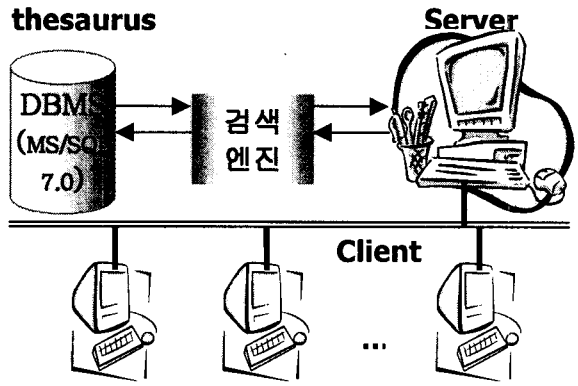
3.1 설계 및 구조

[그림 2]의 데이터베이스에 저장된 시소러스는 비주얼 베이직(Visual Basic 6.0)에서 RDO(remote data object)라는 class를 이용하여 ODBC(open database connectivity)를 통해 데이터베이스와 연결하는 메커니즘을 사용하였다.



[그림 2] 데이터베이스에 저장된 시소러스 구조

데이터베이스에 저장된 시소러스 구조는 [그림 2]와 같다. 데이터베이스에 저장된 시소러스 구조의 검색 시스템에서 시소러스를 이용할 때 하나의 서버에 여러 클라이언트가 연결되어 있는 [그림 3]과 같은 구조를 생각할 수 있다.



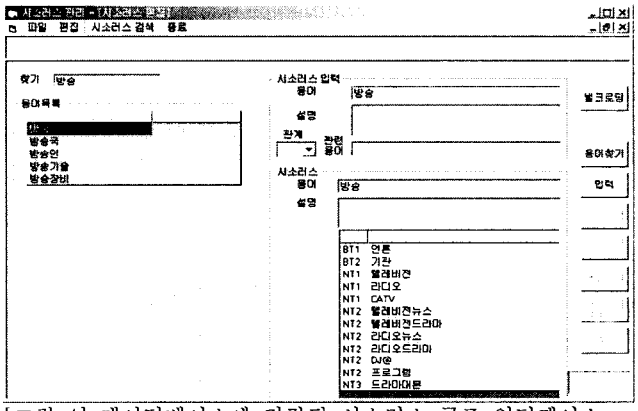
[그림 3] 데이터베이스(MS/SQL 7.0)과 비주얼 베이직(Visual Basic 6.0) 연동 구조

먼저 검색서버가 거대한 양의 문서집합과 검색엔진, 색인파일, 시소러스를 가지고 클라이언트로부터의 요청을 처리하는 것이다. 사용자는 클라이언트에 접속하여 질의를 하고 클라이언트는 단순히 이러한 사용자의 요구를 서버와 연계하여 서비스를 요청하고 받아오는 작업을 수행한다. 시소러스는 클라이언트로부터 받은 질의어를 확장하여 보다 정확한 결과 리스트를 클라이언트에게 돌려준다. 질의를 했을 때 데이터베이스로부터 클

라이언트까지 검색 결과가 상당한 시간이 소요됨을 알 수 있었다.

3.2 데이터베이스에 저장된 시소러스 구조 검색 인터페이스

[그림 4]의 질의 인터페이스는 데이터베이스에 저장된 시소러스 구조를 이용하여 데이터베이스(MS/SQL 7.0)와 비주얼 베이직(Visual Basic 6.0)을 연동하여 만든 것이다. 10,000 개의 데이터를 데이터베이스(MS/SQL 7.0)에 저장하고 사용자가 클라이언트에서 질의를 하면 용어(term)에 관계된 데이터를 찾아와 질의를 수행한 결과이다.



[그림 4] 데이터베이스에 저장된 시소러스 구조 인터페이스

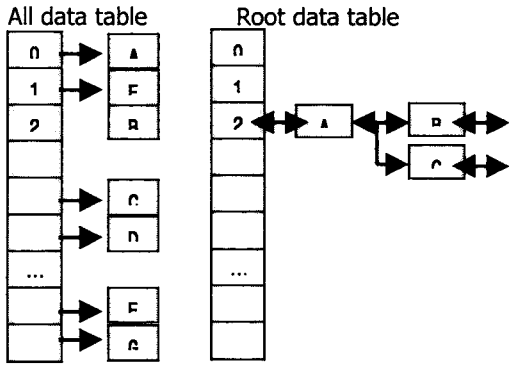
서버는 여러 클라이언트들로부터 들어온 작업들을 순서대로 처리하여 서비스하는 일을 한다. 이때 서버는 용어(term)를 찾아주는데 많은 시간을 소비한다. 데이터베이스(MS/SQL 7.0)와 비주얼 베이직(Visual Basic 6.0)을 연동하여 만든 이 시소러스는 많은 양을 저장할 수 있지만 검색 효율면에서는 떨어진다. 특히 다수의 클라이언트가 동시에 질의를 했을 때는 많은 시간을 기다려야 함으로 서버에서의 작업이 부담이 될 거라고 예상된다.

4. 해시를 이용한 리스트 구조의 시소러스 설계

4.1 설계 및 구조

앞에서 보여준 데이터베이스에 저장된 시소러스 구조의 문제점을 해결하고자 [그림 5]의 해시를 이용한 리스트 구조를 제안한다.

[그림 5]에서 왼쪽에 있는 All data table에는 모든 데이터가 해시값에 해당하는 각각의 버켓에 저장되는 것이다. 해시의 가장 중요한 점은 모든 데이터가 균형있게 분포되게 하는 점이다. 또한 중복되는 데이터가 있으면 저장되지 않도록 설계했다. 오른쪽에 있는 Root data table에는 All data table에 저장되어 있는 데이터를 관계에 의해 다시 해시함수에 저장하는 것이다. 즉 루트를 가지고 있는 것이다. 하나의 버켓에 루트를 가지는 데이터가 트리의 형태로 매달려 있다. 여기에서 각각의 용어(term)는 더블 링크드 리스트 구조로 연결되어 있다. 따라서 검색을 하면 하나의 용어(term)를 중심으로 BT, NT를 모두 찾는다.



[그림 5] 해시를 이용한 리스트 구조

[그림 5]의 해시를 이용한 리스트 구조에서 All data table에 모든 데이터를 저장하고 구조는 다음과 같다.

```
typedef struct _hash_all_data
{
    char term[MAXNAME + 1];
    int root_bucket;
    struct _hash_all_data *next;
} hash_all_data;
```

[그림 5]의 해시를 이용한 리스트 구조에서 Root data table에 루트에 관계된 데이터를 저장하고 구조는 다음과 같다.

```
typedef struct _hash_root_table
{
    char term[20];
    struct _hash_root_table *prev;
    struct _hash_root_table *next[MAX_LEAF];
} hash_root_table;
```

해시를 이용한 리스트 구조의 해시함수 구조는 다음과 같다.

```
int hash_func(char *str) // 해쉬 함수의 정의
{
   int h = 0;
   while (*str)
   h += *str++;
   return abs(h % HASH_SIZE);
}
```

논문에서 구축한 해시를 이용한 리스트 구조에서 정의한 함수는 일반적으로 개념들 사이의 의미 관계를 상위어(BT: Broader Term), 하위어(NT: Narrower Term), 관련어(RT: Related Term), 비우선어(UF: Used for), 그리고 우선어(USE: use)로 이루어진다. 각각의 함수를 다음과 같이 정의한다.

```
void root_view()     // BT/NT를 모두 보여주는 함수
void root_view_nt()  // NT만 보여주는 함수
```

```
void root_view_bt()  // BT만 보여주는 함수
void root_view_rt()  // RT만 보여주는 함수
void root_view_uf()  // UF만 보여주는 함수
void root_view_use() // USE만 보여주는 함수
```

4.1.1 BT/NT를 모두 보여주는 함수

```
void root_view(){
  for( i = 0 ; i < HASH_SIZE ; i++)
  {
      printf("\nBucket %d:", i);
      t = &root_table[i];
      t = t->next[0];
      if(t == NULL) continue;
      put(t);
      while(!is_queue_empty())
      {
        t = get();
        if( !strcmp( t->term, input_key) ){
          toggle = 1;
          printf("\n%s ->", t->term);
          bt_iterator = t;
          }
      }
  }
}
```

이 함수에서는 계층관계를 모두 보여줌으로써 찾고자 하는 용어를 중심으로 NT와 BT를 HASH_SIZE만큼 보여줌으로써 용어의 위치를 쉽게 알 수 있다.

4.1.2 NT를 모두 보여주는 함수

```
void root_view_nt(){
  if(!strcmp(rel_key, "NT") || !strcmp(rel_key, "nt"))
  {
    init_queue();
    for( i = 0 ; i < HASH_SIZE ; i++)
    {
       printf("₩nBucket %d:", i);
       t = &root_table[i];
       t = t->next[0];
       if(t == NULL) continue;
       put(t);
       }
  }
}
```

이 함수에서는 용어(term)를 검색하였을 때 NT관계만을 보여준다. 최근 인터넷 검색에서 검색결과가 너무 많이 나올 때 범위를 좁게 해서 볼 필요가 있을 것이다. 이때 용어(term)의 NT를 찾아서 보면 적은 정보를 얻을 수 있다.

4.1.3 BT를 모두 보여주는 함수

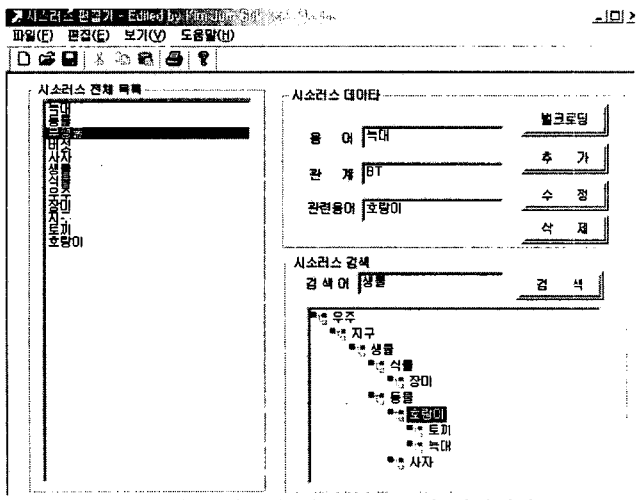
```
void root_view_bt(){
  if(!strcmp(rel_key, "BT") || !strcmp(rel_key, "bt"))
  {
    init_queue();
    for( i = 0 ; i < HASH_SIZE ; i++)
    {
       while(!is_queue_empty()){
          t = get();
          if( !strcmp( t->term, input_key) ){
             toggle = 1;
             bt_iterator = t;
           }
       }
  }
}
```

이 함수에서는 용어(term)를 검색하였을 때 BT관계만을 보여준다. 최근 인터넷 검색에서 검색결과가 너무 적게 나올 때 범위를 넓게 해서 볼 필요가 있을 것이다. 이때 용어(term)의 BT를 찾아서 보면 많은 정보를 얻을 수 있다.

4.2 해시를 이용한 리스트 구조의 검색 인터페이스

[그림 6]는 해시를 이용한 리스트 구조의 시소러스 인터페이스이다. [그림 6]에서 많은 데이터를 효율적으로 저장하는 벌크로딩(Importing) 기능을 넣었다. 모든 데이터를 메모리에 올린다음 하나의 용어(term)를 검색하여 질의를 수행한 결과를 트리 구조에 뿌려주었다. 그 결과 모든 용어가 메모리에 올라가 있기 때문에 검색 속도가 훨씬 빨라짐을 확인했다.

또한 용어의 관계를 트리 구조로 보여줌으로써 사용자의 편의를 제공했다. 용어(term)를 검색하면 트리 구조로서 데이터의 상하관계를 보여주게 되어 있다.



[그림 6] 해시를 이용한 리스트 구조의 인터페이스

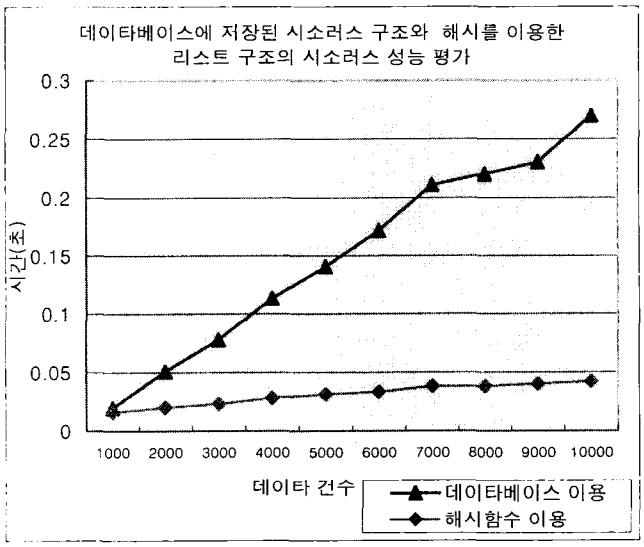
5. 데이터베이스에 저장된 시소러스 구조와 해시를 이용한 리스트 구조의 시소러스 성능 평가

실험한 조건은 펜티엄 450에서 측정한 결과이다. 각각의 구조에 데이터를 저장 할 때에는 동일한 조건으로 데이터(용어:term)의 크기가 같고 개수도 같게 했다.

데이터베이스에 저장된 시소러스 구조와 해시를 이용한 리스트 구조에 동일한 데이터를 넣었다. 먼저 데이터베이스에 저장

된 시소러스 구조에 데이터를 벌크로딩(Importing)한 결과는 상당한 시간이 소요 되었다. 그러나 해시를 이용한 리스트 구조는 아주 적은 시간에도 많은 데이터가 삽입됨을 알았다.

다음으로 데이터베이스에 저장된 시소러스 구조와 해시를 이용한 리스트 구조에서 용어(term)를 검색하여 트리구조에 뿌려주는 시간을 [그림 7]와 같이 비교해 보았다.



[그림 7] 데이터베이스에 저장된 시소러스 구조와 해시를 이용한 리스트 구조의 성능 평가 결과

[그림 7]에서는 데이터베이스에 저장된 시소러스 구조에서 20 개의 데이터를 검색하였다. 그 결과 검색 속도가 현저히 감소하는 걸 알 수 있었다. 또한 해시를 이용한 리스트 구조도 마찬가지로 같은 데이터를 검색한 결과 검색 속도가 별차이가 없음을 알 수 있었다. 그래프에서는 보이지 않았지만 더 많은 데이터를 삽입해서 검색해도 결과는 달라지지 않았다.

따라서 [그림 7]와 같이 데이터베이스에 저장된 시소러스 구조가 해시를 이용한 리스트 구조보다 검색에서도 효율적이었다. 해시를 이용한 리스트 구조는 모든 데이터를 메모리리 올려서 검색하였다. 반면에 데이터베이스에 저장된 시소러스 구조는 모든 데이터를 데이터베이스(MS/SQL 7.0)를 이용하여 저장한 다음 클라이언트쪽에서 질의어를 던지면 데이터베이스에서 찾아와 화면에 보여주게 되어있다. 그래서 데이터베이스에 저장된 시소러스 구조에서의 검색이 비효율적임을 알 수 있었다.

단어의 개수는 1,000 개부터 10,000 개까지 각각 동일한 데이터를 넣어 보았다. 그 결과는 많은 데이터를 넣을수록 데이터베이스(MS/SQL 7.0)에서 데이터를 찾아서 클라이언트쪽으로 보여주는 검색 속도는 현저하게 데이터베이스에 저장된 시소러스 구조가 떨어짐을 확인했다.

6. 결론 및 향후과제

검색 시스템에서 시소러스의 사용은 검색 효율에 중요한 영향을 미친다. 따라서 효과적으로 구축하고 검색 속도를 증가시키는 것이 중요한 의미를 가진다.

본 논문에서는 시소러스 검색에서 처리 속도의 향상을 위해 해시를 이용한 리스트 구조를 제안한 것이다. 먼저 데이터베이스에 저장된 시소러스 구조에서 데이터베이스(MS/SQL 7.0)에 데이터를 저장하고 클라이언트에서 용어(term)를 검색한 결과 처리속도가 떨어짐을 알았다. 이 문제점을 해결하기 위해 해시를 이용한 리스트 구조에서는 모든 데이터를 메모리에 올렸다. 따라서 시소러스 전체 검색의 처리속도에서 효율적임을 성능평가를 통해서 얻었다. 또한 용어의 수가 대규모인 시소러스의 경우도 가능했다. 다만 해시를 이용한 리스트 구조에서 모든 데이터를 메모리에 올리기 때문에 메모리 크기에 영향을 받는다.

향후 과제로는 인터넷 시대에 살고 있는 우리로써 검색 속도 향상은 꼭 필요한 조건이다. 사용자가 사용할 때 편리한 인터페이스와 시소러스의 보급이다. 또한 데이터베이스에 저장된 시소러스 구조에서도 서버와 클라이언트가 연동시 구조 설계에 효율적인 방법을 이용하면 좀 더 나은 처리속도를 기대 할 수 있을 것이다.

참고문헌

[1] 최재훈, "객체기반 시소러스 관리 시스템의 설계 및 구현에 관한 연구" 2000

[2] 장유진, "클라이언트/서버 환경에서 효율적인 시소러스 작업 모형 설계" 한국정보과학회 추계 학술발표 논문집 P.157-159, 1999

[3] J. L Milstead, "Specifications for Thesaurus Software ", Information Procesing and Management, Vol.27, No.2/3, pp. 165-175, 1991

[4] D. A. Krooks and F. W. Lancaster, "The Evolution of Guidelines for Thesaurus Construction," Libri, Vol.43, No.4, pp.326-42, 1993

[5] 정재헌, "정보검색을 위한 효율적인 시소러스 구조에 관한 연구",한국정보과학회 춘계 학술발표 논문집 Vol. 22 No.1. pp.949-952, 1995

[6] 맹성현, "정보검색의 의미 및 관련 시스템의 소개", 네트웍 타임즈, pp140-143, 1995.11

[7] Brachman R.J. and Levesque H.J., "Readings in Knowledge Representation", Morgan Kaoufmann, Los Altos, CA, 1985.