Journal of Korea Multimedia Society Vol. 22, No. 2, February 2019(pp. 225-230)
https://doi.org/10.9717/kmms.2019.22.2.225

그래프이론에 의한 데이터베이스 세그먼트 분산 알고리즘

김 중 수†

Database Segment Distributing Algorithm using Graph Theory

Joong Soo Kim†

ABSTRACT

There are several methods which efficiencies of database are uprise. One of the well-known methods is that segments of database satisfying a query was rapidly accessed and processed. So if it is possible to search completely parallel multiple database segment types which satisfy a query, the response time of the query will be reduced. The matter of obtaining CPS(Completely Parallel Searchable) distribution without redundancy can be viewed as graph theoretic problem, and the operation of ring sum on the graph is used for CPS. In this paper, the parallel algorithm is proposed.

Key words: Database, Graph Theory, Parallel Algorithm,

1. 서 론

일반적으로 컴퓨터에서 데이터를 저장하기 위해서는 용도에 따라서 페이지나 세그먼트로 저장한다. 그러나 데이터베이스 시스템에서는 사용자 요구가 더 중요한 요소이므로 세그먼트 형태로 보관하는 것이 효율적이다. 따라서 사용자가 요구하는 하나의 쿼리(query)를 처리하기 위해서는 여러 곳에 분배되어 있는 세그먼트들을 읽어 와야 한다. 이러한 여러 세그먼트를 여러 컴퓨터나 기기에 분배하는 방법은 응답속도와 처리의 신뢰성에도 많은 영향을 미친다[1,2].

지금까지 데이터 베이스에서는 각 쿼리를 처리하기 위해서는 쿼리가 필요로 하는 세그먼트를 효율적으로 분배하는 방법들이 연구 되고 있다. 데이터의 효과적인 분배를 위하여 데이터 중복을 줄이고 데이터 일관성을 높이는 방법[3], 빠른 데이터 처리를 위하여 집중도를 높이는 방법[4]가 소개되었다. 또 공간관계와 공간조인을 위하여 중복 및 이중 데이터 처리방법을 보여주고 있다[5]. 바이오 데이터를 처리하는 방법으로 중복처리감소를 위한 단백질 순서 데이타셋(protein sequence datasets)를 소개하기도 하였다[6]. 검색능력과 불필요한 중복간의 상쇄[tradeoff] 관계를 보여주고 또한 정보의 이론적인 특징 평가 전략을 제시 하였다[7]. 증가하는 갱신 전달을 위해서 소프트웨어 디자인으로 데이터베이스를 구축하려는 연구[8]도 나타나고 있다.

그리고 데이터베이스에서 쿼리의 빠른 처리를 하기 위하여 각 세그먼트의 분배문제를 해결하기 위하여 노력한 연구들도 있다. Ghosh는 하나의 을 만족하는 데이터 베이스 세그먼트들을 중복 없이 분배하는 방법으로 알고리즘1, 2를 제안[9]하였고, Srinivasan과 Sankar는 이 알고리즘을 보다 수행속도가 개선된 알고리즘 M1, M2[10]를 발표했다. 이 방법들은 하나

※ Corresponding Author : Joong Soo Kim, Address: (36729) Kyungdong-ro 1375, Andong Kyungbuk, Korea, TEL : +82-54-820 -5476, FAX : +82-54-820-6154, E-mail : kimjs@andong.ac.kr

Receipt date : Jan. 14, 2019, Approval date : Jan. 23, 2019
† Dept. of Computer Engineering Andong National University
※ This work was supported by a Research Grant of Andong National University

의 쿼리을 처리하기 위하여 각 세그먼트를 중복없이 완전 병렬로 접근이 가능하도록 각 세그먼트를 분배할 수 있 알고리즘들이다. 이 두 연구들은 이 분배 문제를 행렬을 이용하여 해결을 하고 있다. 그리고 이러한 분배를 절점과 호선의 그래프 이론의 문제로 볼 수도 있다.

본 논문은 그래프이론을 이용하여 완전 병렬 접근이 가능한 데이터베이스 세그먼트 분배 알고리즘을 Gr1 개발하고 그 수행과정을 예시하였다. 이 알고리즘도 세그먼트의 중복없이 각 기기에 분배될 수 있도록 개발되었다. 알고리즘 Gr1의 시간 및 공간 복잡도를 고찰하였다.

2. 그래프 이론 및 알고리즘 Gr1

2.1 그래프 이론

이 알고리즘에 필요한 정의는 다음과 같다.

정의1. 그래프

각 쿼리의 목표세그먼트(target segment)는 그래프의 절점(vertex)이 되고 임의의 2개 세그먼트가 한 쿼리의 목표세그먼트로 되면 두 절점은 호선(edge)으로 연결된다. 그리고 이것은 절점의 집합 V와 호선의 집합 E로 표시된다. 따라서 그래프는 $G=(V,E)$로 된다.

정의2. 그래프의 링섬(ring sum)

2개의 그래프가 $G_1=(V_1,E_1)$, $G_2=(V_2,E_2)$일 때,

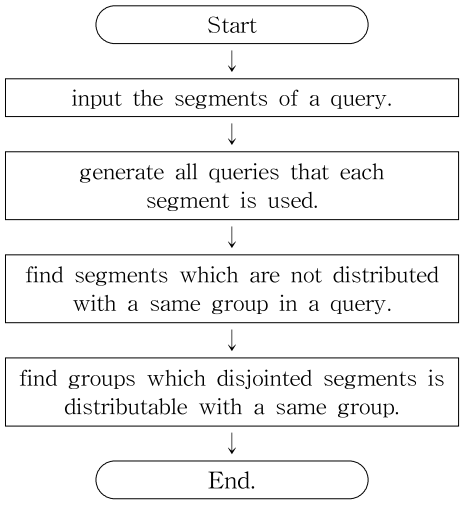

Fig. 1. flowchart of Gr1.

두 그래프의 링섬은 다음과 같이 표현된다.

$$G_1 \oplus G_2 = (V_1 \cup V_2, (E_1 \cup E_2) - (E_1 \cap E_2)) \quad (1)$$

여기서 $\oplus$는 링섬의 연산자를 의미한다.

주어진 쿼리 전체에 대해서 정의1을 적용하면 쿼리 그래프가 만들어 진다. 이것을 g라 하고 g의 완전 그래프(complete graph)를 G라 할 때 정의2를 적용하면 G의 보 그래프(complement graph of the query graph)가 얻어진다[11,12].

이 보 그래프에서 같은 절점이 없는 완전 부분그래프(disjoint complete subgraph)를 찾아서 이를 군 그래프라 하면 각 군에는 완전평행검출이 될 수 있도록 세그먼트들이 분배된다. 이러한 원리로 순서도로 나타내면 아래와 같다.

2.2 알고리즘 Gr1

위의 순서도를 알고리즘으로 나타내었다.

```
Algorithm Gr1
   For  Q∈(Q1, Q2,..., Qr) do
   For  S∈(S1, S2,..., Sn) do
SMT1  IF S∈Test(Q) → Mset(S): extend(Q);
   rof
   rof
   V : = (S1, S2,..., Sn) :
   C : = (  ) ;
   Flag(S): = 1 ;
   Gset : = (  ) :

SMT2 For  V//C≠Φ→ X
          Flag(X) : pop
          V//C : pop(X);
          (If V ≠Φ→ X, V: pop(X);
          If C ≠Φ→ X, C: pop(X);)
   For  Q∈Mset(X) do
SMT3  S: = Test(Q)
   For  Y ∈ S do
SMT4  If  Y = X → Test(Q):pop(X);
SMT5  If  Flag(Y) = 1 → V : pop(Y)
        ( Y∈ V)
          C : extend(Y)
          Flag(Y) : = 0
          Conf(Y) : extend(X);
SMT6  If Flag(Y) = 0 → If X∉Conf(Y) → Conf(Y):
        extend(X):
        ( Y∈ C)
```

```
    rof
    rof
SMT7 Arrange(X);
    rof.

    Arrange(X)
    if (∃G∈ Gset) disj (G,Conf(x)) → G:extend(X) ;
    (∀G∈ Gset) cor (G,Conf(x)) →
                        Gset:extend(G=X);
    fi
    (found = false
    do Gset ≠Φ and not false → Gset:pop(G)
    found: = disj (G,Conf(X));
    od
    if found → G:extend(X)
     not found → Gset;extend(G=X);)
```

※ 참고로 위 알고리즘에 사용된 구문들은 for ~ rof , do ~ od , if ~fi 로 각 구문이 이루어 진다.

각 SMT의 역할을 살펴보면,

SMT1은 입력 $Test(Q)$를 $Mset(S)$로 만들고 SMT3에서 $Test(Q)$와 $Mset(S)$가 입력으로 다시 사용하게 된다.

SMT2는 각 세그먼트를 V와 C에서 X에 할당해 주고 할당된 세그먼트와 그 $Flag(X)$를 제거한다. SMT3는 $Mset(Q)$에 속한 Q에 대해서 다시 $Tset(Q)$의 원소 S를 찾아내는 역할을 한다. SMT4는 $Tset(Q)$의 원소와 X가 같을 경우 그 원소를 제거한다.

SMT5는 $Flag(Y)$=1이면 처음으로 Y와 X가 충돌관계에 있으므로 $Conf(Y)$에 X를 넣고 충돌관계가 다른 S와 더 있는 가를 알아보기 위하여 Y를 C로 보낸다. SMT6은 $Flag(Y)$=0이면 Y는 C에 속하는데 X가 $Conf(Y)$에 속하지 않을 경우 $Conf(Y)$에 X를 더 첨가시킨다. SMT7은 $Arrange(X)$로서 알고리즘 Gr1의 예에서 자세히 설명된다.

3. 알고리즘 Gr 1의 수행

예를 들어 각 쿼리에 대한 목표세그먼트가 아래와 같을 때

$$Q_1 = (S_1, S_2, S_3, S_4)$$
$$Q_2 = (S_2, S_5, S_6)$$
$$Q_3 = (S_1, S_5, S_7)$$
$$Q_4 = (S_1, S_6)$$

정의1에 의해서 쿼리그래프를 g를 만들면 Fig. 2와 같고 정의2에 의해서 g의 완전그래프와 보그래프는 Fig. 3과 Fig. 4와 같다.

Fig. 4에서 처리하는 순서를(S_1, S_2, S_3, S_4, S_5, S_6, S_7)으로 하면 같은 절점이 없는 완전부분그래프는

$$G_1 = (S_1)$$
$$G_2 = (S_2, S_7)$$
$$G_3 = (S_3, S_5)$$
$$G_4 = (S_4, S_6)$$

가 된다.

또 처리하는 순서를 다르게 (S_1, S_3, S_2, S_4, S_6, S_7, S_5)와 같이 하면 같은 절점이 없는 완전부분 그래프는 Fig. 4에서 같은 절점이 없는 완전부분그래프가 다르게 선정되어

$$G_1 = (S_1)$$
$$G_2 = (S_3, S_6, S_7)$$
$$G_3 = (S_2)$$
$$G_4 = (S_4, S_5)$$

로 된다.

다음 내용들은 위의 예제 쿼리에 알고리즘 Gr1을 적용하였다.

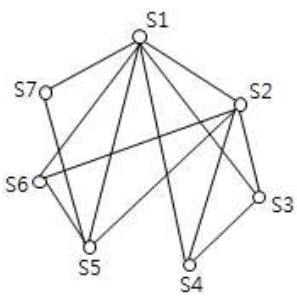

Fig. 2. Query graph g.

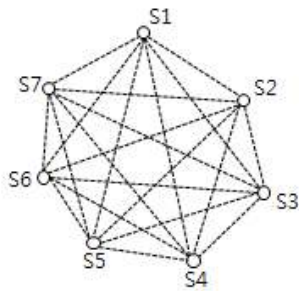

Fig. 3. Complete graph of g.

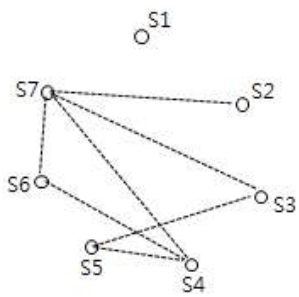

Fig. 4. Complement graph of g.

먼저 입력으로

$Test(Q_1) = (S_1, S_2, S_3, S_4)$
$Test(Q_2) = (S_2, S_5, S_6)$
$Test(Q_3) = (S_1, S_5, S_7)$
$Test(Q_4) = (S_1, S_6)$

가 들어가면 SMT1에서 다음과 같은 Mset(S)를 만들어 낸다.

$Mset(S_1) = (Q_1, Q_3, Q_4)$
$Mset(S_2) = (Q_1, Q_2)$
$Mset(S_3) = (Q_1)$
$Mset(S_4) = (Q_1)$
$Mset(S_5) = (Q_2, Q_3)$
$Mset(S_6) = (Q_2, Q_4)$
$Mset(S_7) = (Q_3)$

여기서 $Test(Q)$와 $Mset(S)$는 대칭관계에 있다.

이 $Tset(Q)$와 $Mset(S)$는 STM3의 입력으로 사용되는데 각 X에 대해서 SMT6까지 나온 중간결과는 다음의 Fig. 5에서 11까지 이다.

$Conf(S_1) = \Phi$
$Conf(S_2) = (S_1)$
$Conf(S_3) = (S_1)$
$Conf(S_4) = (S_1)$
$Conf(S_5) = (S_1)$
$Conf(S_6) = (S_1)$
$Conf(S_7) = (S_1)$

Fig. 5. when $X=S_1$, $Conf\ Set$

$Conf(S_1) = \Phi$
$Conf(S_2) = (S_1)$
$Conf(S_3) = (S_1, S_2)$
$Conf(S_4) = (S_1, S_2)$
$Conf(S_5) = (S_1, S_2)$
$Conf(S_6) = (S_1, S_2)$
$Conf(S_7) = (S_1)$

Fig. 6. when $X=S_2$, $Conf\ Set$

$Conf(S_1) = \Phi$
$Conf(S_2) = (S_1)$
$Conf(S_3) = (S_1, S_2)$
$Conf(S_4) = (S_1, S_2, S_3)$
$Conf(S_5) = (S_1, S_2)$
$Conf(S_6) = (S_1, S_2)$
$Conf(S_7) = (S_1)$

Fig. 7. when $X=S_3$, $Conf\ Set$

$Conf(S_1) = \Phi$
$Conf(S_2) = (S_1)$
$Conf(S_3) = (S_1, S_2)$
$Conf(S_4) = (S_1, S_2, S_3)$
$Conf(S_5) = (S_1, S_2)$
$Conf(S_6) = (S_1, S_2)$
$Conf(S_7) = (S_1)$

Fig. 8. when $X=S_4$, $Conf\ Set$

$Conf(S_1) = \Phi$
$Conf(S_2) = (S_1)$
$Conf(S_3) = (S_1, S_2)$
$Conf(S_4) = (S_1, S_2, S_3)$
$Conf(S_5) = (S_1, S_2)$
$Conf(S_6) = (S_1, S_2, S_5)$
$Conf(S_7) = (S_1, S_5)$

Fig. 9. when $X=S_5$, $Conf\ Set$

$Conf(S_1) = \Phi$
$Conf(S_2) = (S_1)$
$Conf(S_3) = (S_1, S_2)$
$Conf(S_4) = (S_1, S_2, S_3)$
$Conf(S_5) = (S_1, S_2)$
$Conf(S_6) = (S_1, S_2, S_5)$
$Conf(S_7) = (S_1, S_5)$

Fig. 10. when $X=S_6$, $Conf\ Set$

$Conf(S_1) = \Phi$
$Conf(S_2) = (S_1)$
$Conf(S_3) = (S_1, S_2)$
$Conf(S_4) = (S_1, S_2, S_3)$
$Conf(S_5) = (S_1, S_2)$
$Conf(S_6) = (S_1, S_2, S_5)$
$Conf(S_7) = (S_1, S_5)$

Fig. 11. when $X=S_7$, $Conf\ Set$

SMT7에서 처리하는 순서를 $(S_1, S_2, S_3, S_4, S_5, S_6, S_7)$으로 하면 $X=S_1$일 때 Fig. 5에서 $Conf(S_1)=\Phi$ 이므로 S_1은 G_1에 들어간다.

$X=S_2$ 일 때 Fig. 6에서 $Conf(S_2)=(S_1)$ 이므로 새로운 G_2에 들어간다.

$X=S_3$ 일 때 Fig. 7에서 $Conf(S_3)=(S_1, S_2)$ 이므로 S_3는 새로운 G_3에 들어간다.

$X=S_4$ 일 때 Fig. 8에서 $Conf(S_4)=(S_1, S_2, S_3)$ 이므로 S_4는 새로운 G_4에 들어간다.

$X=S_5$ 일 때 Fig. 9에서 $Conf(S_5)=(S_1, S_2)$ 이므로 S_5은 G_3에 들어간다.

$X=S_6$ 일 때 Fig. 10에서 $Conf(S_6)=(S_1, S_2, S_5)$이므로 S_6은 G_4에 들어간다.

$X=S_7$ 일 때 Fig. 11에서 $Conf(S_7)=(S_1, S_5)$ 이므로 S_7은 G_2에 들어간다.

여기서 각 그룹G를 정리하면

$G_1 = (S_1)$
$G_2 = (S_2, S_7)$
$G_3 = (S_3, S_5)$
$G_4 = (S_4, S_6)$

가 된다.

SMT2에서 처리하는 순서를 $(S_1, S_3, S_2, S_4, S_6, S_7, S_5)$로 바꾸면 같은 방법에 의해 각 그룹G는

$G_1 = (S_1)$
$G_2 = (S_3, S_6, S_7)$
$G_3 = (S_2)$
$G_4 = (S_4, S_5)$

로 된다.

4. 알고리즘 Gr1의 고찰

알고리즘 Gr 1을 알고리즘 1, 알고리즘 M1의 시간 및 공간 복잡도에 대해 비교하면 다음과 같다.

먼저 시간 복잡도에서 알고리즘 Gr 1을 살펴 본다.

SMT1의 수행수는 $\sum_Q \#Tset(Q)$와 $\sum_S \#Tset(S)$의 합이므로 $2n^2$이다.

SMT2의 수행수는 $3n$이다.

SMT3의 수행수는 $\sum_S \sum_{Q \in Mset(s)} \#Test(Q)$이다. 이 SMT의 수행수는 수행과정 중에서 계속 변하므로 정확한 수를 알 수 없으나 알고리즘 Gr1에서 $S := Test(Q)$에 해당하는 수행과정은 $Y \in S$에 해당하므로 Y에 대한 각각의 수행수를 모두 합하면 된다. 그러므로 SMT3의 수행수는 SMT4의 수행수, SMT5의 수행수와 SMT6의 수행수를 합한 것이다.

SMT4의 수행수는 $\sum_Q \#Tset(Q)$이므로 n^2이다.

Fig. 12에서 SMT5의 수행수는 $4(n-1)$이다.

SMT6의 수행수는 $1/2(n-2)(n-1)$이다.

SMT7의 수행수는 아래삼각행렬의 검출수와 각 그룹에 할당하는 수의 합이므로 $1/2(n-1)+n$이 된다.

알고리즘 Gr1의 전체 수행수는 각 SMT의 수행수를 합한 것과 같으므로 그 결과는 $11/2n^2+17/2n-6$이 된다. 이 수를 Big O로 나타내면 $O(n^2)$이 된다.

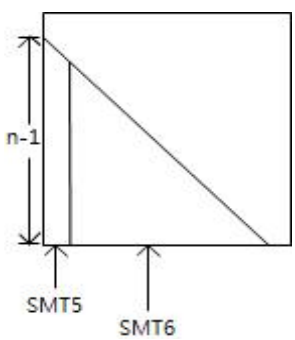

Fig. 12. $Conf(X)$.

그런데 작은 수의 컴퓨터에 여러 개의 세그먼트수는 $\sum_Q \#Tset(Q)$이므로 이 수가 $1/4n^2 \sim 1/2n^2$ 정도가 되면 효율적인 분배가 된다. 따라서 세그먼트수를 $1/3n^2$정도로 가정하면 몇 개의 SMT들의 수행수는 아래와 같이 바뀐다.

SMT1의 수행수는 $\sum_Q \#Tset(Q) = \sum_S \#Tset(S) \simeq 1/3n^2$ 이므로 $2n^2$에서 $2/3n^2$으로 된다.

SMT4의 수행수는 $\sum_Q \#Tset(Q) \simeq 1/3n^2$이 되고 SMT3까지 고려하면 $2n^2$에서 $2/3n^2$으로 된다.

SMT5의 수행수는 $1/2n^2$에서 $1/3n^2$으로 된다. 또 SMT7을 고려하면 $11/2n^2$은 아래와 같다.

$$11/2n^2-(2n^2-2/3n^2)\times 2-(1/2n^2)\times 2 = 5/2n^2$$

알고리즘 1의 경우를 보면 이 알고리즘에서는 행렬의 곱셈을 하므로 만약 행렬이 n×n이면 시간 복잡도는 $O(n^3)$이 된다.

그리고 알고리즘 M1을 살펴보면 이 알고리즘은 행렬의 열벡타를 가지고 각 열벡타를 차례로 AND연산을 하므로 시간 복잡도는 $O(n^3)$이 된다.

따라서 알고리즘 Gr1의 시간 복잡도는 알고리즘 1, M1의 $O(n^3)$인데 비해서 $O(n^2)$으로 감소함을 보였다.

공간 복잡도에서 알고리즘 Gr1은

$Test(Q) = n^2,$
$Mset(S) = n^2,$
$Conf(S) = 1/2n(n-1)$
$Flag(S) = n$

이므로 합은 $5/2n^2+1/2n$ 이다.

효율적인 분배가 되려면

$Tset(Q)$는 n^2에서 $1/3n^2$,
$Mset(S)$는 n^2에서 $1/3n^2$,
$Conf(S)$는 $1/2(n^2-n)$에서 $1/3n^2$

정도가 되므로 거의 n^2에 가까이 간다.

알고리즘 1에서 보면 행렬 M, M^T, R을 합하면 공간 복잡도는 $3n^2$이 된다. 알고리즘 M1에서는 행렬 M만을 사용하므로 공간 복잡도는 n^2이 된다.

따라서 알고리즘 Gr1의 공간 복잡도는 알고리즘 1보다는 좋으나 알고리즘 M1보다는 나쁘다. 그러나

효율적인 분배에 있어서는 거의 알고리즘 M1과 같아진다.

5. 결 론

각 이 필요로 하는 데이터베이스 세그먼트들이 평행검출이 가능하도록 분배가 되면 쿼리에 대한 응답시간을 줄일 수 있다. 이러한 문제를 해결하기 위하여 그래프이론을 이용하여 만든 데이터베이스 세그먼트 분배알고리즘 Gr1의 시간 복잡도가 행렬을 이용하여 만든 알고리즘 1과 M1의 시간 복잡도 $O(n^3)$에서 $O(n^2)$으로 감소함을 보였고, 또 공간 복잡도에서는 알고리즘 Gr1은 알고리즘 1보다는 효율이 좋으나 알고리즘 M1보다는 뒤떨어진다. 그러나 효율적인 분배에서는 알고리즘 M1에 가까이 감을 보였다.

REFERENCE

[1] P.A. Berstein, V. Hadzilacos, and N. Goodman, *Concurrency Control and Recovery in Data-Base Systems,* Addison-Wesley, Boston, 1987.

[2] P.A. Berstein and E. Newcomer, *Principles of Transaction Processing,* Morgan Kaufmann, San Francisco, 1996.

[3] T.W. Ling, M.L. Lee, and G. Dobbie, *Semistructured Database Design*, Springer, Berlin, 2005.

[4] D.B. Little, S, Farmer, and Q. El-Hilali, *Digital Data Integrity : The Evolution from Passive Protection to Active Management,* John Wiley and Sons ltd, Symantec Corporation, Wiley, 2007.

[5] D. Dittrich and B. Seeger, "Data Redundancy and Duplicate Detection in Spatial Join Processing," *Preceeding of International Conference on Data Engineering,* pp. 535, 2000.

[6] K. Sikic and O. Carugo, "Protein Sequence Redundancy Reduction: Comparison of Various Method," *Bioinformation,* Vol. 5, No. 6, pp. 234-239, 2010.

[7] C. Hazirbas, J. Diebold, and M. Cremers, "Optimizing the Relevance-Redundancy Tradeoff for Efficient Semantic Segmentation," *Proceeding of International Conference on Scale Space and Variational Methods in Computer Vision,* pp. 243-255, 2015.

[8] A.S. Oh and O.H. Kwon, "Database Construction for Design of the Components Software by Using an Incremental Update Propagation," *Journal of Korea Multimedia Society*, Vol. 6, No. 4, pp. 583-593, 2003.

[9] S. Ghosh, "Distribution a Database with Logical Associations on a Computer Network for Parallel Searching," *IEEE Transactions on Software Engineering,* Vol. SE-2, No. 2, pp. 106-113, 1976.

[10] B. Srinivasan and R. Sankar, "Algorithms to Distribute a Database for Parallel Searching," *IEEE Transactions on Software Engineering,* Vol. 7, No. 1, pp. 112, 1981.

[11] N. Deo, *Graph Theory with Applications to Engineering and Computer Science,* Prentice Hall, New Jersey, 1974.

[12] J.L. Gros and J .Yellen, *Graph Theory and Its Application*, CRC Press, Boca Raton, 2005.

김 중 수

1982년 2월 경북대학교 전자공학과 졸업(학사)
1984년 2월 경북대학교 전자공학과 전산전공(석사)
1996년 6월 경북대학교 전자공학과 전산전공(박사)

1987년 3월~현재 안동대학교 컴퓨터공학과 교수
관심분야 : 데이터베이스, 영상처리, 빅데이타