참조파일 유지를 통한 분산 데이터베이스 통신량 감축에 대한 연구

황영헌*, 이우기**, 강석호*

초 록

분산 데이터베이스에서의 통신량을 줄이기 위한 기법과 관련한 많은 연구가 수행되고 있으며, 이러한 연구는 주로 데이터베이스 설계 초기에 고려되는 파일의 배치에 대한 문제를 다루고 있다. 이러한 파일 배치 문제에서는 분산 데이터베이스 상황을 가정하고, 공식화(furmulation)한 후 이에 대한 해법을 제시하는 방식을 취하는 것이 일번적이다.

본 연구에서는 하나의 파일이 다른 지점에 있는 파일을 참조하는 경우 두 파일에 대한 접근량을 줄이기 위한 방법을 제안함으로써 분산 데이터베이스의 물리적 설계의 최적화를 도모하였다. 본 연구에서는 기존의 방법과는 달리 참조정보를 유지하는 참조 파일(referential file)을 추가함으로써 두 지점간의 통신량을 줄이고자 하는 방안을 제시하였으며, 이 때 참조 파일의 종류는 참조하는 레코드의 수를 유지하는 방법과 참조 여부에 관한 정보만 유지하는 두 가지 방법을 모두 고려하였다.

본 연구에서는 기존의 방식과 본 연구에서 제안한 두 가지 방법간의 통신비용을 계산함으로써 본 연구의 타당성을 검증하였다. 이 때 참조 무결성을 유지하는 대표적인 세 가지 제약조건, 즉 Cascade delete, Restricted, Nullify를 대상으로 분석함으로써 이와 관련된 대다수의 프로토콜에서 본 연구에서 제안한 방법의 효용성을 입증코자 했다. 이를 위해 필요한 공식들을 유도해 내는 작업도 논문에서 중요하게 다루었다.

이 연구는 분산 데이터베이스 설계시 통신량을 줄이는 중요한 고려요소로 활용될 수 있을 것이다.

1. Introduction

분산 데이타베이스에서의 통신량을 줄이기 위해 많은 연구가 수행되어 왔으며, 특히 파일배정과 관련한 많은 연구가 이루어졌다. 파일배정 문제란 다수의 site와 다수의 화일이 존재하는 분산 데이타베이스에서의 최적 배정대안을 찾는 최적화 문제라고 할 수 있다. 이 때 최적화의 기준은 비용(cost) 최소화, 수행도(performance) 최대화 두 가지라고 할 수 있다 [Dowdy and Foster]. 이러한 최적화는 분산 데이터베이스 설계 단계에서의 파일배정 문제를 통해서도 달성될 수 있으나[Rothnie and Goodman], 동적인 재배치 방법으로도 달성될 수 있으며[Brunstrom et. al., 황영헌 외], 테이블 갱신비용을 줄이기 위해 디퍼렌셜 파일을 사용하는 방법[Segev, 이우기]도 연구된 바 있다.

본 연구에서는 이러한 방법 외에 레퍼렌셜 파일을 사용하여 통신비용을 줄이는 방법을 제안하고자 한다. 이 방법은 서로 다른 지역에 있는 두 파일이 외래 키(foreign key)로 연결되어 있는 경우 두 파일간의 참조 정보를 유지하는 참조 파일(referential file)을 추가함으로써 두 지점간의 통신량을 줄이고자 하는 방법이다. 좀더 자세히 설명하면 아래 그림과 같이 두 개의 테이블이 서로 다른 지점에 있고, 하나의 테이블(그림에서 테이블A)이 다른 테이블(그림에서 테이블B)의 키를 참조할 때, 이 두 테이블간의 integrity의 유지를 위해 필요한 데이터의 통신량을 줄이기 위한 방법으로 참조 파일을 추가하는 방법을 제안한 것이다.

테이블 A

PK	Att1	Att2	···	FK

테이블 B

PK	Att1	Att2	Att3	···

[그림1] 외래키를 가지는 분산 파일

이를 위해 2장에서는 기존의 방식과 참조 파일 방식에서 일어나는 트랜잭션을 비교하였으며, 3장에서는 통신량의 계산에 필요한 몇 개의 수식을 유도하였다. 마지막으로 4장에서는 두 방식을 비교하기 위한 비용함수를 유도하였으며, 5장에서는 결론을 다루었다.

2. 외래 키를 가지는 파일에서 일어나는 Tx

한 테이블의 키가 다른 테이블에서 외래 키로 사용하는 경우 무결성(integrity)의 유지를 위해 특별한 제약조건이 사용되고 있다[Markoxitz]. 이 때 두 개의 테이블이 같은 장소에 저장된 경우는 큰 문제가 없지만 서로 다른 지점에 위치하고 있을 경우에는 과도한 통신량이 문제가 된다.

발생하는 통신량을 계산하기 위해서는 데이터베이스에서 일어나는 트랜잭션과 관련하여 두 개의 테이블에서 수행되는 작업(operation)을 알아야 한다.

참조 파일이 없는 경우와 있는 경우를 나누어 없을 경우를 나누어 트랜잭션과 관련한 작업을 알아보기로 한다. 편의상 위 [그림1]에 표시된 테이블 A, B에 대한 Query를 나누어 표를 만들었으며, 세 가지 제약조건을 동시에 표현하였다.

2.1 참조 파일이 없는 경우

참조 파일이 없는 경우에 트랜잭션의 처리를 위해 필요한 작업은 본 논문에서 생략하였다. 이에 대한 내용은 2.2절에서 소개한 [표2]와 [표3]으로부터 재구성할 수 있다.

2.2 참조 파일을 사용하는 경우

참조 파일이란 외래 키를 참조하는 정보만을 저장함으로써 참조하는 테이블을 읽는 대신, 참조 정보

* 서울대학교 산업공학과
** 성결대학교 전산학과

만을 읽도록 함으로써 통신량을 줄이고자 하는 방법을 말한다[Lee et. al.]. 참조 파일은 위 예에서 테이블 B가 저장된 지역에 배치시키게 된다.

이런 참조 파일에는 두 가지 방식이 있을 수 있다. 즉, 외래 키의 참조 정보를 나타내기 위해 참조 여부만 유지할 수도 있고, 참조 레코드 수를 유지하는 방식을 취할 수도 있다. 두 방법은 테이블 A를 수정할 때 검색 방식에 차이가 난다. 자세한 내용은

[표3] 테이블 B에서의 Operation

Cascade 일 때

		i A	r A	w A	i B	r B	w B	r RF	w RF	비 고
insert					✓	✓	✓	✓	✓	s, 참조 레코드 0
update(non-PK)					✓	✓	✓			s
update(PK)	FK에 영향 없음				✓	✓	✓	✓	✓	s
	FK에 영향 미침				✓	✓		✓		f
delete	FK에 영향 없음				✓	✓	✓	✓	✓	s, RF에서 삭제
	FK에 영향 미침	✓	✓	✓	✓	✓	✓	✓	✓	s, RF에서 삭제

Nullify 일 때

		i A	r A	w A	i B	r B	w B	r RF	w RF	비 고
insert					✓	✓	✓	✓	✓	s, 참조 레코드 0
update(non-PK)					✓	✓	✓			s
update(PK)	FK에 영향 없음				✓	✓	✓	✓	✓	s
	FK에 영향 미침				✓	✓	✓	✓	✓	f
delete	FK에 영향 없음				✓	✓	✓	✓	✓	s, RF에서 삭제
	FK에 영향 미침	✓	✓	✓	✓	✓	✓	✓	✓	s, RF에 null 추가

Restricted 일 때

		i A	r A	w A	i B	r B	w B	r RF	w RF	비 고
insert					✓	✓	✓	✓	✓	s, 참조 레코드 0
update(non-PK)					✓	✓	✓			s
update(PK)	FK에 영향 없음				✓	✓	✓	✓	✓	s
	FK에 영향 미침				✓			✓		f
delete	FK에 영향 없음				✓	✓	✓	✓	✓	s
	FK에 영향 미침				✓			✓		f

테이블 A에서 delete나 nullify를 위해서는 해당 모든 레코드를 찾아서 처리해야 함.

[표4] 테이블 A에서의 Operation

Cascade 일 때

		i A	r A	w A	i B	r B	w B	r RF	w RF	비 고
insert	null FK 입력									f
	FK가 PK에 없음	✓						✓		f
	FK가 PK에 있음	✓	✓	✓				✓	✓	s, rec# + 1
update(non-FK)		✓	✓	✓						s
update(FK)	FK를 nullify									f
	새 FK가 PK에 없음	✓						✓		f
	새 FK가 PK에 있음	✓	✓	✓				✓	✓	s, rec# +- 1
delete		✓	✓	✓				✓	✓	s, rec# - 1

Nullify 일 때

		i A	r A	w A	i B	r B	w B	r RF	w RF	비 고
insert	null FK 입력	✓	✓	✓						s
	FK가 PK에 없음	✓						✓		f
	FK가 PK에 있음	✓	✓	✓				✓	✓	s, rec# + 1
update(non-FK)		✓	✓	✓						s
update(FK)	FK를 nullify	✓	✓	✓				✓	✓	s, rec# + 1
	새 FK가 PK에 없음	✓						✓		f
	새 FK가 PK에 있음	✓	✓	✓				✓	✓	s, rec# +- 1
delete		✓	✓	✓				✓	✓	s, rec# - 1

Restricted 일 때

		i A	r A	w A	i B	r B	w B	r RF	w RF	비 고
insert	null FK를 입력									f
	FK가 PK에 없음	✓						✓		f
	FK가 PK에 있음	✓	✓	✓				✓	✓	s, rec# + 1
update(non-FK)		✓	✓	✓						s
update(FK)	FK를 nullify									f
	새 FK가 PK에 없음	✓						✓		f
	새 FK가 PK에 있음	✓	✓	✓				✓	✓	s, rec# +- 1
delete		✓	✓	✓				✓	✓	s, rec# - 1

참조 여부만 표시하는 경우(예를 들어 참조는 1, 없으면 0)는 위 테이블에서 rec#를 추가시키는 경우, 기존의 참조 데이터가 0이면 1로 바꿔주고, 반면 rec#를 줄이는 경우는 Table A를 관련된 다른 레코드를 찾을 때 까지 읽어 있으면 그대로 두고, 없으면 0으로 바꿔줌.

[표1]과 [표2]에 소개되어 있으며, A에서의 작업을 정리한 표에서는 참조 레코드의 수를 유지하는 방법을 다루었고, 차이점을 따로 설명하였다.

3. 참조 여부 파악 위해 읽어야 하는 블럭 수 계산

통신량의 계산을 위해서는 각 작업에 소요되는 통신량의 파악이 필요한데, 이 중 참조 여부를 알기 위해 읽어야 하는 블럭의 수는 통신량에 많은 영향을 미치므로 자세한 고려가 필요하다. 여기서는 지면 관계상 간단한 정리만 소개하기로 한다.

계산을 위해 다음과 같이 용어를 정의한다.

U^{R_A}: 테이블 R_A의 레코드 수
W_{R_A}: 테이블 R_A의 레코드 길이
B: 블럭의 크기
r: 한 블럭이 포함할 수 있는 레코드의 수
n: 테이블 R_A가 차지하는 블럭의 수
E_k: 조건을 만족하는 레코드가 k개 일때, 조건을 만족하는 레코드를 찾을 때까지 읽어야 하는 블럭 수의 기대값
$\underline{E_k}$, $\overline{E_k}$: E_k의 하한값, 상한값
p_k: 조건을 만족시키는 레코드가 임의의 블럭에 하나라도 포함될 확률
$\underline{p_k}$, $\overline{p_k}$: p_k의 하한값, 상한값
p^i: i번째 레코드가 임의의 블럭에 있을 확률

실제 데이터베이스 테이블에서의 레코드들은 동적으로 재배치 되므로, 실제 상황을 모두 고려하여 문제를 다룰 수는 없으므로, 두 가지 가정을 주고 문제를 풀었다.

<가정1> 레코드들은 임의로(randomly) 배치된다.
<가정2> 하나의 블럭에 포함될 수 있는 레코드의 수는 무한하다.

이 때 다음 <정리1>을 얻을 수 있다.

<정리1> 가정을 만족시킬 때, 조건을 만족시키는 레코드가 임의의 블럭에 하나라도 포함될 확률, 즉 p_k의 하한값

$$\underline{p_k} = \sum_{i=1}^{k} \frac{(n-1)^{(i-1)}}{n^i} \text{ 이다.}$$

<증명>

$k=1$일 때: $\underline{p_1} = \underline{p^1} = \frac{1}{n}$ 은 자명하다.

$k=m$일 때 $\underline{p_m} = \sum_{i=1}^{k} \frac{(n-1)^{(i-1)}}{n^i}$ 라고 가정하면,

$\underline{p^1} = \underline{p^2} = \cdots = \underline{p^m} = \underline{p^{m+1}} = \frac{1}{n}$ 이므로,

$k=m+1$일 때

$$\begin{aligned}
\underline{p_{m+1}} &= \underline{p_m} \times \underline{p^{m+1}} + \underline{p_m} \times (1-\underline{p^{m+1}}) + (1-\underline{p_m}) \times \underline{p^{m+1}} \\
&= \underline{p_m} \times \frac{1}{n} + \underline{p_m} \times \frac{n-1}{n} + (1-\underline{p_m}) \times \frac{1}{n} \\
&= \frac{(n-1)\underline{p_m}}{n} + \frac{1}{n} \\
&= \sum_{i=2}^{k+1} \frac{(n-1)^{(i-1)}}{n^i} + \frac{1}{n} \\
&= \sum_{i=1}^{k+1} \frac{(n-1)^{(i-1)}}{n^i}
\end{aligned}$$

즉, 레코드의 수가 k인 경우 $\underline{p_k} = \sum_{i=1}^{k} \frac{(n-1)^{(i-1)}}{n^i}$ 가 된다.<증명 끝>

<정리2> 조건을 만족시키는 레코드 수가 k이면, 조건을 만족시키는 레코드를 찾을 때까지 읽어야 하는 블럭 수의 기대값의 상한값은

레코드가 없는 경우: $\overline{E_0} = n$

레코드가 k개인 경우: $\overline{E_k} = \sum_{j=1}^{n} j\underline{p_k}(1-\underline{p_k})^{j-1}$이다.

증명은 생략한다. 여기서 <가정2> 대신 다음과 같은 <가정3>과 <가정4>를 추가하면 아래와 같은 정리를 얻을 수 있다.

<가정3> 모든 블럭은 유한한 수의 레코드를 포함하며, 그 값은 모두 동일하다.
<가정4> 레코드들은 저장될 수 있는 최소한의 블럭에 테이블을 저장한다.

R_A가 저장된 블럭의 수 $n = [\frac{U^{R_A} \times W_{R_A}}{B}]$이며, 전체 레코드 수 T는 $r(n-1)+1 \le T \le nr$이 된다.

<정리3> 가정을 만족시킬 때, 조건을 만족시키는 레코드가 임의의 블럭에 하나라도 포함될 확률, 즉 p_k의 상한값

$$\overline{p_k} = 1 - \frac{{}_{nr-r}C_k}{{}_{nr}C_k} \text{ 이 된다.}$$

<증명> 전체 레코드가 U^{R_A}일 때,

$p_k = 1 - \frac{{}_{U^{R_A}-r}C_k}{{}_{U^{R_A}}C_k}$ 임은 쉽게 알 수 있다. 이 때 $r(n-1)+1 \le T \le nr$이므로, 모든 T에 대해 $\frac{{}_{nr-r}C_k}{{}_{nr}C_k} \le \frac{{}_{T-r}C_k}{{}_{T}C_k}$ 임 또한 자명하다. 그러므로 모든 T에 대해 p_k의 상한값

$\overline{p_k} = 1 - \frac{{}_{nr-r}C_k}{{}_{nr}C_k}$ 이 성립한다.<증명 끝>

<보조정리1> 조건을 만족시키는 레코드 수가 k이면, 조건을 만족시키는 레코드를 찾을 때까지 읽어야 하는 블럭 수의 기대값의 하한값은

레코드가 없는 경우: $\underline{E_0} = n$

레코드가 k개인 경우: $\underline{E_k} = \sum_{j=1}^{n} j\overline{p_k}(1-\overline{p_k})^{j-1}$이다.

실제 데이터베이스에서는 레코드들이 블럭을 완전 채우지 않는 경우가 대부분이다. 이러한 가정을 제외했을 때 레코드들을 포함하는 블럭의 수를 n'라 하면 $n' \ge n = [\frac{U^{R_A} \times W_{R_A}}{B}]$가 된다.

위에서 얻어진 식에 실제 숫자를 입력하여 확인하여 보자. 우선 $k=2$일 때의 확률은 다음과 같이 계산된다.

$p'_2 = 1 - {}_{nr-r}C_2 / {}_{nr}C_2$

$= \frac{r}{nr} \times \frac{r}{nr-1} + \frac{r}{nr} \times \frac{nr-r}{nr-1} + \frac{nr-r}{nr} \times \frac{r-1}{nr-1}$

$= \frac{2nr^2 - r^2 - r}{nr(nr-1)}$

이 때 $\Delta p_2 = p'_2 - p_2 = \frac{n-1}{n^3 r - n^2} > 0, \ (n \geq 2)$

임을 간단한 계산을 통해 알 수 있다. 즉, $p_2 < p'_2$이라는 관계를 얻을 수 있다.

만일 $n=2$, $r=1$이라고 하면, $\Delta p_2 = 0.25$이며, $n=10$, $r=5$이라고 하면, $\Delta p_2 \simeq 0.002$가 된다.

여기서 r을 무한대로 보내면,

$\lim_{r \to \infty} \Delta p_2 = 0$이 되어 $p_2 = p'_2$임을 알 수 있다.

같은 방법으로 계산하면, $k=3$일 때

$p'_3 = \frac{3n^2r^3 - 3nr^3 + r^3 - 3nr^2 + 2r}{nr(nr-1)(nr-2)}$이 되므로,

역시 $\lim_{r \to \infty} \Delta p_3 = 0$이 됨을 알 수 있다.

4. 통신량의 계산

통신량 계산을 위해 다음 용어를 정의하자.

R_A, R_B, R_{RF} : 테이블 A, B, RF

W_{R_X} : R_X의 튜플 크기

U^{R_X} : R_X의 튜플 수(여기서, $U^{R_{RF}} = U^{R_B}$)

그러므로, 테이블 X의 크기는 $U^{R_X} \times W_X$

W_{key_A} : 테이블 A의 Primary key의 크기

W_{key_B} : 테이블 B의 Primary key의 크기

W_{B_X} : 테이블 R_X의 primary key에 대한 B^+ tree 레코드 크기

$H_{B\_R_X}$: 테이블 R_X의 B^+ tree 레코드의 높이(페이지 수)

B : 페이지 크기(bytes)

C_{I/O_i} : i site에서의 I/O 비용 (ms/block)

C_{comm} : 전송율 (bits/s)

$f(N, P, K)$: P페이지를 차지하는 화일에서 N개의 튜플 중 K개에 접근할 때 fetch되는 평균 페이지 수 [Yao77]

이 때 통신량을 다음과 같은 식들을 각 트렌잭션이 일어날 확률값을 고려하여 계산할 수 있다. 이러한 계산은 [Whang]을 참고하였다.

테이블 A의 읽는 비용: $C_{I/O_1} \times [(H_{B_{R_A}} - 1) + 1]$

테이블 A의 쓰는 비용:

$f(U^{R_A}, U^{R_A} \times W_{R_A}/B, 1)/2$ (insert 할 때)

$C_{I/O_1} \times [(H_{B_{R_A}} - 1) + 1 + \alpha]$(delete, update 할 때)

테이블 B의 쓰는 비용:

$C_{L_B} \times f(U^{R_B}, U^{R_B} \times W_{R_B}/B, 1)/2$ (insert 할 때)

$C_{L_B} \times f(U^{R_B}, U^{R_B} \times W_{R_B}/B, 1)$(delete, update 할 때)

테이블 B의 탐색 비용:

$C_{I/O_2} \times [(H_{B_{R_B}} - 1) + f(U^{R_B}, U^{R_B} \times W_{R_B}/B, 1)]$

테이블 RF를 읽는 비용: $C_{I/O_2} \times (H_{B_{R_B}} - 1)$

테이블 RF의 쓰는 비용:

$C_{L_{RF}} \times f(U^{R_{RF}}, U^{R_{RF}} \times W_{R_{RF}}/B, 1)/2$ (insert 할 때)

$C_{L_{RF}} \times f(U^{R_{RF}}, U^{R_{RF}} \times W_{R_{RF}}/B, 1)$(delete,update할 때)

insert, delete, update와 관련한 PK 전송비용:

$8 \times W_{key_B}/C_{comm}$ (maybe $\propto 0$)

5. 결론

본 연구를 통해 분산 데이터베이스의 물리적 설계단계에서 참조 파일을 구성함으로써 통신량을 줄일 수 있음을 보였으며, 서로 다른 세 가지 제약조건 하에서 같은 결과가 도출됨을 보였다. 또한 참조 파일은 단순히 참조 여부를 유지하는 것 보다, 참조하는 레코드의 수를 유지하는 것이 유리한 방법임을 보였다.

앞으로 보다 다양한 트렌잭션에 대해 본 결과를 적용해 봄으로써 결과를 검증하는 것과 테이블에 대한 접근 형태에 따른 참조 파일의 위치에 대한 연구가 이루어져야 할 것이다.

<참고문헌>

1. 이우기, 분산 데이타베이스 구현을 위한 레플리케이션 서버 체계에 관한 연구, 박사학위 논문, 서울대학교 산업공학과, 1996.
2. 황영헌, 김대환, 김영호, 강석호, "분산 데이타베이스에서의 동적 화일배정에 관한 연구," 한국경영과학회 '96 추계학술대회 논문집, pp.275-278.
3. Brunstrom, A., S. T. Leutenegger and R. Simha, "Experimental Evaluation of Dynamic Data Allocation Strategies in a Distributed Database with Changing Workloads", Conference Information and Knowledge Management, Baltimore, 1995, pp.395-402.
4. Date, C. J., "Intro. to Database Systems," Addison-Wesley, 5th ed. vol. 1, 1990.
5. Dowdy, L. W. and D. V. Foster, "Comparative Models of the File Assignment Problem", ACM Computer Survey, June, 1982, 14(20), pp.287-313.
6. Lee, W., J. Park, and S. Kang, "Implementing Distributed Referential Integrity Rules, Proc. of the KOR/MS Conference, 1997.
7. Markowitz, V. M., "Safe Referential Integrity Structure in Relational Databases," Proc. of the 17th Int. Conf. Very Large Data Bases, 1993, pp.389-398.
8. Özsu, M. T. and P. Valduriez, Principles of Distributed Database Systems, Prentice-Hall, 1991, p.137.
9. Segev, A. and J. Park, "Updating Distributed Materialized Views," IEEE Transactions on Knowledge and Data Engineering, Vol.1, no.2, June 1989.
10. Whang, K., "Constructing Cost Formulars for Relational Database Query Optimizers," IEEE TENCON, 1987, pp.132-140.
11. Yao, S. B., "Approzimating Block Accesses in Database Organizations," Communications of the ACM, vol.20, 1977, pp.260,261.