韓國 OA學會 論文誌
第4卷 第4號, 1999.12.

99-4-4-3-9

분산 객체 지향 데이타베이스에서 객체의 분산 설계

이 순 미*, 오 석**

Design of Object Distribution in Distributed Object-Oriented Databases

SoonMi Lee*, Suk Oh**

요 약

본 논문에서는 분산 객체 지향 데이타베이스에서 객체를 여러 사이트에 분산시키는 기법에 관하여 연구하였다. 제안된 객체의 분산 기법은 클래스의 분할 과정과 할당 과정으로 구성된다. 클래스의 분할 과정에서는 메소드, 계승 및 복합 객체와 같은 객체 지향 데이터베이스의 특성을 반영한 애트리뷰트 분할 알고리즘의 정의하였고 할당 과정에서는 함께 참조되는 다른 클래스의 프래그먼트들은 같은 사이트에 할당함으로써 데이타 전송량을 줄일 수 있는 할당 수식을 정의하였다.

Abstract

This paper addresses the design of object distribution in distributed object-oriented databases. The proposed strategy of object distribution consists of two-step design of fragments. One step is class fragmentation and the other is allocation of fragments. We define partitioning algorithms to reflect the characteristics of object-oriented databases in the class fragmentation and define expressions to reduce the amount of data transfer by allocating fragments of other classes referred together to one site in the allocation of fragments

* 경인여자대학 멀티미디어정보전산학부 전임강사
** 경인여자대학 멀티미디어정보전산학부 조교수
논문접수:1999.11.15. 심사완료:1999.12.8

Ⅰ. 서 론

최근에 하드웨어의 발달로 인해 고성능 워크스테이션 상에서 객체 지향 데이타베이스 기법을 요구하는 응용이 자주 발생함에 따라 분산 객체 지향 데이타베이스가 등장하게 되었다[7]. 분산 객체 지향 데이터베이스는 성능과 가용성의 향상을 위하여 클래스를 여러개의 프래그먼트로 분할하여 여러 사이트에 할당하는 객체의 분산 기법이 필요하다. 객체를 분산시킴으로써 동시성을 높이고 데이타 전송 및 중복을 줄이며 불필요한 데이타 접근을 줄일 수 있다. 객체 지향 데이터베이스에서 클래스를 분할하는 방법은 메소드, 계승, 복합 객체 등과 같은 객체 지향적인 특성으로 인해 관계형 데이터베이스의 분할 기법과 차이가 있다. 본 논문에서는 분산 객체 지향 데이타베이스에서 객체의 분산 기법을 클래스의 분할 과정과 할당 과정으로 나누어 설계하였다.

본 논문의 구성은 다음과 같다. 2장에서는 클래스에 속한 객체의 분산 개념과 전제 조건을 서술하고 3장에서는 클래스 분할 알고리즘을 정의하였다. 4장에서는 할당 수식을 정의하였으며 5장에서 결론을 맺었다.

Ⅱ. 전제조건

분산 객체 지향 데이터베이스에서 클래스를 분할하는 것은 기존의 관계형 데이터베이스에서의 분할과 다음과 같은 차이점이 있다. 첫째, 메소드의 동작이 분할에 반영되어야 한다. 객체 지향 질의는 애트리뷰트뿐만 아니라 메소드에도 접근할 수 있기 때문에 접근된 메소드가 참조하는 애트리뷰트도 분할에 반영이 되어야 한다. 둘째, 계승관계, 복합 객체 관계 등과 같은 다른 클래스와의 관계가 반영되어야 한다.

본 논문에서 제안된 객체의 분산 기법은 하나의 클래스가 여러개의 클래스 프래그먼트로 분할되어 여러 사이트에 할당되는데 클래스 프래그먼트는 애트리뷰트 프래그먼트와 메소드 프래그먼트로 구성된다. 그림 1에서와 같이 클래스에서 정의된 애트리뷰트의 집합과 메소드의 집합은 응용 질의를 기초로 하여 수직으로 분할되어 각각 애트리뷰트 프래그먼트와 메소드 프래그머트를 생성하게 된다.

본 논문에서 제안된 수직 분할 기법은 다음과 같은 전제 조건 하에서 수행된다.

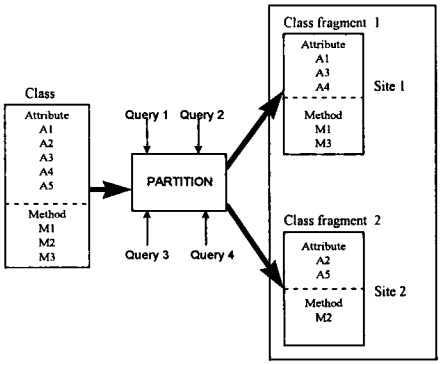

그림 1. 클래스 분할

【 전제 조건 】

첫째, 상위 클래스에서 하위 클래스로 애트리뷰트가 계승될 때에, 애트리뷰트의 실제 값은 정의된 상위 클래스에만 존재하며 하위 클래스로는 포인터만 계승된다.

둘째, 분할이 사용자 질의를 기초로 하여 수행되며, 데이타베이스에 관한 정보, 질의의 내용과 발생 빈도 등에 관한 정보는 분할되기 전에 이미 알려져 있다.

셋째, 질의는 애트리뷰트와 메소드로 구성된다. 질의내에서 메소드를 사용하는 것은 미리 정의된 메소드의 수행결과에 접근하는 것을 의미한다.

Ⅲ. 클래스 분할 알고리즘

1. 분할 행렬

클래스는 애트리뷰트와 메소드로 구성되기 때문에 클

래스의 분할도 애트리뷰트의 분할과 메소드의 분할로 구성된다. 본 논문에서 제안한 분할 방법은 우선 애트리뷰트를 분할한 후에 애트리뷰트를 참조하는 메소드들을 묶어서 메소드를 분할한다.

애트리뷰트의 분할은 질의를 기초로 이루어지게 되는데 객체 지향 데이터베이스에서 질의는 애트리뷰트 뿐만 아니라 메소드에도 접근할 수 있다. 또한 질의는 계승 관계나 복합 객체 관계를 통하여 다른 클래스에 속한 애트리뷰트나 메소드에도 접근할 수 있다. 이러한 객체 지향 질의의 특성을 반영하기 위하여 UQA 행렬, UMA 행렬, UAU 행렬을 정의하였는데 상세한 내용은 [8]에 나와있다.

- ▶UQA 행렬 : UQA(Universal Query Access) 행렬은 응용 질의를 통하여 참조되는 애트리뷰트와 메소드를 나타낸다.
- ▶UMA 행렬 : UMA(Universal Method Access) 행렬은 메소드가 접근하는 애트리뷰트를 나타낸다.
- ▶UAU 행렬 : UAU(Universal Attribute Usage) 행렬은 UQA 행렬과 UMA 행렬로부터 유도된 애트리뷰트를 포함한 질의에 의해 접근되는 모든 애트리뷰트를 나타낸다.

그림 3은 그림 2의 Employee 클래스에 대해 수행된 질의를 예로 한 UQA 행렬이다. 행렬에서 행(2Q1~2Q3)은 Employee 클래스에 대해 수행된 질의를 나타내며 열은 각 클래스에서 정의된 애트리뷰트와 메소드를 나타낸다. 원소 "1"은 질의가 애트리뷰트나 메소드를 참조했음을 나타내며 질의 빈도는 여러 사이트에서 발생한 질의 빈도의 합을 뜻한다. 예를 들어, 그림 3에서 질의2Q2는 애트리뷰트 eno 와 메소드 2m2에 접근한다. 그림 4에 의하면 2m2는 애트리뷰트 dpt 와 pno, name에 접근하는 메소드이다. 따라서, 그림 5에서 질의 2Q2에 의해 접근되는 모든 애트리뷰트는 eno, dpt (EMPLOYEE) 과 pno, name(PERSON로부터 계승) 이다.

UAU 행렬에서, 질의 qi에 대한 애트리뷰트 Aj의 사용 값 use(qi, Aj)는 다음과 같이 정의된다.

use(qi, Aj) = 1 qi가 Aj를 참조하는 경우
. 0 그렇지 않은 경우

UAU 행렬이 생성된 후에, 각각의 클래스에 대하여 애트리뷰트들 간의 결합력을 나타내 주는 AA (Attribute Affinity) 행렬이 생성된다. 한 클래스 Cm

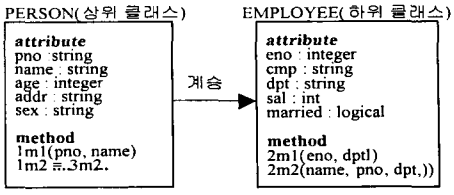

그림 2. 클래스 다이어그램

에 속한 두 애트리뷰트 Ai와 Aj 사이의 결합력은 다음과 같이 정의된다.

$$affm(Ai, Aj) = \sum_{k \mid use(qk, Ai)=1 \wedge use(qk, Aj)=1} acc(qk)$$

여기서, affm(Ai, Aj)는 클래스 Cm의 애트리뷰트인 Ai 와 Aj 사이의 결합값이며, acc(qk)는 질의 qk의 접근 수인데 이 때에 qk는 클래스 Cm 에 대한 질의 뿐 만 아니라 다른 클래스에 대한 질의도 될 수 있다. affm(Ai, Aj)는 애트리뷰트 Ai와 Aj를 함께 접근하는 모든 질의에 대한 acc의 합이다. 예를 들어, 그림 5의 UAU 행렬에서 EMPLOYEE 클래스의 eno, dpt에관한 결합값은 다음과 같다.

affEMPLOYEE(eno, dpt) = 40 + 30 = 70

그림 6은 DEPARTMENT 클래스에 대한 AA 행렬의 예이다. AA 행렬을 클러스터링한 후에 이것을 여러

	PERSON							EMPLOYEE							접근
	pno	name	age	addr	sex	1m	1m2	eno	cmp	dpt	sal	marr	2m1	2m2	빈도
2Q1	0	1	0	1	0	0	0	1	0	0	1	0	1	0	40
2Q2	0	0	0	0	0	0	0	1	0	0	0	0	0	1	30
2Q3	0	1	0	0	0	1	0	0	1	0	0	1	0	0	35

그림 3. Employee 질의 대한 UQA 행렬

	PERSON							EMPLOYEE							접근
	pno	name	ag.	addr	sex	1m	1m2	eno	cmp	dp	sal	marr	2m1	2m2	빈도
2m1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	40
2m2	1	1	0	0	0	0	0	0	0	1	0	0	0	0	30

그림 4. UMA 행렬

	PERSON					EMPLOYEE					접근
	pno	name	age	addr	sex	eno	cmp	dpt	sal	marr	빈도
2Q1	0	1	0	1	0	1	0	1	1	0	40
2Q2	1	1	0	0	0	1	0	1	0	0	30
2Q3	1	1	0	0	0	0	1	0	0	1	35

그림 5. UAU 행렬

애트리뷰트 프래그먼트로 나누게 되는데, 애트리뷰트를클러스터링하기 위하여 Bond Energy Algorithm (BEA)[5]이 사용된다. BEA를 사용하는 목적은 애트리뷰트 결합 행렬의 큰 값은 큰 값들끼리 모으고 작은 값은 작은 값들끼리 모으기 위함이다. 그림 6의 애트리뷰트 결합 행렬을 클러스터링한 결과가 그림 7의 CA(Cluster Affinity) 행렬이다. 애트리뷰트의 분할은 CA 행렬의 대각선을 따라 이루어지게 된다.

2. 애트리뷰트 분할 알고리즘

그림 7의 대각선 위에 한 점 X가 있다고 가정을 하자. 점 X에 의해 애트리뷰트는 두 부분, 즉 좌측 상단과 우측 하단 부분으로 나누어지며 이것을 각각 T와 B로 표기한다. 각 영역 T와 B에 속한 애트리뷰트들이 각각 애트리뷰트 프래그먼트에 해당한다. 임의의 클래스 Ck에 속한 애트리뷰트들을 분할하기 위하여 다음과 같은 정의가 필요하다.

A(Ck)={An | An은 클래스Ck에서 정의된 애트리뷰트이다.}

HQ(Ck)={qm | qm은 클래스 Ck에 대한 질의이다.}

FQ(Ck)={qm | qm은 Ck가 아닌 다른 클래스에 대한 질 의로서, Ck의 애트리뷰트에 접근하는 질의이다.}

HQ(Ck)는 내부질의, FQ(Ck)는 외부질의라고 부른다. 외부질의는 계승, 복합 및 메소드 링크 관계를 통해 Ck의 애트리뷰트에 접근하는 다른 클래스에 관한 질의를 의미한다.

UQ(Ck) = {qm | use(qm, An)=1 ∧ An∈A(Ck)}
= {qm | qj∈HQ(Ck) ∪ qm∈FQ(Ck)}
UA(Ck, qm)= {An | use(qm, An)=1 ∧ An∈A(Ck)}
TQ = {qm | UA(Ck, qm)⊆T}
BQ = {qm | UA(Ck, qm)⊆B}
IQ = UQ - {TQ ∪ BQ}

UQ(Ck)는 클래스 Ck의 애트리뷰트의 합집합이다. UA(Ck, qm)는 질의 qm가 사용한 클래스 Ck의 애트리뷰트의 집합이다. TQ는 상위 영역인 T에 속한 애트리뷰트만을 접근하는 질의이며 BQ는 하위 영역 B에 속한 애트리뷰트만을 접근하는 질의이고 IQ는 T와 B를 함께 접근하는 질의이다.

attribute	dname	member	budget	memNo	manager	comp
dname	140	35	105	0	35	25
member	35	100	0	0	100	0
budget	105	0	105	0	0	25
memNo	0	0	0	50	0	50
manager	35	100	0	0	100	0
comp	25	0	25	50	0	75

그림 6. DEPARTMENT에 대한 AA 행렬

attribute	memNo	comp	budget	dname	manager	member
memNo	50	50	0	0	0	0
comp	50	75	25	25	0	0
budget	0	25	105	105	0	0
dname	0	25	105	140	35	35
manager	0	0	0	35	100	100
member	0	0	0	35	100	100

그림 7. CA 행렬

TQ나 BQ와 같이 하나의 프래그먼트만을 접근하는 질의의 총 접근 수는 최대화하고 IQ와 같이 두 프래그먼트를 모두 접근하는 질의의 총접근 수는 최소화하는 위치가 최적의 위치이다. 최적의 위치를 찾아 분할을 하기 위하여 다음과 같은 수식의 정의가 필요하다.

$$CTQ = \sum_{qm \in TQ} acc(qm),$$

$$CBQ = \sum_{qm \in BQ} acc(qm)$$

$$CIQ = \sum_{qm \in IQ} acc(qm)$$

CTQ와 CBQ는 각각 한 프래그먼트 T나 B만을 접근하는 질의의 접근 수의 합이며 CIQ는 두 프래그먼트를 모두 접근하는 질의의 접근 수의 합이다. 수직 분할을 위한 분할 함수는 다음과 같이 정의된다.

$$Z = CTQ \times CBQ - CIQ2 \quad \cdots\cdots\cdots\cdots (1)$$

함수 Z의 값을 최대화시키는 점 X를 찾음으로써 최적의 수직 분할이 이루어진다.

Algorithm : QUERY_CLASSIFY(X)
input: CA matrix
x: a point at the diagonal of CA matrix.
T: attributes set of upper left-hand block of x
B : attributes set of lower right-hand block of x

```
output : TQ(x) : top query set at x.
         BQ(x) : bottom query set at x.
         IQ(x) : intersection query set at x.
begin
  TQ(x), BQ(x), IQ(x) ← empty set
  calculate HQ(CA)
  calculate FQ(CA)
  UQ(CA) ← HQ(CA) ∪ FQ(CA)
  repeat each qm∈UQ(CA) do
    UA(CA,qm)={An | use(qm,An)=
    1∧An∈A(CA)}
    if UA(CA, qm)⊆T then
       TQ(x) ← TQ(x) ∪ qm
    else if UA(CA, qm)⊆B then
       BQ(x) ← BQ(x) ∪ qm
    else
       IQ(x) ← IQ(x) ∪ qm
    end-if
  until no further element in UQ(CA)
end.
```

```
Algorithm : ATTRIBUTE_PARTITION(CA)
 input  : CA matrix
 output : Attribute Fragments
 begin
   x=1
   QUERY_CLASSIFY(1)
   CTQ = Σ_{qm∈TQ(1)} acc(qm)
   CBQ = Σ_{qm∈BQ(1)} acc(qm)
   CIQ = Σ_{qm∈IQ(1)} acc(qm)
   max = CTQ × CBQ - (CIQ)2
   begin
     for i from 2 to n-1 do
     begin
        z = 0
        QUERY_CLASSIFY(i)
        CTQ = Σ_{qm∈TQ(i)} acc(qm)
        CBQ = Σ_{qm∈BQ(i)} acc(qm)
        CIQ = Σ_{qm∈IQ(i)} acc(qm)
        z = CTQ × CBQ - (CIQ)2
      if z 〉 max then
         max = z :  x = i
      end-if
     end-for
   end-begin
   T←attribute set of upper left block of x
   B←attribute set of lower right block of x
   AF1 ← ∏T(CA)      * projection *
   AF2 ← ∏B(CA)
end.
```

Ⅳ. 프래그먼트의 할당

프래그먼트의 할당 단계에서 물리적인 비용 요소를 고려하여야 한다. 관계형 데이타베이스의 경우에는 할당 시에 데이타 전송 비용, 불필요한 애트리뷰트 접근 비용, 애트리뷰트의 크기, 프래그먼트 접근 비용 및 저장 장치 비용 등을 고려하고 있으나, 객체 지향 데이타베이스의 경우에는 추가로 다른 클래스와의 관계가 고려되어야 한다. 즉, 계승 관계, 복합 관계 및 메소드 링크 관계[8]를 통해 함께 참조되는 다른 클래스의 프래그먼트들은 같은 사이트에 할당함으로써 데이타 전송량을 줄일 수 있다. 본 장에서는 다른 클래스의 프래그먼트들 사이에서 데이타 전송을 줄이기 위한 할당 수식을 정의한다.

할당 수식을 서술하기 위하여 다음과 같은 정의가 필요하다.

Li : 애트리뷰트 i의 길이
Fij : 클래스 i의 프래그먼트 j
Alloc(F): 프래그먼트 F의 할당 사이트
Accqi : 사이트 i에서 발생한 질의 q의 접근 수
A(q) : 질의 q가 접근하는 애트리뷰트의 집합
HQ(Fij) : Fij에 대하여 발생한 질의의 집합
FQ(Fij) : Fij가 아닌 다른 클래스의 프래그먼트에

대한 질의로서, Fij에 속한 애트리뷰트에 접근하는 질의의 집합한 클래스 프래그먼트에 대한 질의가 계승, 복합 및 메소드 링크 관계를 통하여 다른 클래스 프래그먼트에 속한 애트리뷰트를 호출할 수 있는데, 이러한 프래그먼트들은 다음과 같이 정의된다.

【정의1】 서로 다른 클래스에 대한 프래그먼트 Fmn과 Fij에서, Fmn의 내부 질의인 HQ(Fmn)에 속하는 질의가 Fij에 속하는 애트리뷰트를 호출할 때에 Fmn와 Fij의 관계를 Fmn Fij 로 표기하며, Fmn를 호출 프래그먼트라고 정의하며 Fij를 피호출 프래그먼트라고 정의한다.

한 피호출 프래그먼트 Fij를 호출하는 호출 프래그먼트들의 집합은 다음과 같은 수식으로 표시된다.

$$Invoke(Fij)=\{Fmn \mid \forall q,\ q\in FQ(Fij) \wedge q\in HQ(Fmn)\}$$

Invoke(Fij)는 Fij에 속한 애트리뷰트를 호출하는 호출 프래그먼트의 집합으로서 Invoke(Fij)에 속하는 프래그먼트는 계승, 복합 및 메소드 링크 관계를 통하여 프래그먼트 Fij에 접근한다. 그림 8은 호출 프래그먼트 Fmn과 피호출 프래그먼트 Fij의 관계를 그림으로 표시한 것이다. 그림 8에서 Fmn과 Fij는 다른 클래스의 프래그먼트이지만 서로 연관되어 있기 때문에 같은 사이트에 할당을 함으로써 데이타 전송량을 줄일 수 있다. 그러나, Fmn과 Fij를 같은 사이트에 할당할 때에, 호출 프래그먼트가 저장된 사이트에 피호출 프래그먼트를 할당하여야지 반대의 경우는 오히려 데이타 전송량을 증가시킨다. 그림 8에서 질의 q는 Fmn의 a1, a3과 Fij의 b1, b2에 접근한다. 피호출 프래그먼트 Fij를 사이트 1에 할당할 경우에는 질의 q를 실행하기 위하여 데이타 전송이 필요하지 않지만, 반대로 사이트 2에 호출 프래그먼트 Fmn을 할당하는 경우에는 사이트 1에서 발생한 질의 q를 실행하기 위하여 a1, a3, b1 및 b2를 전송해야 한다.

따라서, 프래그먼트들 사이에서 데이타의 전송을 줄이기 위한 할당 수식은 피호출 프래그먼트를 대상으로 실행되며, 다음의 할당 수식 값이 가장 작은 사이트에 피호출 프래그먼트를 할당한다.

피호출 프래그먼트 Fij를 할당하기 위한 수식은 다음과 같이 정의된다.

$$W4 = A\times B\times C$$

이 때에,

$$A = \sum_{Fmn \mid (Fmn\in Invoke(Fij))\wedge(Alloc(Fmn)\neq Alloc(Fij))}$$

$$B = \sum_{q,k \mid (q\in HQ(Fmn))\wedge(q\in FQ(Fij)),\ Alloc(Fmn)=k} accqk$$

$$C = \sum_{p\in(Fij\cap A(q))} Lp$$

A는 Fij를 호출하며 Fij와 같은 사이트에 할당되지 않은 모든 호출 프래그먼트 Fmn를 의미하며, B는 Fij의 외부 질의이면서 Fmn의 내부 질의인 q가 Fmn의 할당 사이트인 k에서 발생한 질의 접근수의 합을 뜻한다. C는 질의 q가 접근하는 피호출 프래그먼트 Fij내의 애트리뷰트들의 길이의 합이다.

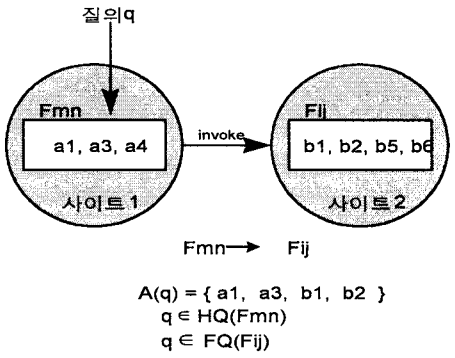

그림 8. 호출과 피호출 프래그먼트의 관계

W4의 값이 가장 적은 사이트에 피호출 플래그먼트 Fij를 할당함으로써 호출과 피호출 프래그먼트 사이의 데이타 전송량을 줄일 수 있다. 프래그먼트의 할당 순서는 호출 프래그먼트를 먼저 사이트에 할당한 후에 피 호출 프래그먼트를 할당하여야 한다.

V. 결론

본 논문에서는 분산 객체 지향 데이타베이스에서 객체의 분산 기법을 클래스의 분할 과정과 할당 과정으로 나누어 설계하였다. 분할 과정에서 정의한 알고리즘은 기존의 관계형 데이터베이스의 분할 기법을 객체 지향 환경으로 확장한 것으로서 메소드, 계승 및 복합 객체와 같은 특성을 반영하였다. 할당 과정에서는 계승 관계, 복합 관계 및 메소드 링크 관계를 통해 함께 참조되는 다른 클래스의 프래그먼트들은 같은 사이트에 할당함으로써 데이타 전송량을 줄일 수 있는 할당 수식을 정의하였다.

추후의 연구과제로서 정의한 분산 기법을 시뮤레이션을 통하여 성능 분석하는 연구가 필요하다.

참고문헌

[1] Ceri, S. & Pelagatti, G. Distributd atabases : Principles and Systems. NY, McGraw Hill, 1984.

[2] Ezeife, C. I. & Barker, K. Vertical Class Fragmentation in a Distributes Object Based System. TR 94-03, Univ. of Manitoba, 1993.

[3] Hoffer, J. A., & Severance, D. G. The Use of Cluster Analysis in Physical Database Design. In 1st VLDB Conference, Framingham, Mass., 1975.

[4] Karlapalem, K., Navathe, S. B. & Morsi, M. M. A. Issues in Distribution design of OODB. In Distributed Object Management, pages 148-164. Morgan Kaufmann Publishers, 1994.

[5] McCormick, W. T. & Schweitzer, P. J., Problem Decomposition and Data Reorganization by a Clustering Technique. Oper. Res. 20, 1977.

[6] Navathe, S. B., Ceri, S. Wiederhold, G. & Dou, J. Vertical partitioning algorithms for database design. in ACM TODS 9(4), 1984.

[7] Ozsu, M. T. and Valduriez, P. Distributed Database System: Where Are We Now? IEEE Computer Vol. 24, No. 8, 1991.

[8] 이순미, 임해철, 분산 객체 지향 데이타베이스에서 클래스의 수직 분할 기법, 한국통신학회 논문지, 제22권 제2호, 1997.

저자소개

이 순 미

1984년 2월 : 이화여자대학교 수학과(학사)

1986년 2월 : 이화여자대학교 대학원 수학과 전자계산 전공(석사)

1997년 8월 : 홍익대학교 전자계산학과(박사)

1998년~현재 : 경인여자대학 멀티미디어정보전산학부

오 석

1986년 2월 : 이화여자대학교 전자계산학과(학사)

1991년 9월 : 파리6대학 인공지능전공(석사)

1995년 6월 : 라호셀대학 전산학(박사)

1986~1990 : 시스템공학연구소

1997년~현재 : 경인여자대학 멀티미디어정보전산학부