論文 98-23-9-29

분산 데이터베이스 시스템에서 질의 이동을 사용한 실용 질의 처리 전략

정회원 이 형 묵*, 김 경 창**

A Simplified Query Processing Strategy using Query Passing in Distributed Database

Hyung-Mook Lee*, Kyung-Chang Kim** *Regular Members*

요 약

본 논문에서는 분산 데이터베이스에서의 실용 질의 처리 전략을 제안한다. 구현하기 쉽고 분산 데이터베이스의 하나인 system R*와 유사한 질의 처리 성능을 지니는 것이 본 논문에서 제안되는 실용 질의 처리 전략이다. 질의 처리 전략들의 성능을 평가하기 위하여 질의 처리 비용을 구하는 비용 모델을 정의하였다. 정확한 실험을 위하여 Wisconsin benchmark에서 사용된 실험용 데이터베이스와 질의들을 성능 평가 실험에 사용하였다. 이를 기반으로 실험용 질의에 다양한 인자를 적용시켜 질의 처리 비용 계산을 실시하여 질의 처리 전략들의 정확한 성능 평가를 실시하였으며 비용 계산 결과를 가지고 제안된 실용 질의 처리 전략의 타당성을 입증하였다.

ABSTRACT

In this paper, a simplified query processing strategy in distributed databases is proposed. A simplified query processing strategy is defined as a query processing strategy which is easier to implement and yet whose performance is comparable to system R* strategy. A cost model is defined to compute the query processing cost to evaluate the performance of the proposed query processing strategy. For a performance evaluation, the cost is computed using a test database and test queries based on the Wisconsin benchmark. Finally, the validity of the proposed query processing strategy is assessed based on the results of the performance evaluation.

Ⅰ. 서 론

분산 데이터베이스(Distributed DATABASE)란 데이터베이스가 하나의 물리적 위치에 저장되어 있지 않으며 지리적으로 흩어져 있는 지역에 따로 따로 존재하고 통신시스템을 통하여 서로 연결되어 있는 것이다.[1,2] 분산 데이터베이스 관리 시스템(distributed DBMS)이란 분산된 데이터 베이스들을 관리하고 사용자들에게 데이터 분산의 투명성을 제공해 주는 소프트웨어 시스템이다.

질의(Query) 처리의 의미는 질의들을 분석하여 데이터를 조작할 수 있는 연산의 순차집합으로 변환시키고 이들을 수행한다는 것이다. 분산 질의 처리의 문제점은 각각의 질의에 대하여 네트워상을 통하여

*홍익대학교 대학원 전자계산학과
**홍익대학교 컴퓨터공학과
論文番號: 97362-1009
接受日字:1997년 10월 10일

가장 비용 대비 성능이 우수한(cost-effective) 실행 전략을 구하는 것이 어렵다는 것이다.[3] 분산된 질의 처리에서는 데이터의 분산현황, 이들에의 접근방법, 네트웍상에서의 전송 비용 등 고려되어야 할 사항들이 많기 때문이다.

효율적인 질의 처리를 위해서는 질의 최적화(query optimization)가 필요한데 질의 최적화란 주어진 질의를 처리하는데 필요한 전체 비용(total cost)이 가장 적은 질의 실행 방법(query execution plan)을 선택하는 과정이다. 전체 비용은 네트웍상의 모든 사이트(site)에서 질의 처리를 위해 발생하는 비용과 사이트간 통신 비용의 합이다. 이 전체 비용의 계산에는 CPU, I/O, 전송비용 등이 포함된다.

앞에서 서술한 비용 계산 과정을 거친 다양한 질의 처리 방법들 중에서 하나가 최적의 질의 실행 계획으로 선택되어 진다. 비용 계산을 위한 인자로는 데이터베이스 통계자료, 테이블(Table, Relation)의 죠인(Join) 순서, 연산과정의 중간 결과물 크기 등 매우 다양하다. 본 논문에서는 죠인 순서와 사이트간의 통신 비용에 초점을 맞추었는데 왜냐하면 이 두 비용이 성능에 큰 영향을 미치기 때문이다.

System R*[4, 7]에서의 질의 최적화 알고리즘은 System R[6]의 알고리즘을 분산 환경에 맞게 확장한 것인데 계산이 복잡하고 구현이 어렵다는 단점이 있다.

본 논문의 목적은 system R*에서 사용된 알고리즘보다 구현이 단순하면서도 성능은 system R*와 비교될 만한 실용 질의 최적화 전략을 제안하는 것이다. 본 논문의 구성은 2장에서는 기존의 질의 처리 전략에 대하여 알아보고 3장에서는 제안된 질의 처리 전략을 설명하였으며 4장에서는 질의 처리 비용 계산을 위한 각종 실험 환경에 대하여 설명하였다. 5장에서는 질의 처리전략에 대한 시뮬레이션 결과를 보였고 6장에 결론을 기술하였다.

Ⅱ. 분산 데이터베이스에서 기존의 질의 처리 전략

분산 데이터베이스 환경에서 질의를 처리하기 위한 전략은 크게 두 가지로 나누어 볼 수 있는데 첫번째는 Master-Slave 질의 처리 전략이고 두번째는 Peer-to-Peer 질의 처리 전략이다.

2.1 Master-Slave 질의 처리 전략

질의가 들어온 site가 Master site가 되며 질의를 해결하기 위하여 참여하여야 할 다른 site들을 Slave site라 한다. Master-Slave 질의 처리 전략의 특징은 Master site가 질의를 처리하는 방법을 결정할 때 Slave site들간의 통신을 고려하지 않고 오직 Master site와의 통신만을 고려한 질의 처리 방법을 작성한다는 것이다. 이 전략의 장점은 Master site가 세울 수 있는 질의 처리 방법의 가지 수가 단순하여 실제 시스템 구현이 쉽다는 것이며 단점은 Slave site들간의 통신이 가능할 경우 얻어질 수 있는 질의 처리 방법이 만들어 지지 않아 질의 처리 능력이 떨어 질 수 있다는 것이다.

2.2 Peer-to-Peer 전략

질의가 들어온 site가 Master site가 되며 질의를 해결하기 위하여 참여하여야 할 다른 site들을 Apprentice site라 한다. Peer-to-Peer 질의 처리 전략의 특징은 Master site가 질의를 처리하는 방법을 결정할 때 Apprentices site들간의 통신도 고려한 방법을 포함시킨다는 것이다. 즉, Master site와 Apprentices site들, 한 Apprentice site와 다른Apprentice site들, 이 두 가지 통신을 모두 고려한 방법을 작성한다는 것이다. 이 전략의 장점은 Master site가 작성 할 수 있는 질의 처리 방법의 가지수가 증가되어 최적의 방법이 선택되어 질 수 있다는 것이며 단점은 많은 방법을 세우고 이 가운데 가장 좋은 방법을 선택하는 등 질의 처리 방법을 구하는 실제 구현 과정이 복잡해져서 실제 시스템 구현이 어렵다는 것이다.

Ⅲ. 새로운 질의 처리 전략의 제안

Master-Slave 질의 처리 전략의 장점은 실제 구현이 간단하다는 것이며, 단점은 Slave site들간의 통신이 고려되지 않아 Slave site들간의 통신이 필요한 질의의 경우에는 Master site를 통해서 다른 Slave site와 통

신해야하므로 통신 비용이 증가한다는 것에 있다. 이러한 단점을 보완하기 위하여 질의를 이동시키는 새로운 전략을 제시한다. 제안된 질의 처리 전략의 명칭은 확장된 Master-Slave 질의 처리 전략 (Enhanced Master-Slave)이라 칭한다.

제안된 질의 처리 전략의 핵심은 질의가 들어온 site가 Master site가 되나 필요할 경우는 질의를 다른 site로 이동시켜 그 site로 하여금 임시 Master site (Temporary Master site)가 되어 질의를 처리하게 하고 질의 처리 후 그 결과를 원래 Master site로 반환하게 하는 것이다. 이렇게 함으로써 Master-Slave 질의 처리 전략에서도 필요할 경우는 site간의 자료 이동이 가능하게 된다. 실험에 사용된 질의를 가지고 제안된 질의 처리 전략을 설명해 보면

```
select *
from R1, R2
where R1.a = R2.a
```

'R1' table은 site 'S1' 에 'R2' table은 site 'S2' 에 있으며 site 'M' 에 질의가 들어왔고 'S2' site에서 join 한다면

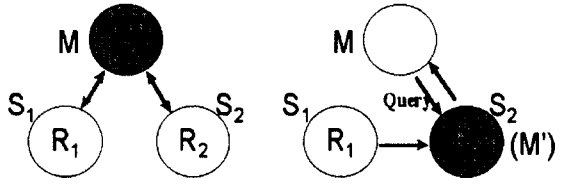


그림 1. 제안된 질의 처리 전략
Fig 1. Proposed Query Processing Strategy

□ 기존의 Master-Slave 방식 [그림 1. (a)]
가. 'S1' site의 'R1' 테이블을 'M' site로 옮긴다.
나. 'M' site의 'R1' 테이블을 'R2' site로 옮긴다.
다. 'S2' site에서 join 한다.
라. 결과를 'M' site로 옮긴다.

□ 제안된 질의 처리 방식 [그림 1. (b)]
가. 'M' site의 질의를 'S2' site로 옮긴다.
나. 'S1' site의 'R1' 테이블을 'S2' site로 옮긴다.
다. 'S2' site에서 join 한다.
라. 결과를 'M' site로 옮긴다.

제안된 질의 처리 전략은 기존의 Master-Slave 질의 처리 전략에서 두 번에 걸쳐 'R1' 테이블을 옮기는 과정을 한번으로 줄임으로써 통신의 양을 대폭 줄이게 되며 Peer-to-Peer 질의 처리 전략과 유사한 성능을 지닐수 있게 된다. 즉, 질의를 처음 받은 Master site가 질의처리에 유리한 'S2' site로 질의를 재전송함으로서 'S2' site가 Master site처럼 'S1' site와 통신을 하여 질의처리를 하고 결과 테이블은 원래의 Master site로 보내는 것이다.

결론적으로 하나의 질의 전송만으로 Peer-to-Peer 질의 처리 전략과 같은 성능을 내는 것이다. 또한 Master-Slave 질의 처리 전략의 기본 개념을 그대로 유지한 채 Master site만을 바꾸어 주는 원리이므로 생성될 수 있는 질의처리 방법의 수도 Peer-to-Peer 질의 처리 전략 보다 적어진다. 물론 처음 Master에 있는 질의를 임시 Master site로(즉 'M' 에서 'S2' site로) 옮기는 비용이 필요하기는 하지만 질의를 하나 옮기는 비용은 join에 필요한 테이블을 옮기는 비용에 비하면 거의 무시할 만 하다. 실제 질의 처리 성능의 비교 결과는 5장에서 보도록 한다.

Ⅳ. 질의 처리 비용 계산

4.1 비용 인자

집중화된 데이터베이스에서의 질의 처리 비용 계산에서는 두 가지의 중요 요소 즉, I/O 동작의 수행 횟수와 CPU의 사용량을 기준으로 하였으나 분산 데이터베이스 환경에서는 집중화된 데이터베이스의 두 가지 요소 이외에 site간의 자료 전송 비용이 질의 처리 비용에 추가 되어야 한다. 점진적인 Network 기술의 발전으로 초당 자료의 전송량이 증가 추세이기는 하나, 한 Site 내에서 Disk간의 자료전송 이나 Main Memory상에서의 자료의 처리 속도에 비하면 아직도 많은 시간을 필요로하기 때문에 자료의 전송 비용이 전체 질의 처리 비용에서 차지하는 효과는 크다고 할 수 있다.

4.2 비용 모델

질의 처리 비용을 계산하기 위한 방법으로 질의를 처리하는데 소요된 모든 자원의 합을 구하는 total cost 측정 방법을 사용하였다. 3.1에서 언급한 바와 같이 전체 질의 처리 비용은 site에서 처리되는 Local cost와 site간의 통신비용을 합한 것이 되며 다음의 식으로 나타내어 진다.

```
TOTAL_COST = LocalCost + CommunicationCost
LocalCost = I/O_Cost + CPU_Cost
I/O_Cost = I/O_WEIGHT * Number_Of_Pages_Fetch
Number_Of_Pages_Fetch= Number_Of_INDEX_Page
                     + Number_Of_DATA_Page
CPU_Cost = CPU_WEIGHT *
                     Number_Of_CPU_instruction
CommunicationCost = MESSAGE_WEIGHT *
                  Number_Of_MESSAGE_SENT +
                     BYTE_WEIGHT *
                  Number_Of_BYTES_SENT
```

Local cost는 각 site에서 질의를 처리하는데 필요로 하는 비용이며 Communication cost는 site간의 자료 전송비용이다. Local cost는 I/O cost와 CPU cost의 합으로 나타낼 수 있는데 I/O cost는 질의를 처리하기 위해 필요로하는 자료의 Input/Output의 양을 의미하며 CPU cost는 질의를 처리하기 위한 instruction의 수로 정량화 할 수 있다. site간의 자료 전송 비용은 Message 전송 비용과 Data 전송 비용의 합으로 구성된다. 질의 처리 비용을 계산하기 위한 분산 데이터베이스 실험 환경을 다음과 같이 가정하였다.

- CPU : 0.868 × 10-6ms/instruction
- I/O : 19.2ms/4Kbyte
- Network : Msg:0.336ms, Data:0.0048ms
- 자료의 I/O를 측정하기 위한 Page size는 4Kbyte 이다.
- Index는 B-tree 구현 형태를 가진다.
- 다음과 같은 통계정보(Statistics)를 유지한다.

NCARD(T) : Relation T의 cardinality
TCARD(T) : Ralation T의 page 수
ICARD(I) : number of distinct keys in index I
NINDX(I) : the number of pages in index I
P(T) : the fraction of data pages in the segment that hold tuples of relation T

4.3 비용 계산을 위한 데이터베이스 및 질의어

질의 처리 전략을 비교하기 위한 논리적 데이터베이스 (Synthetic database) 및 질의어는 Wisconsin Bench-mark[5]를 기본으로 구성하였다. Wisconsin Benchmark는 Relation의 Tuple 개수를 OneK(1K Tuples), TenK(10K Tuples)로 구성하며 Relation은 16개의 attribute (13 integer, 3 string attribute)로 구성되어 하나의 Tuple이 182Byte의 크기를 갖게 구성하였으며 질의어는 Selection, Projection, 2-way, 3-way JOIN, aggregate, Update(insert, delete, update) 등 32개를 사용하였다.

본 논문에서는 기본적인 Relation의 구조는 Wisconsin Benchmark와 동일하게 구성했으나 Relation의 크기를 1K, 10K, 100K의 세가지로 구분하여 자료량의 증가에 따르는 차이를 명확이 보이려고 하였으며 실험용 질의어는 질의 처리 방법에 의해 비용의 차이가 나타나는 2-Way, 3-Way JOIN 질의를 사용하였다.

실험용 질의어의 예는 다음과 같다.

```
select *
from relation_1, relation_2
where (relation_1.field=relation_2.field) and
      (relation_2.field<1000);                    (질의1)
select *
from relation_1, relation_2, relation_3
where(relation_1.field=relation_2.field) and
   (relation_2.field=relation_3.field) and
   (relation_1.field<1000) and
   (relation_2.field<1000);                       (질의2)
```

각 질의에 사용된 Selectivity Factor는 0.01, 0.05, 0.1의 3가지 경우에서 실험하였으며 join 방법으로는 nested-loop, sort-merge방법을 사용하여 join방법에 따

른 실험 결과를 비교하였다. 또한 Relation의 전송 방법으로는 ship whole방법과 Fetch-as-need 방법의 두 경우에서 비교 실험하였다.

4.4 질의 처리 비용 계산

4.3에서 제시된 질의어들에 대하여 4.2 비용모델에서 제시된 계산식에 대입하여 Master-Slave, Peer-to-Peer 각각의 질의 처리 비용을 계산하였으며 Peer-to-Peer 질의 처리 비용의 성능을 100으로 할 때 Master-Slave 질의 처리 비용의 성능을 %로 나타내었다. 다음표는 2-way join에서 nested_loop join, ship whole 조건을 적용시의 비용 계산 결과 일부이다.

도표 1. 질의 처리 비용 계산 결과(일부)
Table 1. Query Processing cost

INDEX	R1	R2	Master-Slave Cost	Peer-to-Peer Cost	PP:MS %
NO INDEX	1K	1K	77870.52	70401.24	90.40
		10K	698263.06	694331.87	99.43
		100K	6902188.50	6902188	99.99
	10K	1K	772562.75	694331.87	89.87
		10K	6921205.50	6846513.00	98.92
		100K	68407632.00	68368320.00	99.94
	100K	1K	7719485.00	6933638.00	89.81
		10K	69150632.00	68368320.00	98.86
		100K	683462080.00	682715136.00	99.89
:	:	:	:	:	:

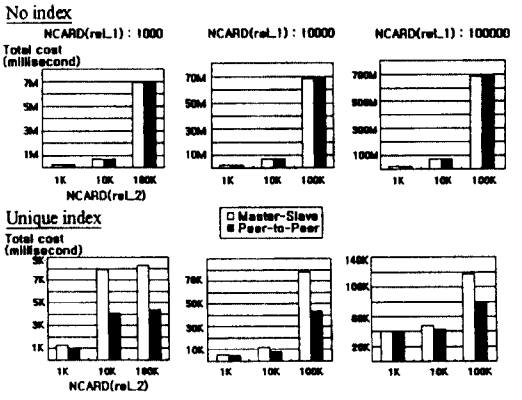
V. 시뮬레이션 결과

5.1 2-way joins

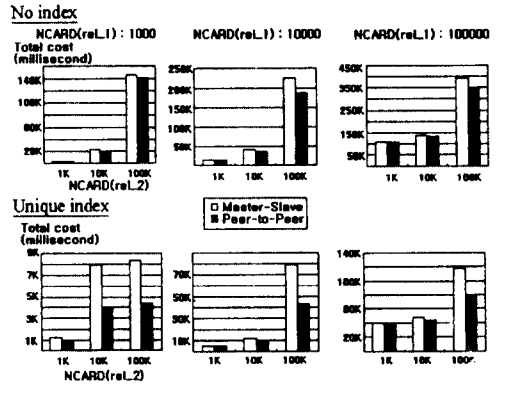
Master site에 join에 참여할 하나의 Relation이 있는, 2 site 만이 참여하는 2-way join에서는 Master-Slave 질의 처리 전략과 Peer-to-Peer 질의 처리 전략의 성능이 두 전략에서 생성되는 질의 처리 방법이 같으므로 동일하게 된다. 따라서 본 연구에서는 3 site가 참여하는 2-way join을 연구 범위로 하였다. 이 경우 Master site는 join에 참여할 어떤 Relation도 가지고 있지 않다.

그림 2는 2-way join을 수행하는 질의에 대한 결과로서 Master-Slave 질의 처리 전략과 Peer-to-Peer 질의 처리 전략의 성능을 Relation의 크기, 접근방법의 차이에 따라 나타내어 주고 있다.

Relation의 크기변화에 따른 실험 결과에서 알 수 있는 것은 Outer Relation과 Inner Relation의 크기 차이가 커질수록 Master-Slave, Peer-to-Peer 질의 처리 전략간의 성능차이가 줄어듦을 알 수 있다. 이것은 I/O 비용이 전송 비용의 차이에서 생기는 효과를 희석시킴으로써 나타나는 현상이다. 두 질의 처리 전략간의 차이는 전송 비용의 차이라고 볼 수 있는데 전체 비용에서 I/O 비용이 증가하므로 전송 비용의 효과가 줄어들었음을 의미한다.



(a) Master-Slave vs. Peer-to-Peer (nested-loop)

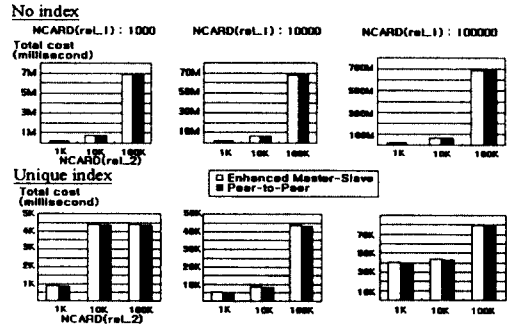


(b) Master-Slave vs. Peer-to-Peer (sort-merge)

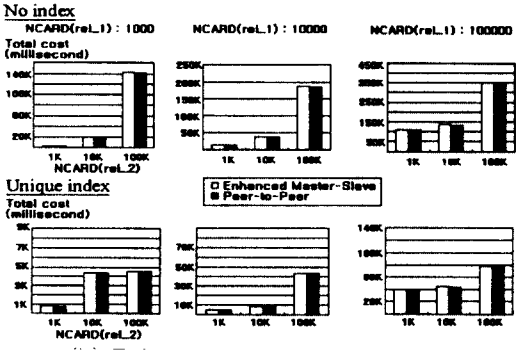
그림 2. 2-way join 실험결과
Fig 2. Simulation Results for 2-way join

no index방식의 경우 Master-Slave 질의 처리 전략의 성능이 Peer-to-Peer 질의 처리 전략의 성능과 거의 같다.(90%～100%) 반면에 unique index방식의 경우는 Relation의 크기에 다소 영향을 받지만 Master-Slave 질의 처리 전략의 전반적 성능이 Peer-to-Peer 질의 처리 전략에 비해 50%～98% 정도밖에 되지 않는다. 동일한 환경에서는 I/O 비용이 월등히 큰 noindex 방식에서 두 방식의 전송비용의 차가 희석되는 원리이다. 이것은 앞서 설명한 Relation의 크기 차이가 커질수록 성능차이가 다소 줄어드는것과 같은 원리이기도 하다.

제안된 질의 처리 전략의 성능을 평가하기 위하여 앞의 기존 질의 처리 전략의 실험과 같은 환경 변수와 같은 데이터베이스 및 같은 질의어를 가지고 실험하였다. 이렇게 실험된 결과는 기존 질의 처리 전략 중 우수한 성능을 보이는 Peer-to-Peer 질의 처리 전략의 성능과 비교하여 다음과 같은 그래프로 나타내었다.



(a) Enhanced Master-Slave vs. Peer-to-Peer (nested-loop)



(b) Enhanced Master-Slave vs. Peer-to-Peer (sort-merge)

그림 3. Enhanced Master-Slave 실험결과

Fig 3. Simulation Results for Enhanced Master-Slave

그림 3에서 제안된 질의 처리 전략의 성능이 index 유무에 관계 없이 Peer-to-Peer 전략의 성능에 94%～100%인 것을 보여주고 있다. 이와 같은 2-way join 실험 결과로 제안된 Enhanced Master-Slave 질의 처리 전략의 성능이 Peer-to-Peer 처리 전략의 성능에 join의 방법과 무관하게 거의 일치한다는 것을 알 수 있었다. 또한, 기존의 Master-Slave 질의 처리 전략과 비교해 본다면 slave site간의 자료 전송량이 줄어듦에 따라 질의 처리 전략의 성능이 향상되었다.

5.2 3-way joins

질의 처리 전략의 성능 차에 대한 보다 정확한 분석을 위하여 2-way join과 같은 조건의 환경하에 3-way join을 포함한 질의에 대한 실험을 수행하였다.

Master-Slave 질의 처리 전략의 성능은 Master site에 join에 참여하는 Relation을 가지고 있는 경우는 Peer-to-Peer 질의 처리 전략 성능의 91%～100% 정도 였으나 Master site에 join에 참여하는 Relation을 가지고 있지 않은 경우(전체 4 site가 필요한 질의)는 site간의

도표 2. 질의 처리 전략의 성능 분석

Table 2. Performance Analysis by Query Processing Strategies

Join	Strategy		M-S	P-P	E M-S
2-way joins	Master does not contains join relation	NL,SW	50～100%	100%	96～100%
		NL,FAN	54～99%		94～99%
		SM,SW	50～99%		96～99%
		SM,FAN	50～99%		90～99%
3-way joins	Master contains join relation	NL,SW	92～100%	100%	M-S와 동일
		NL,FAN	95～99%		
		SM,SW	92～99%		
		SM,FAN	91～99%		
	Master does not contains join relation	NL,SW	18～100%	100%	91～100%
		NL,FAN	19～99%		93～100%
		SM,SW	17～99%		91～100%
		SM,FAN	16～99%		90～99%

NL:nested-loop SW:ship-whole

SM:sort-merge FAN:fetch-as-need

Relation 전송양의 증가로 인하여 16%～100%라는 큰 성능 차이를 보이고 있었다.

한편 제안된 Enhanced Master-Slave 질의 처리 전략의 성능은 Master site에 join에 참여하는 Relation을 가지고 있는 경우는 기존의 Master-Slave 질의 처리 전략과 같은 경우가 되므로 성능 또한 같으며 Master site에 join에 참여하는 Relation을 가지고 있지 않은 경우도 91%～100%라는 Peer-to-Peer 질의 처리 전략의 성능과 거의 유사한 우수한 성능을 나타내고 있었다.

Ⅵ. 결론 및 추후 연구과제

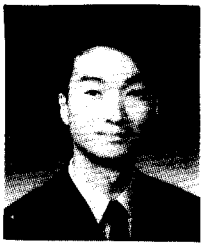
본 연구에서는 우수한 성능을 가지지만 구현상의 복잡성의 단점을 가지고 있는 Peer-to-Peer 질의 처리 전략과 성능은 좀 떨어지지만 구현이 용이한 Master-Slave 질의 처리 전략, 또한 Master-Slave 질의 처리 전략을 개선한 제안된 Enhanced Master-Slave 질의 처리 전략이 어느 정도의 성능 차이를 지니는가에 관하여 실험을 통하여 알아보았다.

결과는 2-way join, 3-way join 질의의 전반적 결과를 볼 때, Peer-to-Peer 질의 처리 전략에 비해 Master-Slave 질의 처리 전략이 16%～100% 정도의 성능을 유지하여 질의의 성격과 질의 처리 환경에 따라 성능에 큰 차이가 나는 것을 볼 수 있었으며 제안된 질의 처리 전략은 91%～100%의 성능을 유지하여 안정적이라는 것을 알 수 있었다. 좀더 다양한 질의에 기반을 둔 질의 처리 실험이 필요하겠지만 분산 데이터베이스 환경에서의 질의 처리 전략으로는 실제 구현이 간단하면서도 Peer-to-Peer 질의 처리 전략과 거의 유사한 성능을 지니는 제안된 Enhanced Master-Slave 질의 처리 전략이 적합한 것으로 생각된다.

추후 연구과제로는 본 논문에서 제시한 질의 처리 전략을 현재의 single processor machine이 아닌 parallel database machine에 적용하여 실험하여 보고 parallel database를 위한 질의 처리 전략을 연구하여 보고자 한다.

참 고 문 헌

1. Stefano Ceri, "Distributed Databases : Principles and Systems", McGraw-Hill Inc., 1984
2. M.T. Ozsu, "Principles of Distributed Database Systems", Prentice-Hall, 1991
3. C.T.Yu and C.C.Chang, "Distributed Query Processing", Computing Surveys, Vol.16, No.4, 1984
4. G. M. Lohman et. al, "Query Processing in R*", Query Processing in Database Systems, W.Kim, D.S.Reiner and D.Batory (eds.), Springer-Verlag, 1985, pp.31-47
5. D. Bitton et. al., "Benchmarking Database Systems, A Systematic Approach", Proc. 1983 VLDB Conference, Florence, Italy, Nov. 1983
6. P.G. Selinger et. al, "Access Path Selection in a Relational Database Management System" , in Proc. ACM SIGMOD, Boston, Mass., 1979, pp.23-34
7. G.M.Lohman and L.F.Mackert, "R* Optimizer Validation and Performance Evaluation for Distributed Queries", in Proc. 11th Intl' Conf. On VLDB, Kyoto, Japan, Aug.1986, pp.149-159



이 형 묵(Hyung-Mook Lee)정회원
1987년 2월:홍익대학교 전자계산학과(이학사)
1989년 2월:홍익대학교 전자계산학과(이학석사)
1993년 3월～현재:홍익대학교전자계산학과(박사과정)

김 경 창(Kyung-Chang Kim) 정회원
1978년 2월:홍익대학교 전자계산 학과(이학사)
1980년 2월:한국과학기술원 전산과(이학석사)
1990년 5월:Univ of Texas at Austin, 전산학과(이학박사)
1991년 3월～현재:홍익대학교 컴퓨터공학과(부교수)