적응적 다단계 거리 조인의 최적화 기법
(Optimization Methods of Adaptive Multi-Stage Distance Joins)

신 효 섭 † 문 봉 기 †† 이 석 호 †††
(Hyoseop Shin) (Boongki Moon) (Sukho Lee)

요 약 거리조인은 두 공간 데이타 집합 사이의 데이타쌍을 거리 상 가까운 순으로 검색하는 공간조인이다. 본 논문에서는 [1]에서 제시한 적응식 다단계 거리 조인 기법을 최적화하기 위한 기법들을 제안한다. 첫째, 평면 스위핑에서 스위핑 축 선택을 위해 사용되는 스위핑 인덱스 공식을 최적화한다. 둘째, 노드쌍을 관리하는데 사용하는 메인큐의 성능 향상을 위하여 노드쌍의 최대 거리값을 큐의 2차 우선 순위로 적용하는 기법을 제안한다. 또한, 균등 분포 및 비균등 분포 가정하의 한계 거리값 예측 기법의 장단점을 비교한다. 실험 결과는 제안하는 기법들을 통하여 알고리즘의 성능이 CPU 비용과 I/O 비용 면에서 크게 향상되었음을 보여준다.

Abstract The distance join is a spatial join which finds data pairs in the order of distance when associating two spatial data sets. This paper proposes several methods to optimize the adaptive multi-stage distance join, presented in [1]. First, we optimize the sweeping index formula which is used for selecting sweeping axis during plane sweeping. Second, to improve the performance of a priority queue used for maintaining node pairs, we propose to use the maximum distance of a node pair as the second priority of the queue. Moreover, we compare trade-offs in estimating the cut-off distance between under uniformity assumption of data distribution and non-uniformity assumption. The experiments show that the proposed methods greatly improve the performance of the algorithm in CPU cost as well as in I/O cost.

1. 서 론

거리조인(Distance Join)은 2차원 이상의 다차원 공간상의 두 데이타 집합에 대하여 공간적인 거리에 기반하여 거리가 가까운 순으로 데이타 쌍을 검색하는 연산이다[1,2,3]. 거리조인의 응용은 공간 데이타베이스, 이미지 데이타베이스, 항공기 운항 관제 시스템, CAD/CAM 등 다차원 표현 데이타를 이용한 여러 분야에서 찾아볼 수 있다. 공간 데이타베이스 시스템에서는 "서울 시내 안에서 가장 가까이 위치한 호텔과 레스토랑을 검색하라"와 같이 공간적으로 가까이 위치한 공간 객체쌍을 검색하는 응용이 있다. 이미지 검색 시스템에서는 서로 다른 두 이미지 데이타 집합을 대상으로 가장 유사한 이미지 쌍을 찾는 응용에 유용하다.

본 논문에서는 [1]에서 제안한 적응적 다단계 거리 조인 알고리즘 AM-KDJ를 최적화하는 방안을 제안하고 실험을 통하여 최적화된 AM-KDJ 알고리즘을 기존 알고리즘과 비교하였다. AM-KDJ 최적화 방안은 첫째, 평면 스위핑에서 스위핑 축 선택을 위해 사용되는 스위핑 인덱스 공식을 최적화하였다. 둘째, 노드쌍을 관리하는데 사용하는 메인큐의 성능 향상을 위하여 노드쌍의 최대 거리값을 큐의 2차 우선 순위로 적용하는 기법을 제안하였다. 또한 실험을 통해서, 균등 분포 및 비균등 분포 가정하의 한계 거리값 예측 기법의 장단점을 비교하였다.

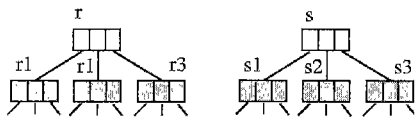
본 논문의 구성은 다음과 같다. 2절에서는 거리조인의 배경 지식을 설명하고, 3절에서는 AM-KDJ 알고리즘을 기술하였고 4절에서는 AM-KDJ의 최적화 방안들을

† 비 회 원 : 서울대학교 컴퓨터공학부
hsshin@db.snu.ac.kr
†† 비 회 원 : 미국 아리조나대학교 전산학과 교수
bkmoon@cs.arizona.edu
††† 종신회원 : 서울대학교 컴퓨터공학부 교수
shlee@cse.snu.ac.kr
논문접수 : 2000년 12월 18일
심사완료 : 2001년 6월 26일

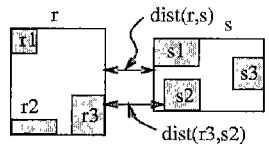
제시하였다. 5절에서는 제안한 기법에 대한 성능평가를 하였으며, 마지막으로 6절에서는 결론을 맺는다.

2. 공간 인덱스를 통한 거리조인

공간 데이타베이스를 구축하는 데 있어서 공간 데이타 집합을 효율적으로 운용하고 접근하기 위하여 다차원 트리 구조의 공간인덱스[4,5,6,7,8,9]를 이용하는 것은 보편화되어 있다. (그림 1)과 같은 트리 구조의 공간 인덱스들은 트리의 계층에 따라 공간 역시 계층적으로 분포한다는 특성을 가진다. 즉, 트리에서 상위 노드는 자기의 모든 하위 노드를 공간적으로 포함하고 있다. (그림 1)처럼 비단말 노드인 r과 s가 있다고 하자. 이때 r과 s사이의 최소거리는 r과 s의 자식노드들끼리의 최소거리보다 항상 작거나 같으며, 마찬가지로, r과 s사이의 최대거리는 r과 s의 자식노드들끼리의 최대거리보다 항상 크거나 같다. 이러한 특징에서 성질 1을 이끌어낼 수 있다.



(a) Tree-Structrured Spatial Index



(b) spatial Containment

그림 1 공간 인덱스의 공간 계층성

성질 1. 트리구조의 공간 인덱스 R과 S에 대해서 루트가 아닌 r∈R, s∈S 두 노드에 대하여, 다음 부등식들이 성립한다.

$$dist(r,s) \geq dist(parent(r),parent(s)),$$
$$dist(r,s) \geq dist(r,parent(s)), \quad (1)$$
$$dist(r,s) \geq dist(parent(r),s).$$

단, $dist(r,s)$는 r과 s의 *MBR* 간의 최소 거리

증명. 공간 인덱스의 공간 계층적인 특성으로부터 자명하다.

성질 1로부터 거리조인을 수행할 때 공간 인덱스를 하향식(top-down) 방식으로 순회하면 탐색 공간을 크게 줄일 수 있다. 예를 들어 노드쌍 <r,s>의 거리가 한계 거리값보다 큰 값이면, r이나 s의 자식노드 들로 이루어진 노드쌍에 대한 순회는 더 이상 필요하지 않게 된다. 성질 1은 R트리와 같은 공간 인덱스들을 이용한 거리조인 수행시 탐색 공간 축소에 있어서 중요한 역할을 한다.

2.1 점진적 거리조인과 K-거리조인

트리 인덱스를 하향식으로 순회하는 동안에 검사한 노드쌍을 우선순위 큐에 저장하는 것이 바람직하다. 큐의 우선 순위는 노드쌍 안에서 두 노드 간의 **거리**(distance)이다. 이렇게 함으로써 두 노드 간의 거리가 작은 쌍부터 순회하게 한다. 이 우선 순위 큐를 **메인큐**(main queue)라고 정의한다. 메인큐는 양쪽 트리의 루트 노드쌍으로 초기화된다. 메인큐로부터 비객체 노드쌍이 검출되면 노드쌍의 한쪽 노드의 자식노드들이 나머지 다른 쪽 노드의 자식노드들과 짝을 지어서 메인 큐에 재삽입된다. 이 **노드 확장**(node expansion) 과정은 메인 큐가 비거나, 또는 질의 결과가 모두 도출되면 종료된다. 노드 확장 방식은 한 쪽은 자식 노드, 다른 한 쪽은 그대로 부모 노드로 짝을 짓는 한쪽 노드 확장(uni-directional node expansion) 방식과 양 쪽 모두 자식 노드들로 짝을 짓는 양쪽 노드 확장(bi-directional node expansion) 방식으로 구분할 수 있다. 만약 메인 큐로부터 검출된 노드쌍이 두 개의 객체로 구성된 객체 노드쌍이라면, 그 노드쌍은 질의 결과로서 즉시 반환된다. 이러한 방식으로 사용자가 원하는 개수만큼의 <object,object> 노드쌍을 점진적으로 생성하는 알고리즘을 **점진적 거리조인**(IDJ, Incremental Distance Join)으로 정의한다. (그림 2)는 점진적 거리 조인의 기본 구조를 나타낸 것이다.

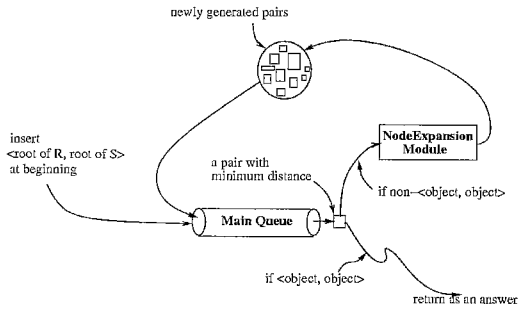


그림 2 점진적 거리조인의 기본 구조

거리조인을 수행하기 이전에 검색할 객체 노드쌍의 개수 K가 미리 정해져 있다면 이를 이용한 알고리즘의 성능 향상을 기대할 수 있다. 즉, K를 미리 알고 있으

면 점진적 알고리즘보다도 불필요하게 발생하는 노드쌍들을 더 많이 가지치기(pruning) 할 수 있다. 질의 수행 단계에서 현재까지의 객체 노드쌍의 거리 중 가장 작은 K개를 유지하고, 그 중 가장 큰 거리값을 한계값 **qDmax**라고 정하고, 차후 생성되는 노드쌍 중 거리가 이 값보다 더 큰 값을 가지는 것들은 메인큐에 삽입하지 않고 제거할 수 있다. K개의 객체쌍들에 대한 거리를 유지하기 위하여 메인큐와는 별도의 **거리큐**(distance queue)를 둔다. 거리큐는 K길이의 최대히프(max heap)로 구성할 수 있다. qDmax는 거리큐의 헤더값을 나타내는데 알고리즘 초기에는 무한대값으로 지정되며, 알고리즘이 수행됨에 따라 점점 더 작아지며, K에 대한 실제 한계거리값 **Dmax** 보다는 항상 크거나 같게 된다. 하지만 Dmax값은 알고리즘이 끝난 이후에나 알 수 있는 값이다. 거리큐를 이용한 거리조인을 **K 거리조인 (KDJ, K Distance Join)**라고 정의한다. (그림 3)은 K 거리조인의 기본 구조를 표현한 것이다.

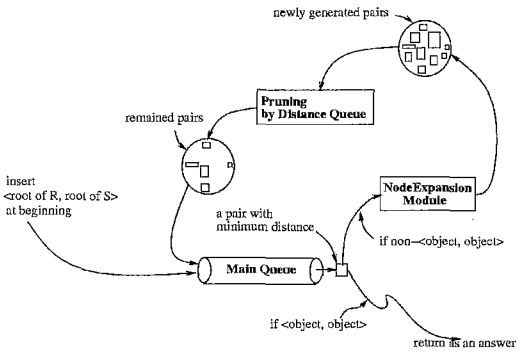


그림 3 K 거리조인의 기본 구조

2.2 관련 연구

[3] 논문에서는 한쪽 노드 확장 방식의 점진적 거리조인과 K 거리조인을 제안하였다. 한쪽 노드만을 처리하는 방식은 양쪽 노드만을 처리하는 방식보다 한번에 생성하는 노드쌍의 개수가 줄어든다는 장점이 있다. 하지만 이 방식은 공간인덱스의 각 노드를 필요 이상으로 접근하는 오버헤드가 발생하므로 양쪽 노드 확장 방식보다 R 트리 등의 공간 인덱스 디스크 노드 접근횟수가 크게 증가한다. 한쪽 노드 확장 방식의 또 다른 문제점은 노드 확장시 가능한 모든 노드간의 거리 비교가 불가피하다는 점이다. 이 점은 양쪽 노드 확장 방식에서 효율적인 평면 스위핑을 이용한 노드쌍의 생성에 비하여 매우 비효율적이다.

[2]에서는 양쪽 노드 확장 방식을 적용한 K 거리조인 알고리즘 B-KDJ을 제시한 반면, [1]에서는 B-KDJ 알고리즘을 다단계 적응 방식으로 개선한 AM-KDJ 알고리즘을 제안하였다. 본 논문에서는 이 알고리즘들을 최적화하기 위하여 메인큐의 우선순위를 개선하였고, 스위핑 인덱스 공식을 최적화하였으며, 균등 분포 뿐만 아니라 비균등 분포 가정하의 한계 거리값 예측 기법이 AM-KDJ의 성능에 미치는 영향력을 실험에 추가하였다.

포함(inclusion), 교차(intersection) 등의 프레디키트를 사용한 기존의 공간 조인 기법들[10,11,12]을 거리조인 질의의 처리에 사용하려면 우선 한계 거리값 Dmax를 예측하고 결과로 도출되는 노드쌍을 거리순으로 정렬하면 된다. 다만 Dmax값은 정확하게 예측한다는 것은 현실적으로 불가능하며, 너무 크거나 작게 예측할 우려가 있기 때문에 안정된 성능을 보장할 수 없다.

3. 적응적 다단계 거리조인

B-KDJ[2] 알고리즘에서는 qDmax값이 초기에는 무한대값으로 설정되고 알고리즘이 진행됨에 따라 점차적으로 줄어든다. 이 qDmax값은 생성된 노드쌍을 메인큐에 삽입할 지의 여부를 판단하는 한계값이므로 B-KDJ 알고리즘의 성능에 중대한 영향을 미치는데, 만약에 qDmax값이 실제 한계값인 Dmax값에 접근하는 속도가 매우 느리다면, 알고리즘 초기 단계에서 필요치 않은 노드쌍들을 메인큐에 삽입하는데 많은 비용을 들이게 된다. 결과적으로는 알고리즘이 종료될 시점에서는 많은 수의 노드쌍들이 불필요하게 메인큐 안에 남아있게 되는 결과를 초래한다. 이러한 **초기 지연**(slow start)은 K값이 큰 경우에 더 심각하다. 실험한 바에 따르면 K가 100,000일 경우에, 질의 결과의 1%, 즉 1,000 개의 객체노드쌍을 도출하는 동안에 경과한 시간이 전체 수행 시간의 95%이상을 차지하였다. AM-KDJ는 B-KDJ의 초기지연 문제를 완화하기 위하여 공격적 가지치기(aggressive pruning)와 보완(compensation) 기법을 적용한 알고리즘이다.

초기지연 문제는 알고리즘이 동적으로 갱신되는 qDmax 값에 의지한 가지치기 전략에 기인한다. 이 문제를 해결하기 위하여 AM-KDJ 알고리즘은 주어진 K에 대하여 예상 한계값 **eDmax**을 새로운 가지치기 수단으로 사용한다. eDmax는 초기에 예상값으로 설정되고 알고리즘이 수행하는 도중 적응 방식으로 보정할 수 있다. AM-KDJ 알고리즘은 공격적 가지치기(aggressive pruning) 단계와 보완(compensation) 단계로 구성된다.

공격적 가지치기 단계에서는,

```
1  set AnswerSet ← an empty set;
2  set QM, QD, Qc ← empty main, distance and compensation queues;
3  set eDmax ← an initial estimated Dmax;
4  insert a pair <R.root, S.root> into the main queue QM;
5  while |AnswerSet| < k and QM ≠ 0 do
6      set c ← dequeue(QM);
7      if c is an <object, object> then AnswerSet ← {c} ∪ AnswerSet;
8      else
9              if qDmax ≤ eDmax then eDmax ← qDmax;
10             if c.distance < eDmax then
11                     reinsert c back into QM;
12                     break;
13             end
14             AggressivePlaneSweep(c);
15             enqueue(Qc, c);
16 end
17 if |AnswerSet| < k then execute Algorithm 3;
18
19 procedure AggressivePlaneSweep(<l,r>)
20 set L ← sort_axis({child nodes of l});   // Sort the child nodes of l by axis values.
21 set R ← sort_axis({child nodes of r});   // Sort the child nodes of r by axis values.
22 while L ≠ 0 and R ≠ 0 do
23     n ← a node with the min axis value ∈ L ∪ R;
24     if n ∈ L then
25             L ← L - {n}; AggressiveSweepPruning(n, R);
26             n.compensate ← a node in R with the min axis value and not paired with n;
27     else
28             R ← R - {n}; AggressiveSweepPruning(n, L);
29             n.compensate ← a node in L with the min axis value and not paired with n;
30     end
31 end
32
33 procedure AggresiveSweepPruning(n, List)
34 Same as the SweepPruning procedure in Algorithm 1 except line 24 replaced with the
35 following :
36     if  axis_distance(n,m) > eDmax then return;
```

알고리즘 1 AM-KDJ : 공격적 가지치기 단계

- eDmax 값은 공격적 가지치기를 위한 값으로서 메인큐와 거리에 대한 삽입 횟수를 제한하기 위하여 노드쌍의 축간거리(axis distance) 값에 기반한 가지치기에 사용되며,
- qDmax 값은 B-KDJ에서와 마찬가리로 축간 거리가 eDmax 값보다 작은 노드쌍에 대하여 실제 거리(real distance) 값에 기반한 가지치기에 사용된다.

결과적으로 알맞게 예측된 eDmax을 값을 통하여, AM-KDJ 알고리즘은 초기 단계에서도 많은 수의 노드쌍을 제거할 수 있으며 초기 지연으로 인한 알고리즘의 성능 저하를 늦출 수 있다. 하지만 eDmax값이 실제 한계값 Dmax보다 더 작은 값으로 예측되었을 때는 도출되어야할 객체쌍이 누락(false dismissal)될 수 있다. 이 문제를 방지하기 위하여 메인큐에서 도출된 모든 비객체 노드쌍은 보완큐(compensation queue)에 재저장된다. 축간 거리가 eDmax 값보다 작은 노드쌍들에 대하여 eDmax값이 아닌 qDmax값으로 실제 거리 가지치기를 하는 이유는 보완 단계에서의 효율성을 위해서이다. 만약에 실제 거리 가지치기에도 eDmax을 적용한다면 보완 단계에서 모든 가능한 노드쌍들을 재추적해야하기 때문에 많은 비용을 부담해야 할 것이다. 또한 qDmax를 사용함으로써 AM-KDJ의 성능이 예측 한계값 eDmax에 덜 민감해지는 요인이 되기도 한다. (알고리즘 1)은 공격적 가지치기 단계의 AM-KDJ 알고리즘을 나타낸다.

예를 들어 (그림 4)에서, 노드 r1에 대해서 s1과 s2는 실제거리 비교가 필요하지만, eDmax보다 축간 거리가 큰 s3와 s4는 가지치기가 된다. 따라서 B-KDJ는 네 개의 노드 쌍 {<r1, s1>, <r1, s2>, <r1, s3>, <r1, s4>}을 메인큐에 삽입해야 했던 반면, AM-KDJ의 공격적

가지치기 단계에서는 두 개의 노드쌍 {<r1, s1>, <r1, s2>} 만을 삽입하면 된다.

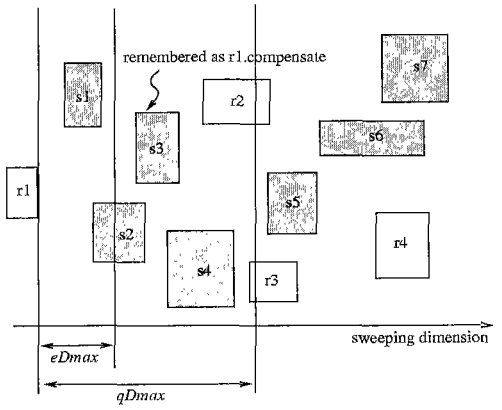


그림 5 AM-KDJ에서 공격적 가지치기

공격적 가지치기 단계는 다음 조건 중 하나가 만족하면 수행이 완료된다: (1) 메인큐가 비었거나(5번째 줄), (2) K혹은 그 이상의 질의 결과가 도출되었거나(5번째 줄), 3) 메인큐로부터 검출된 노드쌍의 거리가 eDmax보다 작거나 일 때이다(10번째 줄). 조건 (2)가 만족되었다면 보완 단계의 수행이 불필요하며 알고리즘은 종료된다. 보완 단계의 수행이 불필요해지는 경우 즉, eDmax값이 실제 한계값 Dmax보다 커지는 경우는 eDmax값이 qDmax보다 커졌을 때도 감지된다. 이 때는 AM-KDJ는 B-KDJ와 동일한 방식으로 동작한다. 조건 (1) 혹은 (3)이 만족되었다면 eDmax값은 저평가된 상태이며 보완큐에 있는 모든 노드는 메인큐에 재삽입되며 보완 단계가 시작된다. 보완 단계에 대한 상세한 내용은 [1]에 기술되어 있다.

4. AM-KDJ 알고리즘의 최적화

본 절에서는 AM-KDJ을 위한 최적화된 기법들을 제안한다.

4.1 평면 스위핑 인덱스의 최적화

B-KDJ[2] 및 AM-KDJ[1]에서는 평면 스위핑 가지치기 시, 노드쌍의 두 노드 간의 위치 관계를 고려하여 스위핑하는 기준축을 선택함으로써 노드간의 거리 계산 횟수와 생성되는 노드쌍의 개수를 줄여주어 알고리즘의 성능을 개선할 수 있다는 점과, 그 방안으로서 스위핑 인덱스 기법을 제시하였다. 본 절에서는 [1]에서 제안한 분리 관계에 대한 스위핑 인덱스 공식을 확장하여 교차, 분리, 포함 관계를 모두 고려한 최적화된 기법을 제안한다.

스위핑 인덱스 값은 확장할 노드 쌍이 주어졌을 때 각 축에 대해서 계산되는 값으로서, 개념적으로 해당 축에 대한 실제 거리(real distance)를 계산하는 횟수의 상대적인 값을 나타낸다. 따라서 각 축에 대하여 스위핑 인덱스 값을 계산하여 가장 큰 값을 가지는 축으로 평면 스위핑을 수행한다. 노드쌍 <r,s>에 대하여 축 x에 대한 스위핑 인덱스는 다음과 같이 표현된다.

$$Sweeping\ Index_x = \int_0^{|r|_x} \frac{Overlap(qDmax, r, t)}{|s|_x}\,dt + \int_0^{|s|_x} \frac{Overlap(qDmax, s, t)}{|r|_x}\,dt \quad (2)$$

위의 공식에서 $|r|_x$와 $|s|_x$는 각각 r과 s의 x축 방향 너비를 나타낸다. 공식의 첫 번째 적분식에서 Overlap (qDmax, r, t)는 x축 상에서 윈도우 [t, t + qDmax]와 겹치는 s의 x 축 방향 길이를 나타낸다. 따라서 Overlap(qDmax, r, t)는 r의 자식 노드 중 x 축 상에서 t 지점에 있는 노드와 실제 거리를 계산해야 하는 s의 자식 노드들이 포함되어 있는 구간의 x 축 길이를 나타낸다. 이 값을 $|s|_x$으로 나누면 s의 전체 자식노드들 중 t 지점에 있는 r의 자식노드와 실제 거리를 계산해야 하는 s 자식 노드의 개수 비율을 구할 수 있다. t가 0에서 $|r|_x$ 까지 변할 때 이 값들을 적분하면 x축에 대한 실제 거리 계산 비율값이 나온다. 마찬가지로, 두 번째 적분식은 s에 대한 r 자식노드들의 실제 거리 계산 비율을 나타낸다.

예를 들어, r 과 s 노드가 x축에 대하여 겹쳐 있지 않고 노드간 거리가 β 일때 (그림 5)의 왼쪽 그림과 같다고 하자.

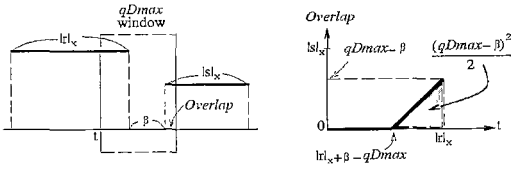


그림 6 스위핑 인덱스 계산 예

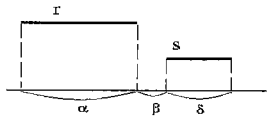
이 예에서는 r이 s의 왼쪽에 위치하기 때문에 공식 (2) 두 번째 적분식 항목은 항상 0이 된다. 한편 r에 대해서는 t가 [0, $|r|_x$]사이에서 변화할 때 주어진 qDmax 윈도우 크기에 대하여 Overlap 값을 계산한 그래프가 (그림 6)의 오른쪽과 같다. 그래프에서 t가 [0, $|r|_x+\beta-qDmax$] 사이에 있을 때는 qDmax 윈도우와

s가 겹치는 구간이 없기 때문에 Overlap 값이 0이 된다는 것을 알수 있다. 빗금친 부분의 면적을 $|s|_x$ 로 나눈 값이 공식 (2)의 왼쪽 적분 값이며, 이 값이 곧 x축의 스위핑 인덱스 값이 된다.

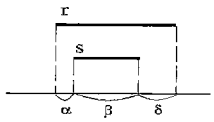
각 축에 대한 스위핑 인덱스 값이 모두 계산되면, 그 중 가장 작은 값을 가지는 축을 해당 노드쌍에 스위핑 축으로 선택한다. 여기서 한가지 주목할 점은 스위핑 인덱스가 적분으로 표현되었다고 하지만, 이 적분식은 결국 간단한 산술식으로 표현될 뿐 아니라 각 부모 노드쌍에 대하여 단 한 번 계산되므로, 수 천개의 자식 노드쌍을 생성하는 비용과 비교했을 때, 알고리즘의 많은 성능 향상을 기대할 수 있다.

적분식으로 표현된 스위핑 인덱스를 최적화된 공식으로 풀어내기 위해서 노드 간의 위치 관계를 (그림 6)과 같이 분리(separated), 포함(contained), 교차(intersected)의 3가지로 구분한다.

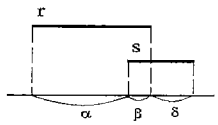
α,β,δ가 그림 6에서와 같이 두 노드의 위치관계에 의해서 정해질 때, 최적화된 스위핑 인덱스 공식은 (표 1)과 (표 2)로 정리된다.



(a) separated



(b) Contalned



(c) Interseceted

그림 6 노드쌍 <r,s>에 대한 축 방향 위치 관계

4.2 메인 큐 우선 순위의 최적화

(그림 2)와 (그림 3)에서 볼 수 있듯이, 노드 확장에

표 1 공식 2의 첫 번째 적분식 항목

r 과 s	스위핑 인덱스 공식	조건
분리	0	if $qDmax < \beta$
	$\frac{(qDmax-\beta)^2}{2\|s\|_x}$	if $\beta \le qDmax < \beta+\min\{\|r\|_x, \|s\|_x\}$
	$\frac{2\|r\|_x(qDmax-\beta)-\|r\|_x^2}{2\|s\|_x}$	if $\|r\|_x+\beta \le qDmax < \|s\|_x+\beta$
	$qDmax-\beta-\frac{\|s\|_x}{2}$	if $\|s\|_x+\beta \le qDmax < \|r\|_x+\beta$
	$\|r\|_x-\frac{(\max\{\|r\|_x+\|s\|_x+\beta-qDmax, 0\})^2}{2\|s\|_x}$	if $\min\{\|r\|_x, \|s\|_x\}+\beta \le qDmax$
포함	$qDmax$	if $qDmax < \alpha$
	$qDmax-\frac{(qDmax-\alpha)^2}{2\|s\|_x}$	if $\delta \le qDmax < \|s\|_x+\alpha$
	$\alpha+\frac{\|s\|_x}{2}$	if $\|s\|_x+\alpha \le qDmax$
교차	$qDmax+\frac{qDmax(qDmax-2\delta)}{2\|s\|_x}$	if $qDmax < \min\{\alpha, \beta\}$
	$\frac{2\|r\|_x qDmax-\alpha^2}{2\|s\|_x}$	if $\alpha \le qDmax < \delta$
	$qDmax-\frac{\delta^2+(\max\{qDmax-\alpha, 0\})^2}{2\|s\|_x}$	if $\delta \le qDmax < \|s\|_x+\alpha$
	$\alpha+\frac{\|s\|_x^2-\delta^2}{2\|s\|_x}$	if $\|s\|_x+\alpha \le qDmax$

표 2 공식 2의 두 번째 적분식 항목

r 과 s	스위핑 인덱스 공식	조건
분리	0	always
포함	$\frac{qDmax\|s\|_x}{\|r\|_x}$	if $qDmax < \alpha$
	$\frac{2qDmax\|s\|_x-(qDmax-\delta)^2}{2\|r\|_x}$	if $\delta \le qDmax < \|s\|_x+\delta$
	$\frac{\|s\|_x(\|s\|_x+2\delta)}{2\|r\|_x}$	if $\|s\|_x+\delta \le qDmax$
교차	$qDmax-\frac{qDmax(2\alpha-qDmax)}{2\|r\|_x}$	if $qDmax < \|r\|_x-\alpha$
	$\frac{(\|r\|_x-\alpha)^2}{2\|r\|_x}$	if $qDmax \ge \|r\|_x-\alpha$

의해서 생성된 노드쌍들은 거리 우선 순으로 메인큐에 저장된다. 그런데 여기서 한가지 주목할 점은 거리가 같은 동점자 노드쌍의 처리문제이다. 특히 중간 노드들끼리에서 생성되는 노드쌍에서는 영역이 교차되는 경우가 많은데 이 때는 거리가 0으로 처리된다. 따라서 동점자 노드쌍에 대한 메인큐의 우선 순위 부여는 이 후 노드 확장에 의해서 생성될 노드쌍의 개수에 많은 영향을 주며 결국 알고리즘의 성능에 상당한 차이를 보일 수 있다. [3]에서는 동점자 노드쌍에 대해서 우선 순위를 트리 인덱스 안에서의 노드의 레벨에 두었다. 레벨이 더 큰 노드가 있는 노드쌍에 더 나은 우선 순위를 매기는 방법이다. 이 방법은 노드쌍이 리프노드들로 이루어진 객체 노드쌍에 근접해 있는 노드쌍들을 먼저 순회하기 위한 휴리스틱에 기반하고 있다.

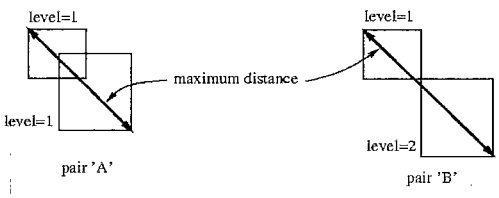


그림 7 동점자 노드쌍의 우선 순위 매기기

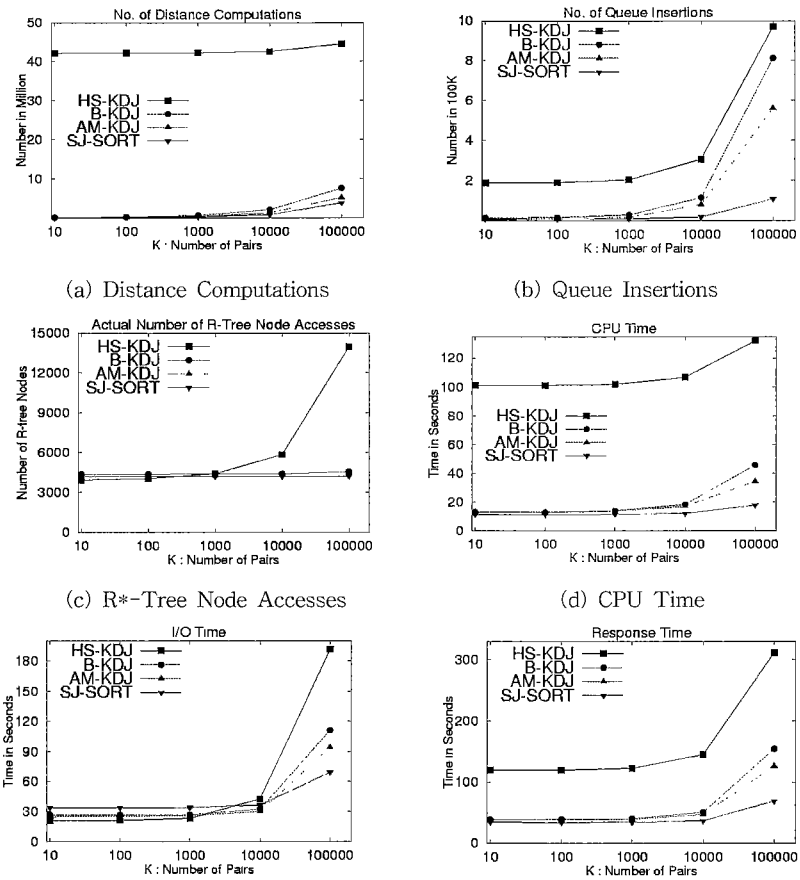
하지만 노드의 깊이에 기반한 이 방법이 거리큐에 의한 qDmax값의 감소를 항상 가속화하지는 않기 때문에 이로운 것만은 아니다. 예를 들어 (그림 7)에서 왼쪽 노드 쌍 A와 오른 쪽 노드쌍 B의 거리가 모두 0인 경우를 고려해 보자. A 노드쌍은 레벨이 1인 노드만 있고, B 노드쌍은 레벨이 2인 노드가 있기 때문에, 이 방법에 따르면 노드쌍 B가 A 보다 더 나은 우선순위를 가진다. 그렇지만 이 상황에서는 노드쌍 A를 먼저 순회하는 것이 이롭다. 왜냐하면 노드쌍 A에는 노드쌍 B보다 더 작은 값의 거리를 가지는 자식노드쌍들이 더 많기 때문이다. 본 논문에서는 이러한 문제를 해결하기 위하여 메인큐의 1차 우선순위인 거리값이 같은 노드쌍들에 대하여 2차 우선순위를 노드쌍의 최대 거리(maximum distance)로 할 것을 제안한다. 그렇게 함으로써 더 작은 값의 거리를 가지는 노드쌍들이 알고리즘의 초기 단계에 더 많이 생성되게 되고, 결과적으로 qDmax의 감소를 가속화시킬 수 있기 때문이다. qDmax의 더 빠른 감소는 노드쌍의 메인큐 삽입 비용을 감소시켜 준다.

5. 성능 평가

실험에 사용한 컴퓨터는 Solaris 2.7이 탑재된 Sun UltraSparc-II, 메인메모리 256M, Ultra 10 EIDE 인터페이스로 연결된 9G 하드디스크의 규격이다. 운영체제의 캐시 효과를 없애기 위해서 Solaris의 Direct I/O 기능을 사용하였다. 실험에 사용한 데이타는 미국 Bureau of Census에서 제공하는 Tiger/Line 97 지리정보 데이타 중 미국 아리조나 주의 도로 및 기타 공간 지역 객체를 나타내는 각각 633,461개와 189,642개의 실제 직선 데이타들이며, 이를 R* 트리[5]로 구성하였다.

5.1 K 거리조인의 성능 비교

최적화된 AM-KDJ 알고리즘의 성능을, HS-KDJ[3], B-KDJ[1,2] 알고리즘의 성능에 대해서 비교하였는데, 비교 항목은 거리 계산 횟수, 메인 큐 삽입횟수, R* 트리 노드 접근 횟수, CPU 시간, 디스크 I/O 시간, 전체 수행 시간 등이다. K에 대한 실제 거리 한계값인 Dmax가 알고리즘 수행 전에 알려져 있다고 가정했을 때의 within 프레디키트를 사용한 공간 조인[11]과 이



(a) Distance Computations (b) Queue Insertions

(c) R*-Tree Node Accesses (d) CPU Time

(e) I/O Time (f) Response Time

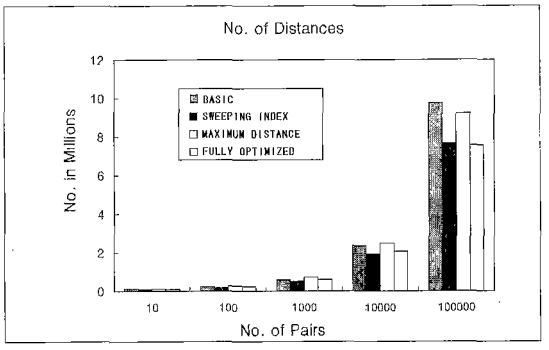
그림 8 K 거리조인의 성능 비교

의 결과를 정렬하는 방식의 수행방법인 가상적 SJ-SORT도 비교 대상에 포함하였다. 실험에서 사용한 R* 트리의 버퍼 크기는 512Kbyte로 제한하였으며, K값은 10에서 100,000 사이로 하였다.

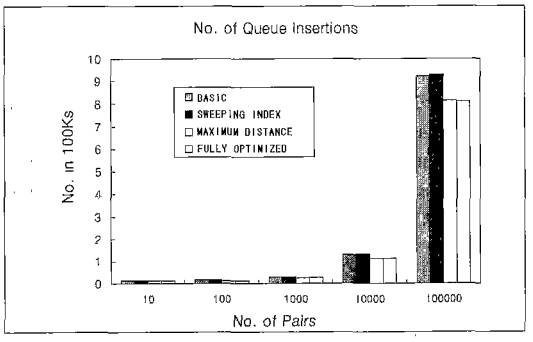
먼저 노드쌍의 거리 계산 횟수는 평면 스위핑을 이용한 양쪽 노드 확장 방식인 B-KDJ와 AM-KDJ 가 한쪽 노드 확장 방식의 HS-KDJ에 비해서 K 값에 관계없이 월등히 적은 수치를 나타내었다((그림 9)-(a)). 메인큐 삽입 횟수 역시 비슷한 양상을 보였지만 K값이 커짐에 따라 B-KDJ에 비해서 AM-KDJ 알고리즘은 점점 더 좋은 수치를 나타내었다((그림 9)-(b)). 이는 AM-KDJ가 eDmax를 이용한 공격적 가지치기를 통하여 B-KDJ의 초기 지연 문제를 극복하고 있음을 보여준다. R* 트리의 노드 접근 횟수 측면에서도 K값이 10 ~ 1,000 사이의 비교적 작은 범위에서는 비슷하지만 K값이 더 커짐에 따라 양쪽 노드 확장 방식인 B-KDJ와 AM-KDJ가 한쪽 노드 확장 방식인 HS-KDJ보다 더 좋은 성능을 나타내었다((그림 9)-(c)). 노드쌍의 거리 계산 횟수는 CPU 시간에 주로 기여하고, 큐삽입 횟수와 R*트리 노드 접근 횟수는 주로 I/O 시간에 기여하는데, (그림 8)의 (d), (e), (f)를 통해서 최적화된 AM-KDJ 알고리즘이 다른 거리조인 알고리즘에 비해서 고루 좋은 성능을 나타냄을 알 수 있다. 한편, (그림 9)-(f)를 보면 최악의 경우에도 AM-KDJ의 반응시간이 이상적인 SJ-SORT기법에 크게 떨어지지 않음(2배를 넘지 않음)을 확인할 수 있다. 이는 Dmax 예측에 대한 시행 착오 방식으로 SJ-SORT를 여러번 수행하는 것보다 AM-KDJ 알고리즘이 안정적인 성능을 보장함을 나타낸 것이다.

5.2 AM-KDJ의 최적화된 성능

본 절에서는 4.1 과 4.2 에서 제시된 AM-KDJ의 최적화 기법들의 효과를 실험하였다. 즉, (1) 최적화된 스위핑 인덱스와 메인큐의 최적화된 2차 우선순위를 동시에 사용했을 때, (2) 최적화된 스위핑 인덱스만을 사용했을 때, (3)최적화된 2차 우선순위 기법만 사용할 때, (4) 2가지 다 사용하지 않을 때의 각각의 AM-KDJ 성능 차이를 거리 계산 비용과 메인 큐 삽입 횟수 측면에서 비교하였다.



(a) Distance Computations



(b) Queue Insertions

그림 9 AM-KDJ의 최적화 효과

(그림 9)의 (a)와 (b)에서 볼 수 있듯이 최적화된 스위핑 인덱스는 최대 35% 정도의 노드쌍의 거리 계산 비용을 줄여주었으며 최적화된 메인큐 2차 우선순위는 최대 56%의 큐 삽입 비용을 줄여주었다. 이들 최적화의 효과는 서로 거의 독립적인 것으로 나타났다. 즉, 이 실험 결과는 최적화된 스위핑 인덱스 기법은 노드쌍의 거리 계산 비용을 줄여주는 데 기여하고, 노드쌍의 최대 거리를 큐의 2차 우선 순위로 사용하는 기법은 큐의 노드쌍 삽입 비용을 줄여주는 데 기여한다는 점을 그대로 반영하고 있다.

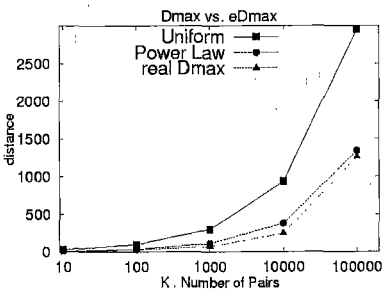
5.3 eDmax 예측 기법 비교

AM-KDJ 알고리즘에서는 주어진 K에 대한 한계 거리값 Dmax를 알고리즘 수행 이전에 예측하여 eDmax로 설정한다. 실제 데이타에 대하여 한계 거리값을 정확히 예측하기는 거의 불가능하기 때문에 [1]에서는 데이타가 공간 상에 균일하게 분포되어 있다는 가정을 하였었다. 또한 그러한 가정에 의해서 예측된 eDmax값이 알고리즘의 성능에 많은 영향을 미치지 않는다는 것을 실험을 통하여 보여주고 있다. 거리 조인의 대상인 두 개의 데이타 집합 R, S에 대하여 데이타의 균등 분포 가정하의 한계 거리 예측값 공식은 다음과 같다.

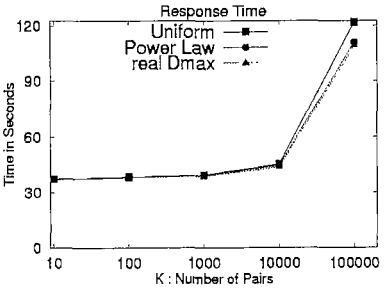
$$eDmax = \sqrt{K \times \rho} \quad (\text{단}, \rho = \frac{area(R \cap S)}{\pi \times |R| \times |S|}) \qquad (3)$$

한편, [13]에서는 파워 규칙(power law)를 이용한 데이타의 균일 분포를 가정하지 않은 공간 데이타에 대한 한계 거리값 예측기법을 제안하고 있다. 그 공식은 다음과 같다.

$$eDmax = L \times \sqrt[S]{\frac{k}{C}} \quad (\text{단, L은 데이타 도메인의 최대 축길이, c와 S는 파워규칙의 상수}) \qquad (4)$$



(a) eDmax



(b) Response Time

그림 10 eDmax 예측값 및 그에 따른 AM-KDJ의 반응 시간

(그림 10)의 (a)를 보면 데이타의 균일 분포 가정에 의한 eDmax 예측은 실제 Dmax 값에 비해서 2~3배 크게 예측되는 경향을 보였으며, 파워 규칙에 의한 eDamx 예측은 실제 Dmax에 많이 근접해 있음을 알 수 있다. 하지만 (그림 10)의 (b)를 보면 두 가지 eDmax 예측값에 의한 AM-KDJ 성능이 Dmax 값에 의한 성능에 비하여 많이 다르지 않음을 나타내고 있다. 이는 AM-KDJ가 예측값 Dmax에 의하여 많은 영향을 받지 않고 안정적인 성능을 보여준다는 것을 뒷받침한다. 또한, 비균등 분포를 가정한 파워 규칙 한계 거리값 예측 기법은 좀 더 정확한 Dmax 값을 예측해 준다는 점에서 장점이 있지만 AM-KDJ알고리즘의 성능에 큰 영향을 주지 못한다. 균일 분포 가정에 의한 eDmax 예측 기법의 장점이라면 파워규칙을 적용하기 위해 필요한 전처리 과정이 없어도 된다는 점이다.

6. 결 론

본 논문에서는 최근에 제안된 공간 조인 기법인 거리 조인 AM-KDJ에 대한 최적화 전략들을 제안하였다. 최적화된 스위핑 인덱스를 제안하여 노드쌍의 거리 계산 비용을 감소시켰으며, 거리가 같은 동점자 노드쌍의 우선 순위를 노드간의 최대 거리 값으로 제안함으로써 메인큐로의 노드쌍 삽입 비용을 감소시켰다. 제안된 최적화 전략들은 이전 알고리즘에 비해서 많은 성능 향상을 나타냄을 보여주었다. 또한 데이타의 균일 분포 가정에 의한 eDmax 예측 기법과 파워 규칙에 의한 예측 기법을 AM-KDJ 성능 측면에서 비교하였으며, AM-KDJ는 eDmax값에 많은 영향을 받지 않음을 실험을 통해 확인하였다.

R-트리를 바탕으로한 거리조인 알고리즘은 2차원 혹은 3차원 등의 저차원 데이타 베이스에서는 효율적인 성능을 발휘하지만, 차원이 증가하면 그 성능의 한계를 드러낸다. 이는 R-트리 자체가 고차원에서는 효율적으로 작동하지 않는다는 데 그 원인이 있다. 고차원에서의 효율적인 거리조인 기법에 대한 연구가 향후 과제로 남는다.

참 고 문 헌

[1] Hyoseop Shin, Bongki Moon, Sukho Lee, "Adaptive Multi-Stage Distance Join Processing," ACM SIGMOD Conference 2000, Dallas, America, May, 14 - 19, 2000

[2] 신효섭, 이석호, "공간 데이타베이스에서 최근접 K쌍을 찾는 효율적 기법", 정보 과학회 논문지: 데이타베이스, 27권 2호, pp.238-246, 2000.

[3] Hjaltason G. R. and Samet H., "Incremental Distance Join Algorithms for Spatial Databases," Proc. of ACM SIGMOD Conf., 1998

[4] Guttman A., "R-Trees: A Dynamic Index Structure for Spatial Searching," Proc. of ACM SIGMOD, 1984.

[5] Beckmann N., Kriegel H. P., Schneider R., Seeger B., "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," Proc. of ACM SIGMOD, 1990.

[6] Sellis T., Roussopoulos N., Faloutsos C., "The R+-Tree: A Dynamic Index for Multi-Dimensional Objects," Proc. of VLDB conf. 1987.

[7] Berchtold S., Keim D., Kriegel H. P., "The X-Tree: An Index Structure for High-Dimensional Data," Proc. of VLDB conf, 1996.

[8] Henrich A., "The LSD^h-tree: An Access Structure for Feature Vectors," Proc. of ICDE, 1998.

[9] Robinson J. T., "The K-D-B-tree: A Search Structure for Large Multidimensional Dynamic Indexes," Proc. of ACM SIGMOD, 1981.

[10] Arge L., Procepiue O., Ramaswamy S., Suel T., Vitter J. S., "Scalable Sweeping-Based Spatial Join," Proc. of VLDG conf., 1998.

[11] Brinkhoff T., Kriegel H. P., Seeger B., "Efficient Processing of Spatial Joins Using R-Trees," Proc. of ACM SIGMOD, 1993.

[12] Patel J. M. and DeWitt D. J., "Partition Based Spatial-Merge Join," Proc. of ACM SIGMOD, 1996.

[13] Christos Faloutsos, Bernhard Seeger, Agma J. M. Traina, Caetano Traina Jr, "Spatial Join Selectivity Using Power Laws," ACM SIGMOD Conference 2000, Dallas, America, May 14 - 19, 2000

신 효 섭

1994년 서울대학교 컴퓨터공학과 학사. 1996년 서울대학교 컴퓨터공학과 석사. 2002년 서울대학교 전기.컴퓨터공학부 박사 졸업 예정. 1999년 ~ 2001년 2차례 미국 아리조나 주립대 전산학과 방문 연구. 2001년 7월 ~ 현재 삼성전자 CTO 전략실 소프트웨어 센타 책임 연구원. 관심분야는 공간 데이타베이스, 고차원 데이타베이스, 임베디드 데이타베이스, XML 등

문 봉 기

1983년 서울대학교 컴퓨터공학과 학사. 1985년 서울대학교 컴퓨터공학과 석사. 1985년 ~ 1990년 삼성전자 및 삼성종합기술원 근무. 1996년 University of Maryland College Park 전산학 박사. 1997년 ~ 현재 University of Arizona 전산학과 조교수 재직. 관심분야는 Spatial/Multidimensional databases, XML, Scalable web server, Moving objects, *Bioinformatics*, *Parallel processing*.

이 석 호

1964년 연세대학교 정치외교학과 졸업. 1975년, 1979년 미국 텍사스대학교 전산학 석사와 박사학위 취득. 1979년 ~ 1982년 한국과학원 전산학과 조교수. 1982년 ~ 1986년 한국정보과학회 논문편집위원장. 1986년 ~ 1988년 한국정보과학회 부회장. 1988년 ~ 1989년 미국 IBM T.J. Watson 연구소 객원교수. 1988년 ~ 1990년 데이타베이스연구회 운영위원장. 1989년 ~ 1991년 서울대학교 중앙교육연구전산원 원장. 1994년 한국정보과학회 회장. 1997년 ~ 현재 한국학술진흥재단 부설 첨단학술정보센터 소장. 1982년 ~ 현재 서울대학교 컴퓨터공학부 교수. 관심분야는 데이타베이스, 멀티미디어 데이타베이스