韓國OA學會論文誌
第1卷 第1號, 1996. 8

論文 96-1-1-4-2

DBFG를 이용한 동시성제어 구현 방법에 관한 연구

남 태 희*, 위 승 민**

The Study for Implementation method of Concurrency Control for DataBase Flow Graphs

Tae Hee Nam*, Seung Min Wee**

요 약

본 논문에서는 작업 스케줄러를 가진 통합된 실 시간 동시성제어 수행 과정을 분석하여 특수화된 데이타 흐름 그래프에 기초로한 동시성 제어구조를 제안하였다. 자료들은 토큰들의 이산 흐름에서 한 노드로 부터 다른 노드까지의 연결 호 상에 나타내었다. 또 한 E-R모델에서 알려진 네트워크는 데이타 흐름 그래프로 나타내는 고정적인 문제점을 질의어 토큰을 이용하여 그래프상에 나타내게 하였다. 그리고 모든 동시성 제어 기법들은 서로 다른 성능을 비교하여 실험으로 측정하였고, DBFG 스케줄링은 분산 환경에서 2PL보다 뛰어난 성능을 갖는다는 것을 비교 분석하였다.

ABSTRACT

This paper proposed a concurrency control structure based on specialized data flow graphs that was analysed a run-time concurrency control activity to be integrated with the task scheduler. Data were viewed as flowing on the arcs from one node to another in a stream of discrete to tokens. The network that is based upon the Entity-Relationship model, can be viewed a fixed problems used query tokens as a data flow graph. The performance was measured used in the various expriments compared the overall performance of the different concurrency control methods. DBFG(DataBase Flow graphs) scheduling had the knowledge to obtain better performance than 2PL in a distributed environment.

. 서 론

본 논문은 태스크 스케줄러로 통합된 실시간 동시성 제어가 가능하고, 동시에 2단계 로킹(2PL)과 같은 전통적인 방법을 통하여 달성할 수 있는 것보다 제한이 훨씬 적은 통상적

* 동주여자전문대학 무역사무자동화과

** 한국해양대학교 대학원 해사수송과학과

스케쥴을 특수한 데이타 흐름 그래프(DFG)에 기초로한 동시성 제어 기법을 분석한다. 이는 스케쥴링 과정을 통합된 동시성 제어 수행에 의해서 처리되는 시간을 최소화하는 것을 시뮬레이션으로 분석하고, 데이타 흐름 모델에서 계산은 데이타 위주로, 명령들은 요구되는 오퍼랜드들이 유용할 때 실행되고, 데이타 플로우 프로그램들은 노드가 실행되는 명령을 나타내고, 호(arc)는 그들 사이에 데이타 종속관계를 나타내는 것을 DFG라 부르는 방향성 그래프로 나타낼수 있는데, 이때 데이타는 이산 토큰 흐름에서 한 노드에서 다른 노드로의 호상에 흐름으로 나타낸다. 또한 명령의 스케쥴링은 그래프에 의해서 특징 지워진 데이타 종속성에 의해서만 구성된다. 즉 동일 방향상의 경로에서 명령은 직렬로 스케쥴 되고, 다른 경로에서의 명령은 독립적으로 스케쥴될 수도 있고,또 다른 토큰상에서 수행하는 동일 명령어의 다른 실행은 동시에 스케쥴링 될 수 있어야 한다. 만약에 데이타 베이스 시스템에서 수행 될 수 있는 모든 명령들을 포함하는 데이타 흐름 그래프가 작성 될 수 있다면 데이타 흐름 스케쥴링 메카니즘은 DFG에 의해서 정의된 종속성 제약 조건을 기초로하여 명령어들을 실행시킬수 있다. 만약에 토큰들이 페이지가 되고 교차 트랜잭션 경계로 흐른다면, 동시성을 최대로 하기 위하여 록킹 단위 또한 페이지 수준에 있어야한다. 페이지의 크기가 특정 지워지지 않고, 동시성 제어 목적을 위한 유일한 요구는 토큰의 크기와 동시성 제어 단위가 같아야 한다. 또한 개개의 트랜잭션을 나타내고 있는 DFG는 모든 활동 트랜잭션을 나타내는 DFG와 명령어들과 충돌하는 DFG를 서로 형성하도록 병합한다. 그 결과 DFG는 트랜잭션 집합에 대한 스키마로 처리된다. 이 기법은 록킹보다도 트랜잭션당 정보를 더 적게 요구하고, 대부분의 처리 오버해드는 처리중일 때보다도 트랜잭션 실행전이나 또는 후에 발생한다.

그러므로 일반적으로 사용되는 질의어로 저장하거나 프리컴파일하는 시스템에 있어서, 대부분의 동시성 제어 오버헤드는 각각의 질의어를 프리컴파일하는 중에 발생되고, 그러므로해서 시스템의 수명이 감소되는 원인이 될수도 있다. 따라서 전/후 프로세서는 시스템에 들어가고 나오는 트랜잭션으로서 DFG를 유지 보수하는데 사용되고, 이 전/후 프로세서는 트랜잭션을 처리하는데 병렬로 실행될 수 있어야한다. 그러므로 트랜잭션을 처리하는 중에 발생되는 오버해드를 최소한으로 줄이는 것이다.

Ⅱ. DBFG의 구성 방법

본 논문은 먼저 직렬성과 트랜잭션의 기본 표기법을 사용한다. 각 트랜잭션에는 Query Dag(QD)이 있는데 이것은 트랜잭션 내에서 명령과 순서 제약조건을 나타낸다.

정점을 V_D, 호를 A_D, 관계를 R_D라고 하면 QD D = (V_D, A_D, R_D)는 방향성 무순환 그래프가 된다. 그리고 그 그래프의 내부 노드는 명령어이고 시작노드(입력이 없음)와 끝노드(출력이 없는 것)는 서로 관계노드이다. 그 트랜잭션에 의해서 액세스된 각 관계에 대해서 최소한 한개의 시작노드가 있고, 만약에 끝노드가 사용자에게 출력관계를 나타낸다면 정확히 한개의 끝노드는 "OUTPUT"이라고 이름을 붙인다. 그러므로 QD는 그 트랜잭션의 관계대수 특성의 직접적인 결과가 된다. 여기서 충돌 형태의 QD만을 고려해보면, QD에서 데이타베이스 운영은 방향성 경로에서 진행된다. (그림 1)은 3가지 종류의 QD로 DFG의 형태로써 나타낼수 있는데, 이는 트리의 동일한 방향성 경로에서 나타나는 모든 운영은 종속적이나, 다른 것은 독립적이다. 또한 관계 사이

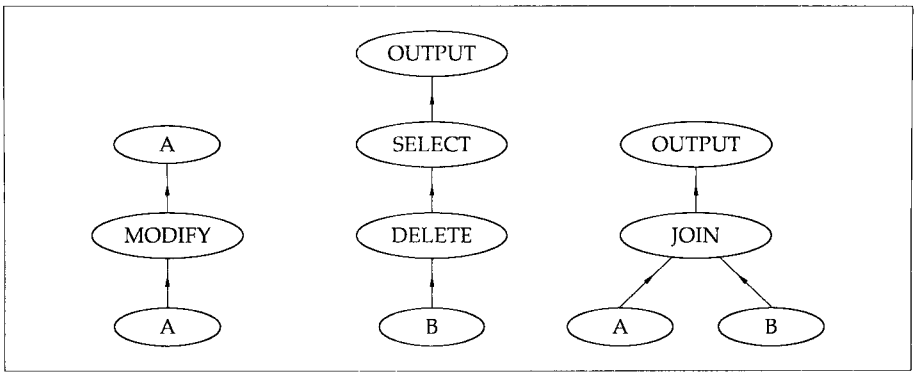


그림 1. Query Dag 예제

Fig 1. Example of Query Dag(QD)

의 페이지는 한페이지에서의 운영과 튜플들사이 호에서의 흐름이 어떤 순서를 유지하는 것으로 보여질 수 있다. 또한 트랜잭션 T에 대한 하나의 검색 작업은 입력 페이지를 조사하고, 입력 페이지로 튜플들의 통과여부를 결정한다. 또한 T에 대한 하나의 갱신 운영은 입력 페이지를 검사하고 그것을 갱신할 것인지를 결정한다. 출력 페이지는 입력 페이지가 변경될 수 있다는 것을 제외하면 입력 페이지와 같은 물리적인 페이지로서 그것이 초과할 경우에 페이지가 2페이지로 나누어 질 수 있음을 가정하면,

표 1. 명령어들 사이의 종속 수준

Table 1. Dependency lebel between operations

T1	T2	
	검색	갱신
검색	독립	노드 종속
갱신	트랜잭션 종속	트랜잭션 종속

연관된 직렬 스케쥴이 T1과 T2인 데이타베이스 시스템에서 동시에 수행되는 2개의 트랜잭션 T1과 T2를 생각하면, T1과 T2가 같은 페이지를 접속하는 운영을 포함하면 그들 사이에 존재할 수 있는 3가지 레벨의 종속성이 있다. 레벨들은 (표 1)과 같이 정의할 수 있는데, 이는 두 운영이 검색이면 그들은 독립적이고 동시에 수행될 수 있다. 그러므로 검색이 갱신의 기능에 의해 노드는 종속레벨이 존재한다. 따라서 구조적인 자료 흐름도에서, T2에서 입력호는 그 페이지에 접속하는 T1에서 마지막 노드로부터의 출력이 된다.

Ⅲ. DBFG의 시스템 설계

본 논문에서는 종속성의 레벨을 모델화 하기 위해 각각의 관계에 대한 토큰 호를 clean과 dirty호로 구분한다. 여기서 갱신된 페이지는 dirty, 갱신되지 않은 페이지는 clean으로 정의하고, clean 페이지만이 clean 호로 흐르고, dirty 페이지는 dirty 호로 흐른다. 그리고 수 페이지의 임시 관계는 항상 dirty로 고려된다. 그리고 운영노드를 통과하는 페이지의 상태(dirty, clean)는 단지 그 페이지가 그러한 운영에 의해 갱신될때만 바뀐다. 또한 트랜잭션 T에 대한 단일 완료노드는 dirty 페이지를 새롭게 한다. 그것은 그 트랜잭션에서 최근 운

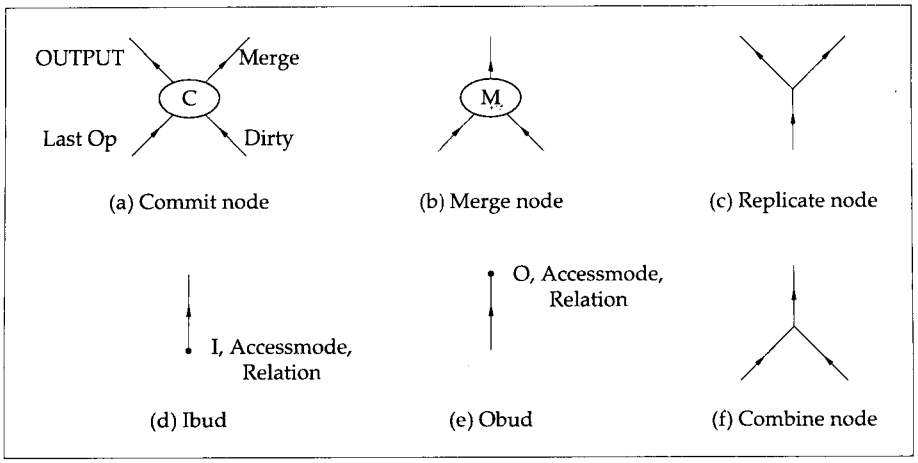


그림 2. 특별한 DBFG와 TFG 노드

Fig 2. Special DBFG and TFG nodes

영노드로부터 한개의 입력 호를 가지고 있고, 또한 각 갱신된 관계로부터 dirty 페이지에 대한 입/출력 쌍을 갖는다. 완료에 대한 입력은 T에서 어떤 갱신 작용에 사용된 모든 페이지로 구성되고, 이들 페이지들은 완료노드를 불변으로 놓지만 clean으로 고려한다. 이러한 새로운 페이지는 변화되지 않고 같은 관계로부터의 페이지와 결합되는 결합노드에 전달된다. 그리고 완료노드는 트랜잭션이 끝날때까지 dirty페이지 큐에 저장하고, 노드의 소멸시까지 DBFG로부터 트랜잭션의 제거에 대한 신호를 큐에 저장한다. (그림 3)은 3가지 종속레벨의 모델화로 독립과 노드 레벨의 종속은 표준 데이타흐름의 형태로 정의된다. 여기서 독립성은 토큰의 한 흐름을 2개의 같은 흐름으로 쪼개는 복사 노드에 의해 출력되고, 노드 종속성을 갖는 관계 페이지는 그 다음 트랜잭션이 검색 노드에 의해 보여진 후에 다음 트랜잭션으로 넘어갈 수 있다. 이것을 모델화 하기 위해 검색작업은 원래의 관계 페이지가 그 노드를 통해 흐른후 놓여지게될 수 있는 추가적인 출력호가 제공된다. 또한 트랜잭션 종속은 (그림 3(c))에서 보여진 것처럼 완료와 결합노드를 사용하여, clean과 dirty 페이지에 대한 호를 각각 C와 D로 이름 붙인다.

먼저 TFG로부터 DBFG 구성을 위해, 자료흐름도를 연결하고 분리하는 점이 정의되고, 그들이 새로운 하위 그래프가 자랄 수 있는 그래프 노드이기 때문에 buds라고 한다. bud는 형, 엑세스 모드 그리고 관계등 3가지로 이름을 붙이고, '형'은 입력(I) 또는 출력(O) bud인가를, '엑세스 모드'는 노드가 단지 읽기 전용 엑세스(S: 공유) 또는 갱신 엑세스(X: 배타적)인가를 구분하고, '관계'는 노드와 관계된 이름임을 나타낸다. 그리고 페이지는 Ibud의 TFG로 흐른다. 여기서 트랜잭션이 한 페이지로 끝날때, 그 페이지는 트랜잭션으로부터 Obud로 흐르게 되어있다. 트랜잭션 사이의 페이지 이동은 Ibuds와 Obuds를 연결함으로써 이루어 지며 이 과정을 접목법이라 한다. 따라서 그들이 수행되는 동안에 한 페이지의 자료가 Obud에 이르면, 그것은 한쌍의 Ibud를 경유하는 다음 트랜잭션에서 입력으로 작용한다. 또한 동시성 제어는 TFG에서 buds의 적당한

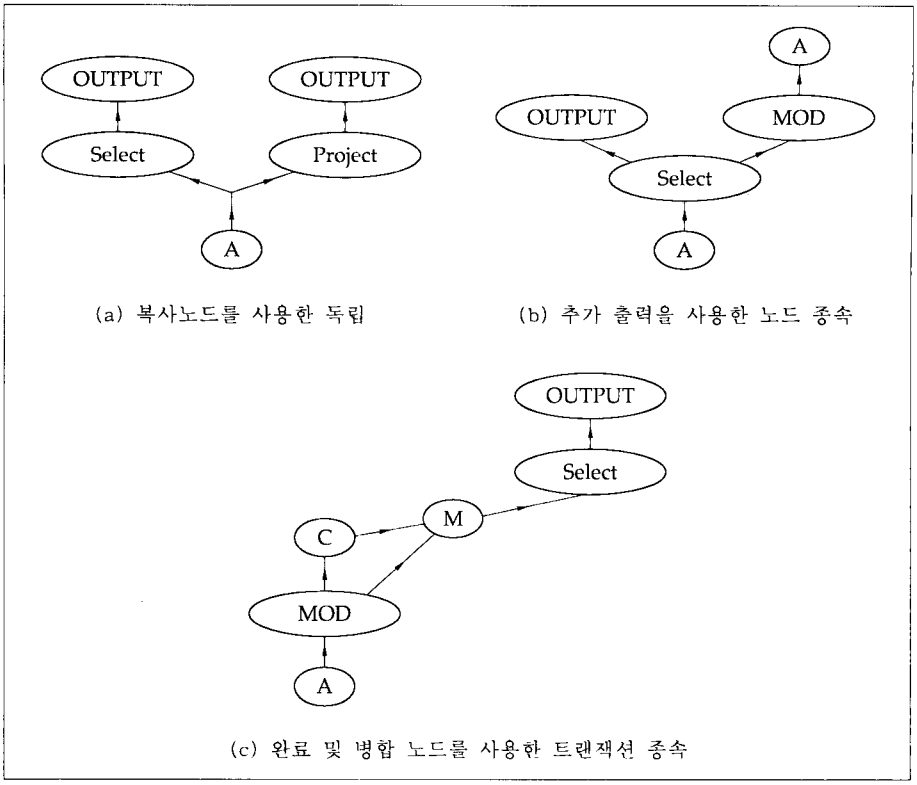


그림 3. 종속 레벨 표현

Fig 3. Representation of dependence levels

위치 설정과 접목법에 의해 제공된다. (그림 4)는 4개의 기본 운영노드를 가진 buds의 트랜잭션과 이름 설정을 보여주는데, 복잡한 운영은 어느 튜플에 영향을 받을것인가를 결정하기위해 다른 관계의 내용을 사용하지만, 단순 운영은 목표 튜플 질의어 그 자체에 의해 지시된다. 따라서 각각의 경우에서 보여진 완료노드는 실제로 일치하는 트랜잭션에서 마지막 노드뒤에 위치될 것이다. 여기서 각각의 관계는 S Obud와 X Obud를 갖는다. S Obud는 대응관계에서 다음의 검색 트랜잭션에 대한 연결지점을 지정하고, X Obud는 갱신트랜잭션를 위한 연결 지점을 지시한다.

또한 자료흐름도는 그래프에서 어느 노드가 만족될 수 있는가의 부분적인 순서를 정의하면, U_r이 관계 r을 갱신하는 명령 노드를 나타내면, S_r이 관계 r을 검색(선택)하는 데이타 노드를 나타내고, 따라서 $O_r = S_r \cup U_r$로 정의할 수 있다. 또한 Ib_r은 관계 r에 대한 Ibud를 나타내고, Ob_r은 r에 대한 Obud를 나타낸다. buds는 그들의 엑세스 모드인 S 또는 X에 의해서 첨자로 정의된다. 본 논문에서는 TFG나 DBFG의 노드 a, b 두개의 분적인 순서를 정의한다.

(1) $a <_r b$: r 흐름의 페이지에서 a에서 b로 호가 존재한다.

(2) $<$: 모든 r에서 $<_r$의 이행적 닫힘을 나타낸다. 그러므로 만약 a에서 b로 방향성 경로가 존재한다면 $a < b$이다.

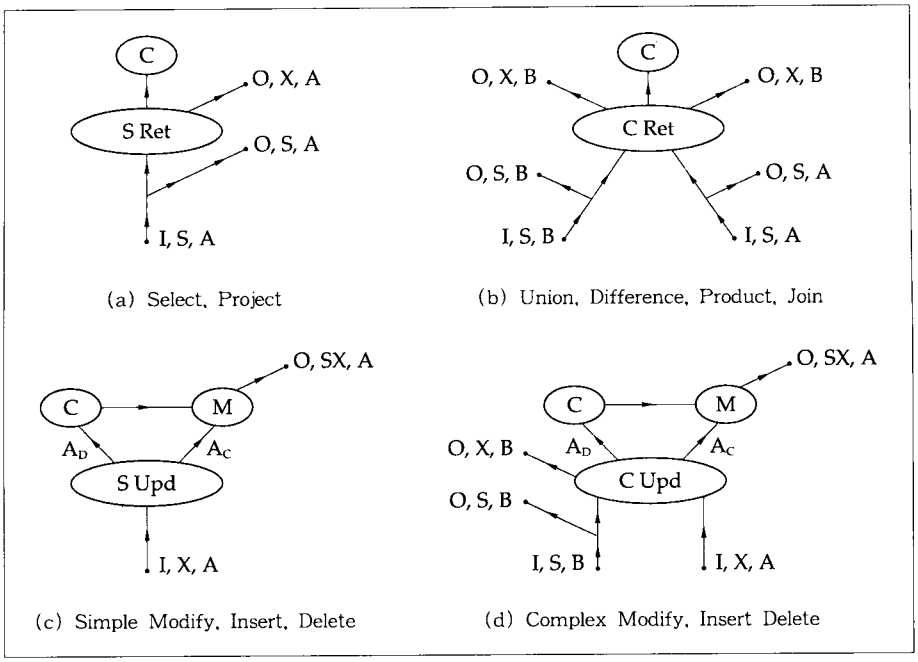


그림 4. 요구된 특별 노드를 포함하는 기본 명령 노드

Fig 4. Basic operation nodes including required special nodes

이제 TFG와 DBFG를 정의하면, 트랜잭션 흐름 그래프(TFG)는 한 트랜잭션과 연관된 방향성 그래프이다: $T = (V_T, A_T, R_T)$ V_T 정점, A_T 호, R_T 관계. 정점은 명령노드, OUTPUT 관계노드, 완료노드, 결합노드, 복사노드, 조합노드, Ibuds, Obuds 그리고 TFG에서 유일한 시발점은 Ibuds이고, 유일한 종착점은 Obuds이며 가능한 하나의 OUTPUT 관계이다. 여기서 처리하는 동안에, 유일한 clean 페이지는 TFG의 안과 밖으로 움직인다. 모든 $o \in O_r$이면 $o <_r Obr_X$이고, 모든 $o \in U_r$이면 $o <_r Obr_S$이다. Obr_X와 Obr_S가 그래프에서 동일한 위치에서 발생할때, 하나의 Obud Obr_{SX}는 보여진다. 따라서 (그림 1)에 있는 QD에 대한 TFG를 (그림 5)로 나타낼 수 있다. 데이타베이스 흐름 그래프(DBFG) $G = (V_G, A_G, R_G, \tau_G)$는 $RG = U_{T \in \tau_G} R_T$인 곳에서 모든 활동 트랜잭션 τ_G에 대한 유향 그래프이다. 정점의 집합 VG는 명령 노드, 관계노드, 완료노드, 병합노드, 대체노드, 조합노드 그리고 Obuds로 구성되어 있다. DBFG가 또다른 DBFG로 병합되지 않았기 때문에 Ibuds는 없다. 유일한 시작노드는 입력관계노드이고, 유일한 종착노드는 Obud와 OUTPUT 관계노드이다.

(그림 6)은 시스템에 처음으로 MODIFY 트랜잭션을 넣고 마지막에 JOIN 트랜잭션을 넣는다고 가정하는 (그림 5)의 TFG에 대한 DBFG를 보여준다. 이때 활동 트랜잭션이 존재하지 않을 때 DBFG는 각 영구관계에 대한 하나의 SX Obud와 하나의 입력 관계 모드로 구성되어 있다.

Ⅳ. DBFG 스케쥴링

DBFG의 궁극적인 처리 목적은 모든 데이

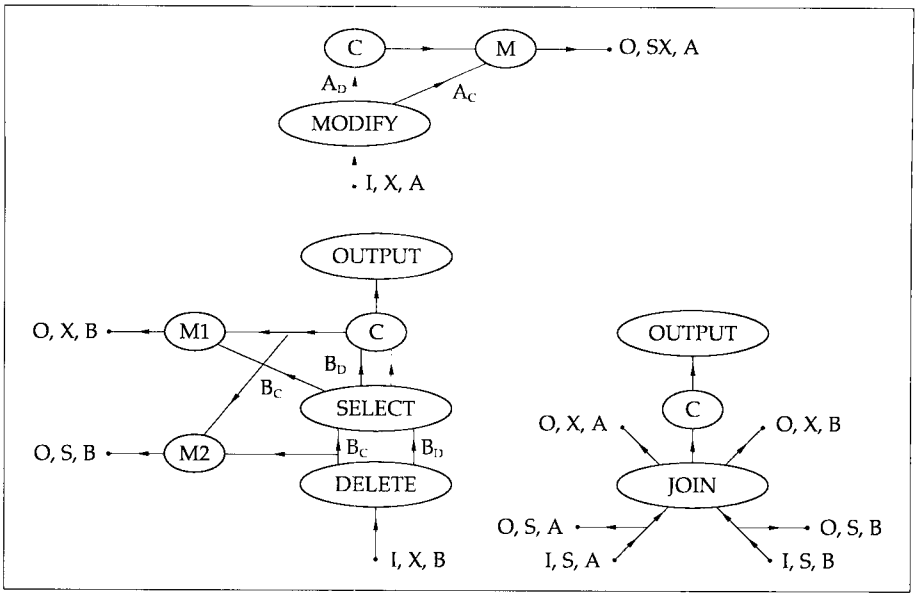


그림5. 그림 1에 있는 QD에 대한 TFG

Fig 5. TFGs for Query Dags at Figure 1

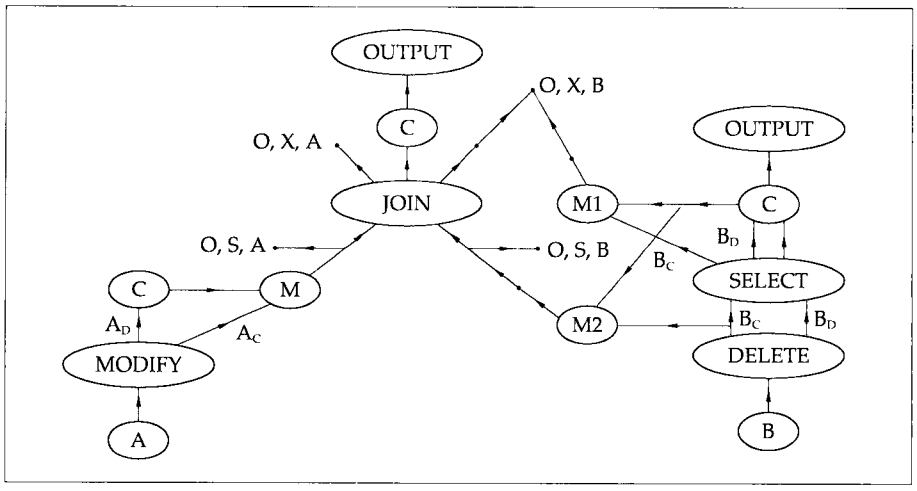


그림 6. 그림 5에서의 DBFG 모델링 트랜잭션

Fig 6. DBFG modeling transactions in Figure 5

타베이스 명령어에 직렬화 자료흐름 스케쥴을 설계하는 것이다. 즉 어떤 DBFG 스케쥴은 개개의 QD에 대한 직렬 실행과 동일하다 동시성 유형은 스케쥴러로서의 기능을 수행하고 DBFG 스케쥴링을 사용하는 그래프 그 자체는 가능한 직렬화 스케쥴의 집합으로 정의한다.

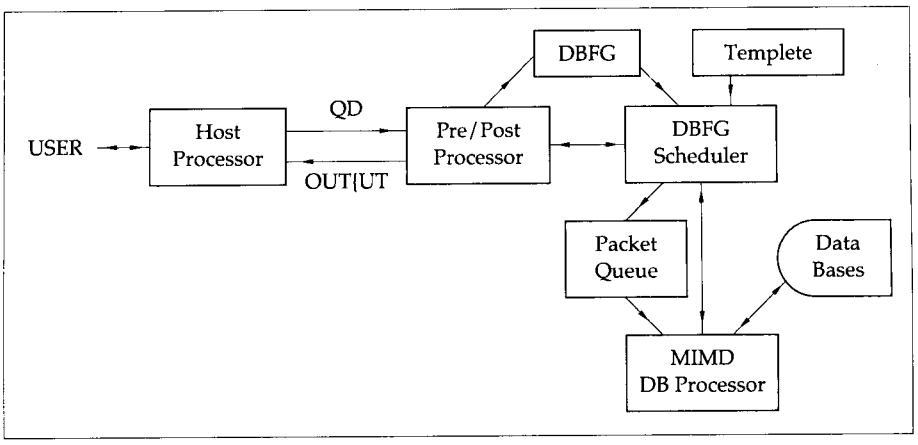


그림 7. DBFG 스케줄링 처리기 모델

Fig 7. DBFG scheduling processor model

(그림 7)에서는 어떤 DBFG 동시성 구현에는 두가지 중요한 요소가 있다. 전/후 처리기는 CREATE/MERGE/REMOVE 명령을 이루고 사용자와 연결한다. DBFG를 실험함에 있어서 DBFG 스케줄러는 무슨 명령어가 실행되고, 패킷 큐에서 유용한 명령 패킷을놓고, 임시영역에서 (임시) 완료 패킷을 부분적으로 놓는지를 결정한다. 그리고 DBFG 스케줄러는 질의어 처리기에 의해서 실행되는 명령어를 스케줄한다. 또한 전/후 처리기는 3가지 중요한 기능을 수행한다. 즉 호스트 처리기로부터 얻어진 QD를 편집 및 검색, DBFG 상에서 유지보수 수행, 그리고 출력자료를 호스트 처리기에 되돌려준다. 또한 호스트 처리기를 가진 I/O는 DBFG 스케줄러에 독립적 이어서 병렬로 DBFG 스케줄러를 운영할 수 있다. 그러나 DBFG 유지보수는 DBFG 스케줄러에 영향을 주지않도록 운영되어야만 한다. 그 두가지는 상호루틴 또는 적절한 통신으로 병렬로 처리될 수 있다. 또한 DBFG의 2가지 복사본은 2번째 복사본을 사용하는 DBFG 스케줄러와 1번째 복사본을 사용하는 전/후 처리기로 구성되는데, DBFG 그 자체는 I/O 오버해드를 없애기 위하여 프로세서 메모리에 항상 남아있어야만 한다. 따라서 유용한 프로세서 할당전략은 프로세서에 가능한 패킷을 할당하기 위하여 사용될 수 있고, 실행은 어떤 유용한 처리기에 할당된 패킷 큐에서 위치한 시간에 기초한 FIFO 양식에서 수행될 수 있을 것이다. 그러나 이것은 메모리 캐쉬와 대량저장 장치 사이에 추가 I/O를 요구하는 것으로, 이것은 훌륭한 처리기 할당 전략이 메모리 캐쉬와 데이타베이스 대량 저장 장치 사이에 페이지 스왑핑량을 감소시키고, 단순한 할당 전략보다 3배 정도 빠르게 될 것이라는 것을 보여준다. 그러므로 DBFG 스케줄러 구현은 본 자료 흐름 처리기 할당 전략 또는 다른 페이지 위주의 전략을 사용한다.

V. 성능 분석

본 논문에서는 2PL과 DBFG 스케줄링의 성능을 비교하여 시뮬레이션 결과를 분석하면 다음과 같다.

1. 시뮬레이션 모델

본 논문의 각 시뮬레시션 실험은 입력에서 몇몇의 트랜잭션을 사용한다고 가정하고, 각각의 실행은 모든 트랜잭션이 완료될 때까지 계속된다. 또한 모든 트랜잭션은 완료 될 때까지 성공적으로 실행된다고 가정하므로써 회복기능은 모델화하지 않았다. 각 트랜잭션의 실험은 다음과 같은 6가지 구성된다.

(1) 모든 CPU 시작 명령은 비트 맵이나 로킹 테이블을 필요로 하는 할당 및 할당된 공간의 초기화, QD의 순회와 저장, 그리고 록킹의 요구를 포함해서 트랜잭션을 요구한다.(전처리기)

(2) 2PL 기법에서 트랜잭션은 처리를 하기 전에 모든 록킹을 얻어야만 한다. 블록된 트랜잭션은 가능한한 빨리 만들어진 록킹을 얻기 위하여 다음 시도로 블록된 큐에 놓여진다.(록키을 얻거나 Ibud의 처리)

(3) 각 트랜잭션은 처리를 하기 전에 각 트랜잭션에 I/O 명령을 검색하기 위하여 페이지당 논리적인 I/O 명령을 작성한다.(I/O 요구)

(4) 질의어 처리기 실행은 그 명령이 간단한지 또는 복잡한지를 분석하여 페이지를 처리하기 위하여 정해진 시간을 가정함에 따라서 시뮬레이트 된다.(QP 요구)

(5) 2PL 기법에서 오버해드는 록킹을 해제하기 위해서 발생한다.(록킹을 해제하거나 Obuds의 처리)

(6) 모든 동시성 기법에 대한 완료 처리는 테이블을 자유롭게하는 시뮬레이팅을 포함하고, 그 공간은 수정된 페이지의 영구적인 갱신과 잘 맞지 않는 데이타의 통과 여부 뿐만아니라 QD를 그 다음의 명령으로 저장하기 위해 사용된다.(후처리기와 완료)

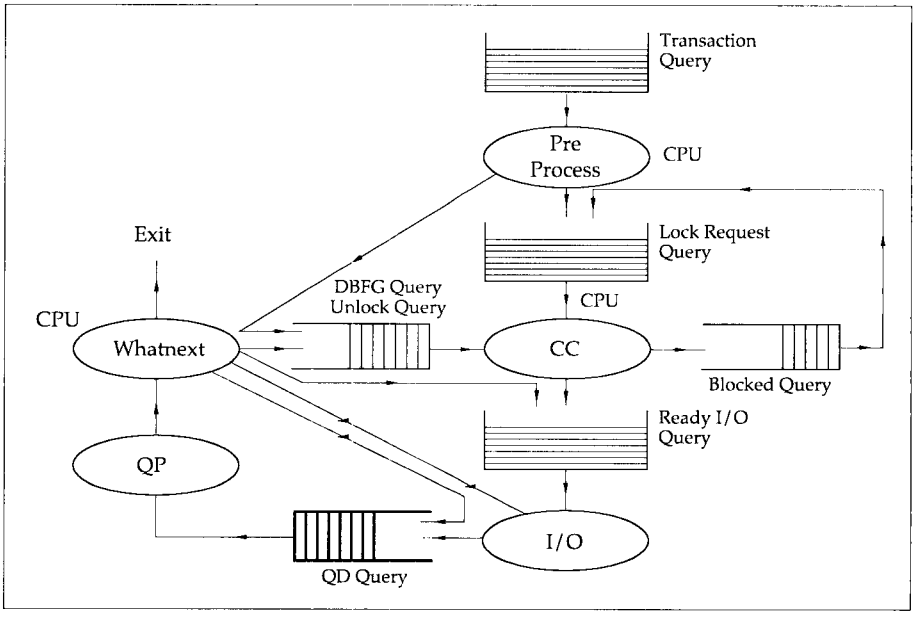


그림 8. 시뮬레이션 모델

Fig 8. Simulation model

DBFG 스케쥴링에서 이것은 DBFG로부터 연관된 트랜잭션 흐름그래프를 제거하는 원리이다. 이러한 단계는 트랜잭션 구조와 평가된 동시성 제어기법에 의해서 결정된 정확한 시간과 순서에 따라 다중 처리으로 이루어 진다. 시뮬레이션 모델의 논리구조는 (그림 8)로 나타낼수 있는데, 시뮬레이션 모델은 3가지 형식의 물리적인 자원 즉 CPU 서버, 몇 개의 I/O 서버, 질의 처리기와 같은 것을 포함한다.

다양한 실험에 사용된 성능 측정은 전반적인 성능과 동시성 제어 기법의 차이와 비교된다. 동시성의 정도를 측정하기 위해서, 트랜잭션당 평균 응답시간과 처리율이라는 2개가 사용된다. 또한 동시성 정도가 충돌할 때 CPU와 I/O 장치의 효용성은 시스템 병목현상을 결정하기 위하여 실험되고, 처리의 오버해드를 비교할 때 트랜잭션당 평균 동시성제어 오버해드와 응답 시간에 대한 동시성제어 오버해드의 비율을 실험한다.

2. 성능 측정

본 논문에서 평가된 성능측정은 트랜잭션당 평균 응답시간이다. DBFG 스케쥴링은 단순 검색 트랜잭션에서 2PL보다 더 좋은 성능을 가진 것이라고 기대했었다. 왜냐하면 이러한 형식의 트랜잭션은 복잡하고 갱신된 트랜잭션보다 더 작은 QD에 더해지는 노드를 요구하기 때문이다. 특별히 CRM(복잡 검색 혼합)에서 발견된 특정변수와 경향을 설명하기 위하여 이런 변수들 각각이 어떻게 결정되는지를 명백히 설명해야 할 필요가 있다. 따라서 트랜잭션의 수와 특정한 혼합을 위해서 다섯 개의 실행이 동시성제어 기법의 각각에 대해 이루어지는 것을 볼 수 있다. 이와같은 실행은 트랜잭션과 입력 파라미터와 같은 집합으로 정확히 구성되어 있다. 물론 트랜잭션의 갯수 보다 더 동일한 입력 파라미터에서도 사용된다고 볼수 있을때, 이들이 어떻게 완전히 서로 다른 결과를 만들수 있는지를 이해하기 위하여 100개 트랜잭션과 CRM에대한 응답 시간을 분석하였다. 분석결과 보여준 응답시간은 다른 값들과 비교해 볼 때 서로 상이하다. 즉 복잡한 질의어가 실행될 때, 무작위수에서 차이는 실행된 명령어와 선택된 페이지의 수에서 상당한 충돌을 가질 수 있고, 그 충돌은 각 레벨에서 생성된 페이지 수와 지수적인 증가 때문에 나타난 조인명령어에 의해서이다. 모든 교차 트랜잭션 혼합을 계산했던 트랜잭션당 평균 동시성 오버해드는 최악의 오버해드를 갖는 방법을 결정하기위해 분석하였다. 이 시간은 실행을 위한 블럭된 시간이나 어떤 대기시간도 포함하지 않고, 단지 동시성 기법을 달성하기위해 요구된 CPU 시간이다. (그림 9)는 각 트랜잭션 수에 대한 제어 방법으로미리 초단위로 평균 동시성 제어 오버해드를 보여준다. DBFG스케줄링에 대한 오버해드는 증가하는 트랜잭션 수로써 보여준대로 상수이고 2PL의 오버해드는 트랜잭션수에 매우 민감하다. 록킹 기법에 대한 시간이 증가하는 중요한 이유는 트랜잭션 수로 인한 블럭 된 큐의 증가된 크기 때문이다. 더 많은 트랜잭션이 블럭화되었기 때문에, 존재하는 록킹이 해제될때 실행하는 블럭된 트랜잭션을 결정하는데 요구되는 시간은 증가한다.그러나 DBFG에 대해서 동시성 제어 활동은 트랜잭션 수에 의존하지 않는다.

Ⅵ. 결론 및 앞으로의 연구

동시성 제어의 기법은 스케쥴러에 의해서 실행된 모든 실시간 동시성 제어 수행이 가능하도록 제안하였다. 이는 분산 환경에서 버스 함유를 감소시키는 잠재적인 장점이 있고, 또

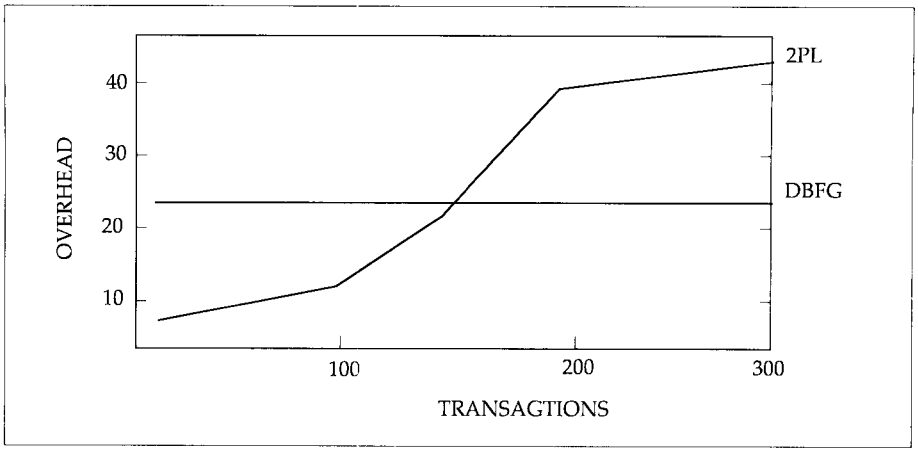


그림 9. 실험 1에서 동시성제어 오버해드

Fig 9. Concurrency control overhead at experiment 1

한 이 기법은 데드락 예방 스케쥴을 가능하게 한다. 그러므로 DBFG의 스케쥴링은 평균 트랜잭션 응답 시간을 기초로한 록킹과 비슷하게 작동된다는 것을 보여주고, 그 결과 트랜잭션 수나, I/O장치의 수, 페이지 수, 기본 단위의 크기, 충돌 확률 크기, 트랜잭션의 혼합등에 거의 영향을 받지 않으며, 따라서 트랜잭션수가 증가할 때마다 록킹 오버해드는 DBFG 스케쥴링의 오버해드에 2배가 된다. 따라서 DBFG 스케쥴링은 2PL보다 더 좋은 성능을 얻을수 있는 잠재성이 있고, 다중 처리기 환경에서 자료흐름 처리기 할당 전략을 사용하기 때문에 페이지들은 종래의 명령어들 사이에 파이프라인으로 처리되고, 또한 동일한 페이지상에서 명령어는 동일 프로세서상에서 스케쥴 되도록 함으로써 I/O 작업을 감소 시킬수 있다. 따라서 DBFG 스케쥴링은 분산 환경에서 2PL보다 훨씬 우수하게 처리 된다는 것을 알 수 있다. 앞으로 본 논문의 연구 방향으로서는 다양한 데이타 흐름 처리기 할당 기법과 인덱스를 갖는 DBFG를 구현하는 원리와 DBFG 골격을 갖는 다양한 트랜잭션 우선순위를 허락하는 방법의 연구가 진행 되어져야 할 것이다.

참 고 문 헌

[1] Agrawal, R. Carey, M. J., and Livny, M. Models for studying concurrency control performance: alternatives and implications. In Proceedings of ACM-SIGMOD 1985 International Conference on Management of Data(Austin, Tex.,May 1985), ACM, New York, 1985, pp.108-121.

[2] A Dan,and P.S.Yu, "Performance analysis of buffer coherency policies in a mulisystem data sharing environment," IEEE Trans. Parallel Distributed syst., vol.4, no.3, pp.289-305, Mar. 1993.

[3] P.A.Franaszek, J.T.Robinson, and A.Tomasian, "Concurrency control forhigh contention environment," ACM Trans. Database Syst., Vol.17, no.2, pp. 304-345, 1992.

[4] Henry F. Korth., and Avi Silberschartz, "DataBase System Concepts", McGraw-Hill, 1992.

[5] Mukesh Singhal, and Niranjan G. Shivaratri, "Advanced Concepts In Operating Systems", McGraw-Hill, 1994.

[6] Ozsu,M. T. and P. Valduriez, "Principles of Distributed Database System,"Prentice-Hill, NJ, 1991.

[7] Y.C.Tay, N.Goodman,and R.Suri, "Locking performance in centralized databases," ACM Trans. Database Syst., Vol.10, no.4, pp.415-462, Dec. 1992.

[8] A.Thomasian and I.K.Ryu, "Performance Analysis of two-phase locking," IEEE Trans. Software Eng., vol.17, no.5, pp.386-401, May 1991.

[9] 남태희외, "지연된 블록킹 방법을 사용한 동시성 제어기법의 성능분석에 관한 연구", 한국 사무자동화학회지,1995.

[10] 문송천, "데이타베이스 시스템 총론", 형설출판사, 1993.

[11] 박 석, "데이타베이스 시스템", 흥릉과학출판사, 1993.

[12] 백두권 외, "데이타베이스 구조", 상조사, 1994.

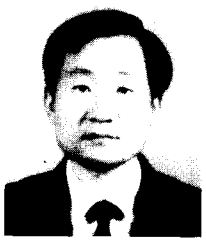
[13] 이병욱, "데이타베이스 시스템", 생능, 1993.

□ 著者紹介

남 태 희

1989년 경성대학교 경영학과 졸업
1992년 경성대학교 산업정보학과 석사
1989~1992년 우성전산 직업 훈련원 전산실장
1993년~현재 동주여자전문대학 무역사무자동화과 전임강사
현재 부산수산대학교 대학원 전자공학과 박사과정

※ 관심 분야 : 데이타베이스, 패턴인식, 화상통신, 멀티미디어



위 승 민

1986년 2월 한국해양대학 항해학과 졸업
1993년 2월 경성대학교 산업정보학과 석사
현재 한국해양대학교 대학원 해사수송과학과 박사과정

※ 관심분야 : 데이타베이스, 멀티미디어, 스케줄링