분산 객체지향 데이타베이스에서 분산 설계 및 구현

이 순 미[†] · 박 혜 숙[†] · 하 얀[††]

요 약

본 논문에서는 인터넷상의 대용량 자료에서 원하는 정보를 검색하기위한 지원 기능으로서 분산 객체지향 데이타베이스에서 클래스를 분할하여 여러 사이트에 분산시키는 기법에 관하여 설계 및 구현하였다. 제안된 분산 기법은 클래스의 분할 과정과 할당 과정으로 구성된다. 클래스의 분할 과정에서는 메소드, 계승 및 복합 객체와 같은 객체지향 데이터베이스의 특성을 반영하여 클래스를 분할하였으며 할당 과정에서는 저장, 질의 처리 및 전송비용을 고려하여 할당수식을 정의하였으며 이를 유전자 알고리즘을 이용하여 구현하였다.

Design and Implementation of Distribution in Distributed Object-Oriented Databases

Soon-Mi Lee[†] · Hea-Sook Park[†] · Yan Ha[††]

ABSTRACT

This paper addresses the design and implementation of class distribution in distributed object-oriented databases. The proposed strategy of distribution consists of two-step design of fragments. One is class fragmentation and the other is allocation of fragments. In step of class fragmentation, we have defined partitioning algorithms to reflect the characteristics of object-oriented databases such as method, inheritance and composite-object. In step of allocation, we have defined the objective function for allocation considering system operating cost including storage, query processing and communication and implemented it using Genetic Algorithm.

키워드 : 클래스 분할(Class Fragmentation), 데이타 할당(Data Allocation), 분산 객체지향 데이터베이스(Distributed Object-Oriented Databases), 유전자 알고리즘(Genetic Algorithm)

1. 서 론

최근에 하드웨어의 발달로 인해 고성능 워크스테이션 상에서 객체지향 데이타베이스 기법을 요구하는 응용이 자주 발생함에 따라 분산 객체지향 데이타베이스가 등장하게 되었다[9]. 또한 인터넷과 WWW(World Wide Web)에 의한 브라우징이 일반화됨에 따라 정보통신의 활용이 확산되고 응용영역도 넓어지고 있다. 이러한 추세에서 대용량 자료에서 원하는 정보를 검색하기 위한 지원기능으로 데이터를 효율적으로 분산시키는 연구가 필요하다.

분산 객체지향 데이터베이스에서 클래스를 여러개의 프래그먼트로 분할하여 지리적으로 떨어져 있는 여러 사이트에 할당함으로써 동시성을 높이고 데이타 전송 및 중복을 줄이며 불필요한 데이타 접근을 줄일 수 있어 보다 빠른 속도로 정보를 검색할 수 있게 된다.

객체지향 데이터베이스에서 클래스를 분할하는 방법은 메소드, 계승, 복합 객체 등과 같은 객체지향의 특성으로 인해 관계형 데이터베이스의 분할 기법과 차이가 있다. 따라서, 본 논문에서는 객체지향의 특성이 반영된 객체의 분산 방법을 연구하였으며 클래스의 분할 과정과 할당 과정으로 나누어 설계 및 구현하였다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구와 전제조건을 서술하였고 3장에서는 클래스의 분할 알고리즘을 기술하였으며 4장에서는 분할된 클래스를 할당하기위한 할당 수식을 정의하였고 5장에서 구현하였으며 6장에서 결론을 맺었다.

2. 전제조건

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

† 정 회 원 : 경인여자대학 컴퓨터정보기술학부 교수
†† 준 회 원 : 경인여자대학 컴퓨터정보기술학부 교수
논문접수 : 2004년 4월 30일, 심사완료 : 2004년 8월 17일

관계 등과 같은 다른 클래스와의 관계가 반영되어야 한다.

본 논문에서 제안된 객체의 분산 기법은 하나의 클래스가 여러 개의 클래스 프래그먼트로 분할되어 여러 사이트에 할당되는데 클래스 프래그먼트는 애트리뷰트 프래그먼트와 메소드 프래그먼트로 구성된다. 클래스에서 정의된 애트리뷰트의 집합과 메소드의 집합은 응용 질의를 기초로 하여 수직으로 분할되어 다른 사이트에 할당된다. 분할 단계에서는, 하나의 클래스가 여러 개의 프래그먼트로 나누어지며, 각각의 애트리뷰트는 2개 이상의 프래그먼트에 중복하여 속할 수 없고 오직 하나의 프래그먼트에만 속하는 비중복 다중분할이 수행된다. 할당 단계에서는 할당 비용을 최소로 하는 사이트에 프래그먼트가 할당되는데 이 때에는 한 프래그먼트가 여러 사이트에 중복하여 할당이 가능하도록 설계하였다.

본 논문에서 제안된 분산 기법은 다음과 같은 전제 조건 하에서 수행된다.

[전제 조건]

첫째, 클래스의 분할은 사용자 질의를 기초로 하여 수행되며, 데이타베이스에 관한 정보, 질의의 내용과 발생 빈도 발생노드 등에 관한 정보는 분할되기 전에 이미 알려져 있다.

둘째, 질의는 애트리뷰트 뿐만 아니라 메소드에도 접근할 수 있다. 질의 내에서 메소드를 사용하는 것은 미리 정의된 메소드의 수행 결과에 접근하는 것을 의미한다.

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

넷째, 클래스 프래그먼트의 할당시에 고려되는 질의 처리 비용에서 무결성(Integrity) 유지 비용과 동시성 제어(Concurrency Control) 비용은 고려하지 않았다.

3. 클래스 분할

3.1 분할 행렬

클래스는 애트리뷰트와 메소드로 구성되기 때문에 클래스의 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 객체지향 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 객체지향 질의의 특성을 반영하기 위하여 UQA 행렬, UMA 행렬, UAU 행렬을 정의하였다.

- UQA 행렬 : UQA(Universal Query Access) 행렬은 응용 질의를 통하여 참조되는 애트리뷰트와 메소드를 나타낸다. UQA 행렬에서는 상속 또는 복합 객체를 통하여 접근하는 애트리뷰트와 메소드까지 표시된다.

(그림 2)는 (그림 1)의 Employee 클래스에 대해 수행된 질의를 예로 한 UQA 행렬이다. 행렬에서 2Q1~2Q3은 Employee 클래스에 대해 수행된 질의를 나타내며 발생 빈도는 여러 사이트에서 발생한 질의 빈도의 합을 뜻한다. 행렬에서 질의 q_i에 대한 A_j의 사용 값 use(q_i, A_j)는 다음과 같이 정의된다. 여기서 A_j는 애트리뷰트 또는 메소드를 나타낸다.

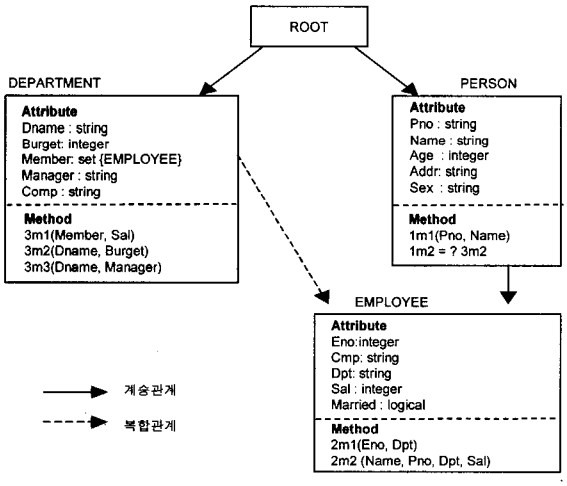
$$\mathbf{use(q_i, A_j)} = \begin{cases} 1 & q_i\text{가 } A_j\text{를 참조하는 경우} \\ 0 & \text{그렇지 않은 경우} \end{cases}$$

- UMA 행렬 : UMA(Universal Method Access)행렬은 메소드가 접근하는 애트리뷰트를 나타낸다.
- UAU 행렬 : UAU(Universal Attribute Usage) 행렬은 UQA 행렬과 UMA 행렬로부터 유도된 행렬로서 질의가 접근하는 모든 애트리뷰트를 나타낸다. 즉, 질의가 직접 접근하는 애트리뷰트 뿐만아니라 메소드를 통하여 참조되는 애트리뷰트까지 표시된다.

(그림 4)는 UAU 행렬의 예로서 1Q1~1Q3은 Person에 대한 질의이며 2Q1~2Q3은 Employee에 대한 질의를 나타낸다. 예를 들어 살펴보면, (그림 2)에서 질의2Q2는 애트리뷰트 eno와 메소드 2m2에 접근하는데 (그림 3)에 의하면 2m2는 dpt와 pno, name에 접근하는 메소드이다. 따라서, (그림 4)에서 질의 2Q2에 의해 접근되는 애트리뷰트는 Employee 클래스의 eno, dpt와 Person으로부터 계승된 pno, name이다.

UAU 행렬이 생성된 후에, 각각의 클래스에 대하여 애트리뷰트들 간의 결합력을 나타내 주는 **AA(Attribute Affinity) 행렬**이 생성된다. 한 클래스 C_m에 속한 두 애트리뷰트 A_i와 A_j 사이의 결합력은 다음과 같이 정의된다.

$$\mathbf{aff_m(A_i, A_j)} = \sum_{k \mid use(qk, Ai)=1 \wedge use(qk, Aj)=1} acc(qk)$$



(그림 1) 클래스 관계

	PERSON							EMPLOYEE							발생 빈도
	애트리뷰트					메소드		애트리뷰트					메소드		
	pno	name	age	addr	sex	1m1	1m2	eno	cmp	dpt	sal	marr	2m1	2m2	
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 2) Employee 질의 대한 UQA 행렬

	PERSON							EMPLOYEE							발생 빈도
	애트리뷰트					메소드		애트리뷰트					메소드		
	pno	name	age	addr	sex	1m1	1m2	eno	cmp	dpt	sal	marr	2m1	2m2	
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 3) UMA 행렬

질의	PERSON					EMPLOYEE					발생 빈도
	애트리뷰트					애트리뷰트					
	pno	name	age	addr	sex	eno	cmp	dpt	sal	marr	
1Q1	1	1	0	1	0	0	0	0	0	0	40
1Q2	1	1	0	1	0	0	0	0	0	0	45
1Q3	0	1	1	0	0	0	0	0	0	0	40
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 4) UAU 행렬

여기서, $aff_m(A_i, A_j)$는 애트리뷰트인 A_i와 A_j 사이의 결합값으로서 애트리뷰트 A_i와 A_j를 함께 접근하는 모든 질의의 발생빈도의 합이다. $acc(q_k)$는 질의 q_k의 발생 빈도인데 이 때에 q_k는 클래스 C_m에 대한 질의 뿐 만아니라 객체지향의 계승, 복합객체 특성을 반영하여 다른 클래스에 대한 질의도 될 수 있다. 예를 들어, (그림 4)의 UAU 행렬에서 Person 클래스의 pno와 name에 관한 결합값은 Person 클래스에 대한 질의인 1Q1과 1Q2 뿐만 아니라 pno와 name를 계승받아 접근하는 Employee 클래스의 2Q2와 2Q3의 질의 빈도의 합이 된다.

$$aff_{PERSON}(pno, name) = 40 + 45 + 30 + 35 = 150$$

위와 같이 계산된 클래스에 속한 애트리뷰트들 간에 결합값을 원소로 하여 AA 행렬이 생성된 후에는 이것을 클러스터링하여 여러 개의 애트리뷰트 프래그먼트로 나누게 된다. 애트리뷰트를 클러스터링하는 데에는 Bond Energy Algorithm(BEA)[7]이 사용된다. BEA를 사용하는 목적은 AA 행렬의 큰 값은 큰 값들끼리 모으고 작은 값은 작은 값들끼리 모으기 위함이다.

(그림 1)의 Department 클래스에 대한 AA 행렬이 (그림 5)와 같다고 하자. 이 행렬을 클러스터링한 후에 여러 애트리뷰트 프래그먼트로 나누게 되는데, (그림 5)를 클러스터링한 결과가 (그림 6)의 **CA(Cluster Affinity) 행렬**이다. 애트리뷰트의 분할은 CA 행렬의 대각선을 따라 이루어지게 된다.

3.2 애트리뷰트 분할

(그림 6)의 CA 행렬의 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 C_k에 속한 애트리뷰트들을 분할하기 위하여 다음과 같은 정의가 필요하다.

$A(C_k)$ = {A_n | A_n은 클래스 C_k에서 정의된 애트리뷰트이다.}
$HQ(C_k)$= {q_m | q_m은 클래스 C_k에 대한 질의이다.}
$FQ(C_k)$= {q_m | q_m은 C_k가 아닌 다른 클래스에 대한 질의로서, C_k의 애트리뷰트에 접근하는 질의이다.}

$HQ(C_k)$는 **내부질의**, $FQ(C_k)$는 **외부질의**라고 부른다. 외부질의는 계승, 복합 및 메소드 링크 관계를 통해 C_k의 애트리뷰트에 접근하는 다른 클래스에 관한 질의를 의미한다.

$$UQ(C_k) = \{q_m \mid use(q_m, A_n) = 1 \wedge A_n \in A(C_k)\}$$
$$= \{q_m \mid q_j \in HQ(C_k) \cup q_m \in FQ(C_k)\}$$
$$UA(C_k, q_m) = \{A_n \mid use(q_m, A_n) = 1 \wedge A_n \in A(C_k)\}$$
$$TQ = \{q_m \mid UA(C_k, q_m) \subseteq T\}$$
$$BQ = \{q_m \mid UA(C_k, q_m) \subseteq B\}$$
$$IQ = UQ - \{TQ \cup BQ\}$$

$UQ(C_k)$는 클래스 C_k의 애트리뷰트를 참조하는 모든 질의로서 내부질의와 외부질의의 합집합이다. **$UA(C_k, q_m)$**는 질의 q_m가 사용한 클래스 C_k의 애트리뷰트의 집합이다. **TQ**는 상위 영역인 T에 속한 애트리뷰트만을 접근하는 질의이

애트리뷰트	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 5) Department에 대한 AA 행렬

애트리뷰트	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 6) 클러스터링한 후의 결과(CA 행렬)

며 **BQ**는 하위 영역 B에 속한 애트리뷰트만을 접근하는 질의이고 **IQ**는 T와 B를 함께 접근하는 질의이다. TQ나 BQ와 같이 하나의 프래그먼트만을 접근하는 질의의 총 접근수는 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.

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

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

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

CTQ와 CBQ는 각각 한 프래그먼트 T나 B만을 접근하는 질의의 발생 빈도의 합이며 CIQ는 두 프래그먼트를 모두 접근하는 질의의 발생 빈도의 합이다. 수직 분할을 위한 **분할 함수**는 다음과 같이 정의된다.

$$Z = CTQ \times CBQ - CIQ^2 \quad (1)$$

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

애트리뷰트를 분할하여 두 개의 애트리뷰트 프래그먼트를 생성한 후에 각각의 프래그먼트에 대해 다시 반복적으로 이중 분할을 수행함으로써 다중 분할이 이루어진다. 위에서 정의한 분할 함수(식 (1))의 최대값이 양수인 경우는 분할이 이루어지며 음수인 경우는 분할 수행을 취소한다.

3.3 메소드의 분할

한 클래스에 속한 애트리뷰트가 분할되어 애트리뷰트 프래그먼트를 형성한 후에 메소드의 분할이 진행된다. 메소드 분할은 애트리뷰트 프래그먼트에 속한 애트리뷰트들을 참조하는 메소드들을 모음으로써 이루어진다.

[정의 1] 클래스 C의 각 애트리뷰트 프래그먼트 AF_i에 대한 **참조 메소드 집합** ***RM***$_i$는 AF_i내의 모든 애트리뷰트들 각각을 참조하는 메소드들의 합집합이다.

메소드가 프래그먼트에 할당될 때에 같은 메소드가 중복되어 할당될 수도 있다. 이것은 한 메소드에 의해 참조되는 애트리뷰트들 중 일부는 한 프래그먼트에 속하고 다른 일부는 다른 프래그먼트에 속함을 의미한다. 클래스 프래그먼트는 일종의 클래스로서 객체지향 개념을 유지하여야 하는데 중복되는 메소드는 객체지향의 캡슐화 개념에 위배된다. 이러한 문제점을 해결하기 위하여 수직 분할을 하는 클래스 C의 내부적인 구조를 C′로 변경하며, C′와 프래그먼트 사이에 클래스 구성 계층이 형성되도록 한다. 그리고, 중복되는 메소드를 참조 메소드 집합으로부터 제거하여 **클래스 구성 계층** 상의 참조하는 클래스(부모 클래스)에 배치시킨다.

[정의 2] 클래스 C의 각 애트리뷰트 프래그먼트 AF_i에 대한 **메소드 프래그먼트** ***MF***$_i$는 C의 모든 참조 메소드 RM_i에서 중복되는 메소드를 제거시킨 것이다.

[정의 3] 클래스 프래그먼트 CF$_i$는 애트리뷰트 프래그먼트 AF_i와 그에 해당하는 메소드 프래그먼트 MF_i를 결합한 것이다.

4. 프래그먼트의 할당

할당 단계에서는 객체지향의 특성을 반영하여 분할된 클래스 프래그먼트를 컴퓨터 통신망을 통해 흩어져 있는 여러 사이트 중 최적의 사이트에 할당한다. 여기서 최적의 사이트란 데이터를 저장하는 비용, 질의를 처리하는 비용 및 원하는 데이터가 자신의 사이트에 없을 때에 발생하는 전송 비용 등을 고려한 할당 비용을 최소로 하는 사이트를 의미한다.

본 논문에서 제안한 분할 기법에서는 메소드를 통하여 접근되는 애트리뷰트까지 반영하여 애트리뷰트 프래그먼트를 생성하기 때문에 할당 시에는 메소드 프래그먼트는 고려하지 않고 애트리뷰트 프래그먼트만을 고려하여 할당 수식을 정의하였으며 최종적으로 애트리뷰트 프래그먼트가 할당될 노드가 결정되면 대응되는 메소드 프래그먼트를 해당 노드에 할당하는 것으로 설계하였다.

할당 수식을 정의하기 위해서는 데이타베이스와 사용자 질의에 관한 정보 및 통신망 정보, 각 사이트에 대한 저장 용량과 처리 성능에 대한 정보가 정량화되어 있어야 한다. 다음은 할당에 필요한 요구사항들을 정량화하여 정의하였다.

데이터 할당 모델은 다음과 같은 세 가지 구성요소로 구성되어 있다.

- 프래그먼트의 집합 : $F = \{F_1, F_2, F_3 \cdots Fn\}$
- 사이트들의 집합 : $S = \{S_1, S_2, S_3 \cdots Sm\}$
- 질의들의 집합 : $F = \{q_1, q_2, q_3 \cdots qn\}$

① 데이터베이스 정보

- 프래그먼트 F_j의 크기 : $size(F_j) = card(F_j) \times length(F_j)$ ($length(F_j)$는 프래그먼트 F_j에 속한 애트리뷰트들의 바이트 수)

② 사용자 질의 정보

- CR_{ij} : 질의 q_i가 프래그먼트 F_j에 대하여 읽기 연산을 수행한 접근의 수
- CU_{ij} : 질의 q_i가 프래그먼 F_j에 대하여 갱신 연산을 수행한 접근의 수
- $O(i)$: 질의 q_i가 발생한 사이트

③ 사이트 정보

- USC_k : 사이트 S_k에 데이타를 저장하는 데에 필요한 단위비용

- UPC_k : 사이트 S_k에서 한 단위의 일을 처리하는 데에 필요한 비용

④ 통신망 정보
- g_{ij} : 사이트 S_i와 S_j 사이에 프래임 당 통신 비용
- $fsize$: 한 프래임의 크기(바이트 수)

4.1 결정 변수

본 논문에서는 다음과 같은 비용 요소를 반영하는 할당 수식을 정의고자 한다.

C1 : 프래그먼트를 사이트에 저장하는 비용
C2 : 질의 처리 비용
C3 : 전송 비용

C1은 모든 프래그먼트와 모든 사이트에 대한 총저장 비용을 의미하며 C2는 프래그먼트를 검색하고 갱신하기 위하여 필요한 CPU 비용으로서 접근(Access) 비용과 무결성(Integrity) 유지 비용과 동시성 제어(Concurrency Control) 비용이 있을 수 있으나 본 논문에서는 무결성 유지 비용과 동시성 제어 비용은 고려하지 않았다. C3은 검색과 갱신 질의를 수행하는 과정에서 발생하는 사이트 간에 메시지와 데이타의 전송 비용을 나타낸다.

위의 비용 요소를 고려하여 총 비용을 최소화시키는 것이 최적의 할당 해가 된다. 할당 수식을 정의하기 위한 결정 변수를 다음과 같이 정의한다.

$$r_{ij} = \begin{cases} 1 & \text{질의 } qi\text{가 프래그먼트 } Fj\text{를 검색한 경우} \\ 0 & \text{그렇지 않은 경우} \end{cases}$$

$$u_{ij} = \begin{cases} 1 & \text{질의 } qi\text{가 프래그먼트 } Fj\text{를 갱신한 경우} \\ 0 & \text{그렇지 않은 경우} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{프래그먼트 } Fi\text{가 사이트 } Sk\text{에 저장된 경우} \\ 0 & \text{그렇지 않은 경우} \end{cases}$$

4.2 할당 함수

할당 함수를 정의하는 목적은 저장비용과 질의처리비용 및 전송비용을 포함한 시스템 운영 비용을 최소화하기 위함이다. 할당함수(AC)는 다음과 같다.

$$\mathrm{AC} = \sum_{\forall Sk \in S} \sum_{\forall Fj \in F} STC_{jk} + \sum_{\forall qi \in Q} QPC_i + \sum_{\forall qi \in Q} TC_i$$

여기서, STC_{jk}는 프래그먼트 F_j를 사이트 S_k에 저장하는 비용이며 QPC_i는 질의 q_i를 처리하기 위한 접근 비용을 의미하며 TC_i는 통신 비용을 의미한다. 할당 함수 AC를 최소로 하는 사이트 할당이 최적의 할당 사이트가 된다.

4.2.1 저장 비용

프래그먼트 F_j를 사이트 S_k에 저장하는데 드는 비용 STC_{jk}는 다음과 같이 산출된다.

$$STC_{jk} = USC_k \times size(F_j) \times x_{jk}$$

여기서, USC_k는 사이트 S_k에 데이타를 저장하는 데에 필요한 단위비용이며 $size(F_j)$는 프래그먼트 F_j의 크기를 의미한다.

4.2.2 질의처리 비용

질의 q_i를 처리하는데 드는 비용 QPC_i는 프래그먼트를 검색 및 갱신하는 데에 필요한 접근 비용(CPU 비용)이다. 본 논문에서는 같은 사이트 내에서 검색과 갱신을 처리하는 비용은 동일한 것으로 가정하였으며 무결성 유지 비용과 동시성 제어 비용은 고려하지 않았다.

$$QPC_i = \sum_{\forall Sk \in S} \sum_{\forall Fj \in F} (r_{ij} \times CR_{ij} + u_{ij} \times CU_{ij}) \times x_{jk} \times UPC_k$$

여기서, $(r_{ij} \times CR_{ij} + u_{ij} \times CU_{ij})$는 질의 q_i가 프래그먼트 F_j에 접근하는 수를 계산하는 식이며 이 때에 CR_{ij}는 읽기 연산을 수행한 접근의 수를 나타내며 CU_{ij}는 갱신 연산을 수행한 접근의 수를 나타낸다. UPC_k는 사이트 S_k에서 한 단위의 일을 처리하는 데에 필요한 비용이다.

4.2.3 전송 비용

분산 데이타베이스에서 갱신 요청을 수행하기 위한 전송 방식과 검색 요청을 수행하기 위한 데이타 전송 방식은 매우 다르다[10, 12].

본 논문에서는 검색 질의 수행과 갱신 질의 수행의 차이점을 반영하기 위하여 전송 비용 수식을 검색 전송 비용(RTC_i)와 갱신 전송 비용(UTC_i)으로 나누어서 정의하였다.

- **검색 연산 전송 비용**

검색에 필요한 전송 비용은 다음과 같이 정의된다.

$$RTC_i = \sum_{\forall Fj \in F} \{ \min_{Sk \in S} (ATC + BTC) \} \times CR_{ijO(i)}$$

단, $ATC = \mathrm{r}_{ij} \times x_{jk} \times g_{o(i),k}$,

$$BTC = \mathrm{r}_{ij} \times x_{jk} \times \frac{seli(Fj)}{fsize} \times g_{k,o(i)}$$

여기서, ATC는 검색요청을 전송하는 비용이며 BTC는 질의 발생 사이트로 결과를 전송하는 비용이 다. $CR_{ijO(i)}$는 질의 q_i가 발생한 사이트 $o(i)$에서 프래그먼트 F_j에 대하여 수행되는 검색질의 q_i의 발생 빈도수를 나타낸다. $g_{o(i),k}$는 질의 발생 사이트인 $o(i)$와 사이트 K 사이의 통신 비용을 의미한다.

- **갱신 연산 전송 비용**

갱신 연산에 필요한 전송 비용은 다음과 같이 정의된다.

$$UTC_i = \sum_{\forall Sk \in S} \sum_{\forall Fj \in F} 2(STC + CTC) \times CU_{ijO(i)}$$

단, $STC = \mathrm{u}_{ij} \times x_{jk} \times g_{o(i),k}$

$CTC = \mathrm{u}_{ij} \times x_{jk} \times g_{k,o(i)}$

여기서, STC는 질의가 발생한 사이트에서 다른 사이트 k로 메시지를 보내는 비용이며 CTC는 사이트 k에서 질의가

	PERSON					EMPLOYEE					DEPARTMENT						접근 빈도
	pno	name	age	addr	sex	eno	cmp	dpt	sal	marr	dname	member	budget	memNo	manager	comp	
1Q1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	40
1Q2	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	60
1Q3	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	45
1Q4	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	45
1Q5	0	1	1	0	0	0	0	0	0	0	1	0	1	0	0	0	40
2Q1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	25
2Q2	0	1	0	1	0	1	0	1	1	0	0	0	0	0	0	0	40
2Q3	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	25
2Q4	1	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	30
2Q5	1	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	35
3Q1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	40
3Q2	0	0	0	0	0	0	1	0	0	1	1	1	0	0	1	0	35
3Q3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	50
3Q4	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	25
3Q5	0	0	0	0	0	1	0	0	1	0	0	1	0	0	1	0	40
3Q6	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	25

(그림 7) 전체 질의 사용 행렬(UAU 행렬)

발생한 사이트로 확인 메시지를 반환하는 비용이다. 갱신 수행 단계에서 STC와 CTC는 각각 2번씩 발생하게 된다. $CU_{ijo(i)}$는 질의 q_i가 발생한 사이트 $o(i)$에서 프래그먼트 F_j에 대하여 수행되는 갱신질의 q_i의 발생 빈도수를 나타낸다.

5. 구 현

5.1 클래스 분할의 구현

본 논문에서 설계된 클래스 수직 분할 기법을 IBM-PC 상에서 C 언어로 구현하였다. 구현 과정은 첫째로 예제 클래스 모델과 질의를 설정한다. 클래스 모델의 스키마는 (그림 1)과 같고 전체 질의 사용 행렬(UAU 행렬)은 (그림 7)과 같다. 둘째, 각각의 클래스에 대해 애트리뷰트 결합 행렬과 클러스터 결합 행렬을 생성한다. 클러스터 결합 행렬의 결과는 (그림 8)과 같다. 셋째, 애트리뷰트 프래그먼트와 메소드 프래그먼트를 생성한다. 분할 방법은 3.2절의 분할수식 식 (1)을 최대화시키는 위치를 클러스터 결합 행렬에서 찾는 것이다.

각각의 클래스에 대한 분할 결과는 다음과 같다.

• PERSON 클래스의 분할

◦ 애트리뷰트 프래그먼트 :

$AF1$ = {*addr, pno, name*}

$AF2$ = {*age, sex*}

◦ 메소드 프래그먼트 :

$MF1$ = {$1m1$, $1m2$, }, $MF2$ = { }

따라서, 클래스 프래그먼트는 다음과 같다.

$P1$ = {$AF1$, $MF1$} = {{*addr, pno, name*}, {$1m1$, $1m2$}}

P = {$AF2$, $MF2$} = {{*age, sex*}, { }}

• EMPLOYEE 클래스의 분할

$E1$ = {$AF1$, $MF1$} = {{*sal, eno, dpt*}, {*2m1, 2m2*}}

$E2$ = {$AF2$, $MF2$} = {{*cmp, married*}, { }}

• DEPARTMENT 클래스의 분할

$D1$ = {$AF1$, $MF1$} = {{*memNo, comp*}, { }}

$D2$ = {$AF2$, $MF2$} = {{*budget, dname*}, {$3m2$}}

$D3$ = {$AF3$, $MF3$} = {{*manager, member*}, {$3m1$}}

PERSON					
애트리뷰트	addr	pno	name	age	sex
addr	185	145	125	0	0
pno	145	210	150	0	0
name	125	150	235	85	45
age	0	0	85	125	85
sex	0	0	45	85	85

EMPLOYEE					
애트리뷰트	sal	eno	dpt	cmp	married
sal	135	110	70	0	0
eno	110	135	95	0	0
dpt	70	95	120	25	0
cmp	0	0	25	95	70
married	0	0	0	70	70

DEPARTMENT						
애트리뷰트	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 8) 클러스터 결합 행렬

	Person	Employee	Department
애트리뷰트 수	5	5	6
Ezeife	9×9	10×10	15×15
본 논문	5×5	5×5	6×6

(그림 9) 클러스터 결합 행렬의 크기

위와 같은 구현 결과를 타 논문과 비교하기위하여 본 논문의 구현 시에 사용된 클래스 스키마와 질의를 클래스 분할에 관한 관련연구인 Ezeife[2]의 연구에 적용 및 비교한 결과는 다음과 같다.

Ezeife 방법은 결합 행렬과 클러스터 행렬에 다른 클래스의 메소드가 포함됨으로 인하여 행렬의 크기가 커지며 불필요한 오버헤드와 복잡성이 증가된다((그림 9) 참조). 반면에 본 논문의 방법은 결합 및 클러스터 행렬과 분할 결과에 다른 클래스의 메소드와 애트리뷰트가 포함되지 않는다. 또한, Ezeife 방법은 중복되는 애트리뷰트를 한 프래그먼트에 할당시킴으로써 메소드는 있으나 그 메소드를 수행할 때에 필요한 애트리뷰트가 존재하지 않아 캡슐화에 위배되나 본 논문에서는 중복되는 메소드를 클래스 구성 계층 상의 부모 클래스에 배치시킴으로써 캡슐화를 유지시켜 준다. 또한, Ezeife 방법은 키 애트리뷰트와 메소드를 중복시키고 있으나 본 논문에서는 키 값을 중복시키지 않았으며 재구성을 할 때에는 객체 식별자를 이용한다.

5.2 할당의 구현

본 논문에서는 위에서 제시한 할당 목적 함수의 해를 찾기 위하여 유전자 알고리즘[11]을 사용하였으며 윈도우 PC에서 C언어로 구현하였다.

할당 목적 함수의 해를 구하는 문제는 일반적으로 NP-Complete로 알려져 있다. 이러한 문제를 해결하는데 있어서 유전자 알고리즘이 유용하게 사용될 수 있다. 유전자 알고리즘은 자연계을 진화과정을 지배하는 적자생존이나 유전정보의 교환에 의한 세대교체에 기반한 문제 해결 기법이다. 유전자 알고리즘은 단일 해가 아닌 해 집단(Solution Population)을 처리하며 이 해 집단의 반복적인 세대교체 과정을 통해 적응적으로 진화하면서 최적의 해를 탐색한다. 유전자 알고리즘의 수행과정은 다음과 같다.

1. 해 집단 생성
   1.1 각각의 프래그먼트가 최소한 한 노드 이상에 할당되도록 할당 해를 임의로 생성한다.
   1.2 일정한 개수의 해 집단이 이루어질 때까지 1.1의 과정을 반복한다.(해 집단의 크기는 입력 매개변수이다.)
2. 세대교체를 반복
   2.1 해 집단으로부터 적합도(fitness)가 높은 2개의 부모 해를 추출한다.
   2.2 상호교환(crossover)과 돌연변이(Mutation) 연산을 수행하여 새로운 할당 해를 생성한다.
   2.3 새로운 해가 해 집단의 최악의 해보다 우수하면 새로운 해를 해집단에 포함시키고 최악의 해는 제거한다.
   2.4 최악의 해가 최상의 해의 일정 퍼센트 이내가 될 때 까지 2.1에서 2.3을 반복한다.

질의	타입	참조 프래그먼트	발생 빈도	노드별 질의 발생빈도			
				노드 1	노드 2	노드 3	노드4
1Q1	검색	P1	400	300	0	100	0
1Q2	갱신	P1	600	400	200	0	0
1Q3	검색	P1, P2	450	0	350	100	0
1Q4	갱신	P1	450	150	0	300	0
1Q5	검색	P1, P2, D1	400	0	400	0	0
2Q1	갱신	E1	250	0	0	0	250
2Q2	검색	P1, E1, E2	400	400	0	0	0
2Q3	검색	E1, E2	250	250	0	0	0
2Q4	검색	P1, E1, E2	300	300	0	0	0
2Q5	갱신	E2	350	0	0	350	0
3Q1	검색	D1	400	300	100	0	0
3Q2	검색	E2, D1, D3	350	350	0	0	0
3Q3	갱신	D2	500	0	50	0	450
3Q4	갱신	D1	250	50	200	0	0
3Q5	검색	E1, D3	400	200	0	0	200
3Q6	검색	E1, D3	250	100	0	0	150

(그림 10) 노드별 검색 및 갱신 질의 발생빈도

클래스	프래그먼트		
Person(60K)	P1(40K)	P2(20K)	
Employee(50K)	E1(30K)	E2(20K)	
Department(80K)	D1(15K)	D2(25K)	D3(40K)

(그림 11) 클래스와 프래그먼트

프래그먼트	노드 1	노드 2	노드 3	노드 4
P1	X	X	X	
P2		X		
E1	X			X
E2	X		X	
D1	X	X		
D2				X
D3	X			X

(그림 12) 프래그먼트의 할당 결과

5.1절의 분할 구현의 모델과 결과를 바탕으로 할당을 구현하였으며 노드는 4개가 있는 것으로 가정하였다. 클래스와 프래그먼트에 관한 정보는 (그림 11)과 같으며 (그림 10)은 노드별로 검색 및 갱신 질의에 대한 발생빈도를 나타내고 있다.

(그림 10) 및 (그림 11)과 같은 구현 환경 하에서 유전자 알고리즘을 실행시킨 할당의 결과는 (그림 12)와 같다. 결과에서 볼 수 있듯이 한 프래그먼트가 여러 노드에 할당이 가능한 중복할당 지원하고 있다. (그림 10)에서 프래그먼트 P1, E1 및 E2는 노드 1에서 질의 2Q2, 2Q3, 2Q4에 의해 함께 접근되는데, (그림 12)의 결과를 보면 P1, E1 및 E2가 노드 1에 함께 할당되었다. 또한, 질의 3Q5과 3Q6에 의하여 같이 참조되고 있는 프래그먼트 E1과 D3도 노드 1, 노드 4에 같이 할당되어 할당결과가 합리적으로 나왔음을 알 수 있다.

6. 결 론

본 논문에서는 인터넷상의 대용량 자료에서 원하는 정보를 검색하기 위한 지원기능으로서 분산 객체지향 데이터베이스에서 데이터를 분산시키는 방법에 관하여 연구하였다.

설계된 분산기법에서는 응용 질의를 기초로 하여 클래스를 분할하는 분할단계와 분할된 프래그먼트를 여러 노드에 할당하는 할당단계로 구성된다. 분할단계에서는 비중복 다중 분할이 지원되도록 설계하였으며 객체지향의 상속이나 복합 객체 관계를 통하여 접근되는 다른 클래스의 애트리뷰트도 분할에 반영되도록 설계하였고 이것을 C 프로그래밍 언어를 사용하여 구현하였다. 타 분할 기법과의 비교에 의하면, 분할 행렬의 크기가 줄었고 불필요한 애트리뷰트와 메소드가 포함되지 않았으며 캡슐화 개념을 유지시켜 주는 것으로 나타났다.

분할된 프래그먼트를 여러 노드에 할당시키는 단계에서는 시스템 운영 비용을 최소화시키는 최적의 할당 노드를 찾기 위한 목적 함수를 기술하였으며 제시된 할당 목적 함수의 해를 구하기 위하여 유전자 알고리즘을 사용하여 구현하였다. 본 논문에서 제시한 할당 모델에서는 저장 비용, 질의처리 비용 및 전송 비용 요소를 고려하였으며 특히 전송 비용의 측정 시에는 검색 연산과 갱신 연산의 데이타 전송 방식이 서로 다른점을 반영하였다.

추후의 연구 과제로서 본 논문에서 제시한 분산 알고리즘의 효율성 분석에 대한 연구가 필요하다.

참 고 문 헌

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

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

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

[4] Karlapalem, K. and Li, Q. Partitioning Schemes for Object Oriented Database. In Fifth International Workshop on RIDE-DOM, 1995.

[5] Karlapalem, K., Li, Q. and Vieweg, S. Method Induced Partitioning Schemes in OODB. In 16th int'l conf. on DCS, Hong Kong, 1996.

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

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

[8] Navathe, S. B., Ceri, S. Wiederhold, G. and Dou, J, Vertical partitioning algorithms for database design. in ACM TODS, Vol.9, No.4, 1984.

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

[10] Salvatore T. March, Sangkyu Rho, "Allocating data and operations to nodes in distributed database design," IEEE Transactions on Knowledge and Data Engineering, 1995.

[11] David E. Goldberg, "Genetic Algorithms : in Search, Optimization & Machine Learning," Addison-Wesley, 1989.

[12] 이순미, "분산 데이타베이스에서 할당 알고리즘의 설계", 정보처리학회 춘계학술발표대회논문집, 제10권 제1호, 2003.

이 순 미

e-mail : leesm@kic.ac.kr

1984년 이화여자대학교 수학과(학사)

1986년 이화여자대학교 대학원 수학과 전자계산 전공(석사)

1997년 홍익대학교 전자계산학과(박사)

1998년~현재 경인여자대학 컴퓨터정보기술학부 교수

관심분야 : 객체지향 데이터베이스, 분산 시스템 등

박 혜 숙

e-mail : edpsphs@kic.ac.kr

1991년 고려대학교 산업공학과(학사)

1993년 고려대학교 대학원 산업공학과 (석사)

1999년~현재 고려대학교 대학원 컴퓨터학과 박사과정

2002년~현재 경인여자대학 컴퓨터정보기술학부 전임강사

관심분야 : 멀티미디어, 데이터베이스, 소프트웨어공학

하 얀

e-mail : white@kic.ac.kr

1992년 덕성여자대학교 전산학과(학사)

1994년 이화여자대학교 전자계산교육전공 (석사)

2000년 전북대학교 대학원 전산통계학과 (이학박사)

2000년~2001년 중앙대학교 정보통신연구소 연구전담교수

2001년~현재 경인여자대학 컴퓨터정보기술학부 전임강사

관심분야 : XML 응용, 객체지향 모델링, 컴포넌트 모델링, 애니메이션, 멀티미디어