분산 데이타베이스 설계를 위한 데이타 분할 방법론

A Data Partitioning Methodology for Distributed Database Design

박성진, 백두권
전산과학과, 고려대학교

Seong-Jin Park, Doo-Kwon Baik
Computer Science Department, Korea University

요 약

최근들어 분산 데이타베이스 시스템에 관한 연구가 연구 목적뿐만 아니라 실질적인 응용 분야에서도 급속히 증가하고 있다. 한편, 분산 데이타베이스 시스템에서 데이타 분할은 주요한 설계 이슈중의 하나이다. 비록 데이타 단편화 설계에 관한 많은 연구가 진행되었지만 기존 연구의 대부분은 일시적인 해결책이나 특별한 경우에 대한 일반적인 해결책에 그치고 있으며 단순히 수평 혹은 수직 분할 방법론만을 선택적으로 적용하고 있다. 본 논문에서는 혼합 단편화에 기반한 분할 방법론을 제안하였다. 제안한 방법론은 후보 단편들을 생성하고 데이타 할당과 중복을 위한 최적의 단위를 제공한다.

1. 서 론

분산 데이타베이스 시스템은 중앙집중식 데이타베이스 시스템에 비교해서 유용성(availability)의 증가, 접근 시간의 감소, 확장의 용이성, 기존 데이타베이스의 통합 용이성 등의 이점을 갖고 있다. 하지만, 이러한 이점들은 최적의 분산 설계(distributed design)를 통해서만 제공될 수 있다. 분산 데이타베이스 시스템은 전역 스키마(global schema)를 지역 스키마(local schema)로 변환하는 분산 설계를 요구하며 데이타 분산(데이타 분할 및 할당)은 분산 데이타베이스 시스템의 구축을 위한 기반이 된다. 그러나, 단편화(fragmentation) 및 데이타 할당(allocation)은 최적의 설계를 위해 고려해야할 요소가 너무 많기 때문에 매우 어려운 문제중의 하나로 간주되고 있다. 특히, 데이타 단편화 문제는 데이타 할당을 위한 최적의 단위를 제공하는 최적화 문제이며 이것은 분산 설계를 위한 기초로서 중요한 연구 분야중의 하나이다.

즉, 분산 설계는 단편화, 할당 및 중복 등의 서브 문제들을 갖는 원거리 트랜잭션 처리 비용을 최소화하는 최적화 문제라 할 수 있다. 전역 최적화 문제는 탐색 공간을 감소시키고 각 서브 문제

들의 복잡성을 감소시키기 위한 소규모의 최적화 문제로 분할이 가능하며 따라서, 각 서브 문제들을 결합한 새로운 방법론이 요구되고 있다.

본 논문에서는 최적의 데이타 분할(partitioning)을 제공하는 분할 방법론을 제안하였다. 제안한 방법론은 수평 및 수직 분할을 혼합한 혼합식 모델에 기반하고 있다.

2. 데이타 분할

분산 데이타베이스의 설계는 데이타 분할, 할당, 중복 및 지역 최적화 등의 상호 밀접하게 관련된 다양한 문제들에 대한 해결책을 요구하는 최적화 문제이다. 각 문제들은 서로 다른 다양한 접근을 통해 해결될 수 있으며 이는 분산 데이타베이스 설계를 매우 어렵게 한다. 전통적으로, 데이타 분할을 포함하는 데이타베이스 설계는 본질적으로 휴리스틱하다.

2.1 데이타 단편화

데이타 단편화(혹은 분할)은 논리적 객체(릴레이션)를 데이타베이스의 논리적 스키마로부터 저장된 데이타베이스의 여러 물리적 객체(화일)로 분할하는 과정으로 정의할 수 있다[5]. 데이타 단편화에는 기본적으로 두 가지 방법이 존재하는데 수직 분할(vertical partitioning)과 수평 분할(horizontal partitioning)이 그것이다.

수직 분할은 애트리뷰트들을 다수의 그룹으로 분리하는 작업으로 분할뒤엔 동일한 데이타를 접근하기 위하여 각 단편들은 전역 객체의 키 애트리뷰트를 포함해야한다. 수직 분할에 관한 기존 연구는 분할을 위한 목적 함수(object function)를 사용해왔다[3,5]. 또한, 이러한 접근 방법이외에도 이진 분할(binary partitioning) 기법이 재귀적으로 적용되어왔으며 목적 함수와 보상 알고리즘이 요구되었다. 그리고, [7]에서는 한번에 모든 의미있는 수평 분할을 생성해내는 그래프 이론 알고리즘(Graph theoretic algorithm)을 제안하였다.

수평 분할은 릴레이션의 튜플들을 다수의 그룹으로 분리하는 작업으로 각 단편들은 특정 단편들의 인스턴스 혹은 튜플들을 적절히 차별화시키는 프레디키트(predicate)와 연관되어 있다. 이전 연구의 대부분에서는 너무 많은 분할이 발생하여 최악의 경우 오직 하나의 튜플만을 갖는 과다한 수평 분할이 생성될 수 있다는 문제점을 지니고 있다. 이러한 이유때문에 프레디키트 클러스터링을 사용하는 새로운 수평 분할 기법이 활발하게 연구되고 있다[8].

2.2 혼합식 단편화

한편, 혼합식 분할(mixed partitioning)을 생성하기위하여 두 종류의 단순 분할 방법들의 혼합이 고려되었다. 데이타베이스 사용자는 일반적으로 전역 릴레이션의 수직 및 수평 단편들의 일부분을 접근하기 때문에 혼합식 분할에 대한 요구가 분산 데이타베이스 설계에서 대두되었다[4]. 특히, 혼합 분할에 관한 기존 연구의 예가 [1]에 제시되어 있다.

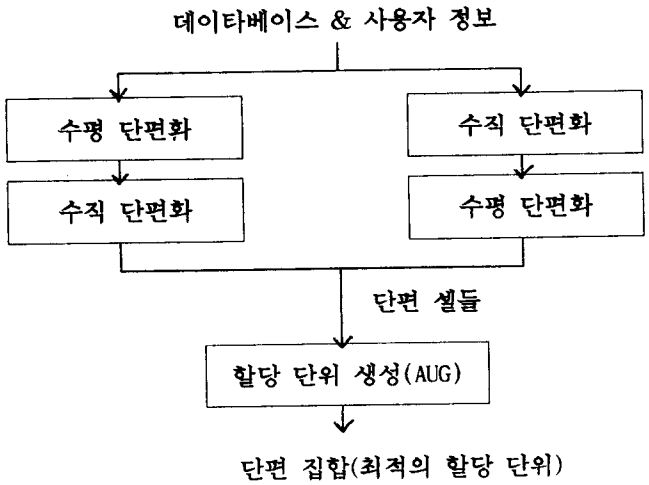
대부분의 경우에 데이타베이스 스키마에 대한 단순한 수평 혹은 수직 단편은 사용자의 응용을 위한 요구 사항을 충분하게 만족시키지 못한다. 따라서, 혼합식 단편화 기법이 데이타 분할을 위

해 바람직하며 본 논문에서도 이 방안을 기반으로 하였다.

3. 제안한 데이타 분할 방법론

3.1 분할 방법론

데이타 할당을 위한 혼합식 단편들의 생성은 다음과 단계를 거쳐 이루어진다. <그림 1>은 본 논문에서 제안하는 혼합식 분할 방법론(mixed partitioning methodology)을 보여준다.



<그림 1> 혼합식 분할 방법론

3.2 사용자 입력 정보 명세

혼합식 단편화를 생성하기 위해서는 다음과 같은 입력 정보가 사용자로부터 제공되어야한다.

a) 데이타베이스 정보 : 릴레이션, 애트리뷰트, 카디널러티, 애트리뷰트 크기, 등.

b) 사용자 정보 : 트랜잭션 빈도, 트랜잭션 타입, 애트리뷰트 사용도, 프레디키트 사용도,등.

<그림 2>에서 애트리뷰트 사용도 행렬(attribute usage matrix)은 트랜잭션들을 행에, 애트리뷰트들을 열에 포함하고 있다. 만약 트랜잭션 i가 애트리뷰트 j를 사용한다면 요소 (i,j)가 1 이며 그렇지않으면 0 이다. 프레디키트 사용도 행렬도 마찬가지이다. 본 논문에서는 가장 빈번하게 사용되는 사용자 질의의 20%가 전체 데이타 접근량의 80%를 차지한다는 80/20 규칙을 가정하였다.

애트리뷰트 사용도 행렬											단위 시간당 접근 횟수
애트리뷰트	1	2	3	4	5	6	7	8	9	10	
트랜잭션											
t1	1	0	0	0	1	0	1	0	0	0	25
t2	0	1	1	0	0	0	0	1	1	0	50
t3	0	0	0	1	0	1	0	0	0	1	25
t4	0	1	0	0	0	0	1	1	0	0	35
t5	1	1	1	0	1	0	1	1	1	0	25
t6	1	0	0	0	1	0	0	0	0	0	25
t7	0	0	1	0	0	0	0	0	1	0	25
t8	0	0	1	1	0	1	0	0	1	1	15

애트리뷰트 사용도 행렬

프레디키드 사용도 행렬									단위 시간당 접근 횟수
프레디키트	1	2	3	4	5	6	7	8	
트랜잭션									
t1	1	0	0	0	0	0	1	0	25
t2	0	1	0	0	0	0	1	0	50
t3	0	0	1	0	0	0	1	0	25
t4	0	1	0	1	0	0	0	1	35
t5	0	0	0	0	1	0	0	1	25
t6	0	0	0	0	0	1	0	1	25
t7	0	0	0	0	1	0	0	1	25
t8	0	0	0	0	0	1	0	1	15

프레디키트 사용도 행렬

트랜잭션 타입 : (t1,w) (t2,w) (t3,r) (t4,r) (t1,w) (t6,r) (t7,w) (t8,w) (t9,r) (t10,r)

<그림 2> 사용자 입력 정보 명세

3.3 혼합식 분할

혼합식 분할 방법론은 수평 분할 기법과 수직 분할 기법이 복합된 형태이다. 오늘날의 일반적인 방법론은 오직 다음과 같은 두 가지 방법중 하나를 통해서 혼합 분할을 생성할 수 있다. 즉, 수직 분할뒤 수평 분할을 수행하든지, 수평 분할뒤 수직 분할을 수행함에 의해서 분할을 생성할 수 있다. 이러한 방법론은 수직 및 수평 분할 알고리즘을 릴레이션에 독립적으로 적용함에 의해 각 셀이 생성되며 수평 및 수직 분할을 위한 효율적인 알고리즘은 이미 [6,8]에서 제시되었다.

3.4 트랜잭션 매핑

먼저, 트랜잭션 매핑(transaction mapping)은 트랜잭션의 애트리뷰트와 프레디키트를 셀(cell)을 생성하는 애트리뷰트와 프레디키트와 연관시킴에 의해 생성된다. <그림 3>은 단편 셀들에 대한 트랜잭션 매핑을 보여준다. 이러한 매핑 정보에 기반해서 셀들은 전체 트랜잭션 처리 비용을 최소화하기 위하여 합쳐진다.

	v1(a1,a5,a7)	v2(a2,a3,a8,a9)	v3(a4,a6,a10)
h1(p3,p4,p6,p8)	t4,t6	t4,t8	t8
h2(p5,p8)	t5	t5,t7	
h3(p1,p2,p7)	t1	t2	
h4(p3,p4,p7)			t3
h5			

<그림 3> 트랜잭션 매핑

수직 및 수평 분할 알고리즘을 독립적으로 적용함에 따른 생성되는 각 셀은 각기 다른 결과를 생성할 가능성이 있고 더우기 효율적인 데이타 분산을 위한 더 작은 단위의 셀들을 생성할 가능성을 배제함으로써 적절하지 못하다. 따라서, 본 논문에서는 전역 트랜잭션 처리 비용을 최소화하기 위해 셀들을 최적으로 생성시키는 혼합식 분할 방법[8]을 적용하였다.

이 단계에서 단편셀들은 전역 트랜잭션 처리 비용을 최소화하기 위하여 합쳐진다. 그에 대한 결정을 위한 비용 모델(cost model)과 합칠것인지의 여부 및 방법들에 대해서는 [8]에서 제시되었다. 따라서, 본 논문에서는 트랜잭션 타입을 가중치로 하는 수정된 비용 모델을 제안하여 적용하였다. 수정된 비용 모델은 다음과 같다.

N : number of tansactions
f : fragment(=set of fragment cell)
afr_k : access frequency for k-th transaction
w_k : weight factor for k-th transaction
pfr_k : project frequency for k-th transaction
sfr_k : select frequency for k-th transaction
jfr_k : join frequency for k-th transaction
ufr_k : union frequency for k-th transaction
v_m : vertical fragment
h_n : horizontal fragment
W_j : join weight
Wu : Union weight
$f_i \cup f_j$: combined fragment by meging fragment f_i, f_j
$acc\_c(f)_k$: cost of accessing fragment f for k-th transaction
$prj\_c(f)_k$: cost of projection fragment f for k-th transaction
$sel\_c(f)_k$: cost of selection fragment f for k-th transaction
$jon\_c(f_i,f_j)_k$: cost of join fragment f_i,f_j for k-th transaction
$uni\_c(f_i,f_j)_k$: cost of union fragment f_i,f_j for k-th transaction

위의 수정된 비용 모델에서는 갱신 트랜잭션과 판독 트랜잭션 타입을 구별하기 위해 트랜잭션 처리 비용에 가중치 'w_k'를 적용하였다. 가중치 'w_k'는 트랜잭션 타입이 판독(read)이면 1, 트랜잭션 타입이 갱신(update)이면 α이다.($\alpha > 1$) 가중치를 처리 비용에 곱함으로써 갱신 트랜잭션의 처리 비용을 반영하여 비용을 증가시킬 수 있다. 처리 비용 관점에서 보았을 경우 이것은 매우 중요하다.

hm_S_k : saving for k-th transaction by merging fragments(f_i,f_j) horizontal

$$= w_k * \{afr_k * acc\_c(v1) + afr_k * acc\_c(v2) + jfr_k * jon\_c(v1,v2) * W_j\} - w_k * \{afr_k * acc\_c(v1 \cup v2) + pfr_k * prj\_c(v1 \cup v2)\}$$

vm_S_k : saving for k-th transaction by merging fragments(f_i,f_j) vertical

$$= w_k * \{afr_k * acc\_c(h1) + afr_k * acc\_c(h2) + ufr_k * uni\_c(h1,h2) * Wu \}$$
$$- w_k * \{afr_k * acc\_c(h1 \cup h2) + sfr_k * sel\_c(h1 \cup h2)\}$$

$$merging\_saving = MAX(\sum^{N}_{k=1} hm\_s_k , \sum^{N}_{k=1} vm\_s_k)$$

각 순환동안 합쳐지는 수직(수평) 단편들의 인접 수평(수직) 단편들은 그것들을 분리해두는 것보다 하나로 합침에 의해 최대의 비용 절감을 가져오는 쌍이다. 본 논문에서는 합침에 따른 더 이상의 비용 절감이 없을때까지 수평 혹은 수직으로 인접한 단편들을 계속적으로 합침을 계속한다.

3.5 할당 단위 생성(AUG:Allocation Unit Gernation)

분산 데이타베이스의 경우에 트랜잭션 처리 비용은 원거리 데이타 항목에 대한 접근 횟수를 줄임에 의해서 즉, (특정 사이트에서의) 지역 트랜잭션 처리(local transaction processing)를 증가시킴에 의해 최소화될 수 있다. 만약 생성된 단편들이 제공되는 트랜잭션 집합의 요구를 적절히 만족시키면 트랜잭션 처리 비용은 최소화될 수 있다. 데이타 분할 기법의 목적은 위와 같은 목적을 만족하는 분할 구조를 찾는 것이다. 그러나, 분할 과정동안 생성된 비오버래핑(nonoverlapping) 데이타 단편들만을 고려하는 분할에 관한 기존 연구는 가능한한 약간의 중복만을 허용한채 다른 사이트에 할당한다. 셀들을 합침에 의해 생성되는 디스조인트(disjoint) 단편들은 데이타 분할을 위해서는 적합하지만 데이타 중복을 위해서는 적합하지 않으며 위와 같은 목적을 만족할 수 없다. 따라서, 분할 과정에서 단편들의 오버래핑 혹은 중복이 고려되어야만 한다. 이것은 최적의 분산 설계를 위해서 매우 중요하다. 만약 프로젝트 비용(project cost)을 갖는 단편들이 중복되면다면 그 단편들안의 비관련 애트리뷰트들의 갱신 비용이 증가하며, 만약 죠인 비용(join cost)을 갖는 단편들이 중복된다면 관련 애트리뷰트들의 원거리 접근 비용이 증가하게 된다.

따라서, 본 논문에서는 단위 생성 단계를 제안하였다. 이 단계는 데이타 단편들이 애트리뷰트를 중복해서 갖는 것을 허용하면서 최적의 할당 단위를 위한 후보 단편(candidate fragment)들을 생성한다.

본 논문에서는 할당 모델로부터 요구되는 모든 후보 단편들이 명세되며 할당 모델은 후보 분할 중 하나를 선택한다고 가정한다. 최적화 모델은 선택적인 후보 분할 중 하나를 결정하고 객체에 그것을 적용한뒤 계속해서 그 단편의 할당을 결정한다. <그림 4>에서 (a)는 단위 생성 단계의 가능한 하나의 결과를 보이며 (b)는 데이타 분할의 최적 사이트 매핑을 보인다.

f1	f11	f2
f3	f31	
f4	f5	
		f6

(a) 후보 단편들

physical site		candidate fragments
site1(t1,t4,t7)	->	f4,f11,f2,f31
site2(t2,t5,t8)	->	f5,f3,f2
site3(t3,t6)	->	f6,f1

(b) 사이트 매핑

<그림 4> 후보 단편 집합

4. 결 론

본 논문에서는 혼합식 단편화에 기반한 분할 방법론을 제안하였다. 혼합식 분할 기법에 기반한 방법론은 다음과 같은 3 단계로 수행된다. : 사용자 입력 정보 명세, 분할, 할당 단위 생성. 제안된 분할 모델은 비용 모델에서 트랜잭션 타입을 가중치로 고려함으로서 데이타 분할의 효율성을 증가시킬 수 있다. 또한, 이러한 분할 모델은 데이타 할당과 중복을 위한 최적의 단위를 제공하기 위한 후보 단편들을 생성한다.

5. 참고 문헌

[1] Apers, P. M. G., "Data Allocation in Distributed Database Systems," ACM Trans. on Database Systems, Vol. 13, No. 3, pp.263-304, Sept, 1988.

[2] Ceri, S., Navathe, S. B., and Wiederhold, G., "Distribution Design of Logical Database Schema," IEEE Trans. on Software Engineering, Vol. SE-9, No. 4, pp.487-504, July 1983.

[3] Cornell, D. W., and Yu, P. S., "On Optimal Site Assignment for Relations in the Distributed Data Environment," IEEE Trans. on Software Engineering, Vol. 15, No. 8, pp.1004-1009, Aug. 1989.

[4] Elmasri, R., and Navathe, S. B., Fundamentals of Database Systems, Benjamin/Cummings Publishing, 1989.

[5] Navathe, S. B., Ceri, S., Wiederhold, G., and Dou, J., "Vertical Partitioning Algorithms for Database Design," ACM Trans. on Database Systems, Vol. 9, No. 4, pp.690-710, Dec. 1984.

[6] Navathe, S. B., and Ra, M., "Vertical Partitioning for Database Design : A Graphical Algorithm," Proc. ACM SIGMOD International Conference on Management of Data, pp.440-450, May 1989.

[7] Ra, M., "A Graph-based Horizontal Partitioning Algorithm for Distributed Database Design," Journal of the Korea Information Science Society, Vol. 19, No. 1, 1992.

[8] Ra, M., Park, Y. S., "Data Fragmentation and Allocation for PC-based Distributed Database Design," Proc. Database Systems for Advanced Applications '93, Apr. 1993.