□특 집□

실시간 데이타베이스에서 동시성 제어 기법

한국과학기술원 홍석희* · 김명호** · 이윤준**

목 차

I. 실시간 데이타베이스 시스템의 소개

실시간 데이타베이스 시스템(Real-Time Database System)의 트랜잭션들은 주어진 시간제약을 준수하여 수행을 해야한다. 이와 같은 시간 제약을 데드라인이라고 하며 각 트랜잭션은 이 데드라인내에 수행을 완료해야 한다[8,11,12].

실시간 데이타베이스 시스템의 트랜잭션이 일반 데이타베이스 시스템의 트랜잭션과 구별될 수 있는 점은 데드라인과 같은 시간 개념을 가진다는 것이다. 실시간 데이타베이스 시스템의 응용 분야는 컴퓨터에 의한 제조, 항공관제 시스템, 교통통제 시스템, 군사 제어 시스템과 같이 점점 다양하게 확대되고 있다. 이같은 실시간 데이타베이스 시스템에 일반 데이타베이스 분야에서 연구된 많은 기법들을 직접 적용시키기는 쉽지 않다. 일반 데이타베이스 시스템에서 성능평가의 기준이 되는 것은 트랜잭션들의 평균 응답시간인데 비해 실시간 데이타베이스 시스템은 데드라인내에 수행을 완료한 트랜잭션들의 수로써 성능을 평가한다. 따라서, 일반 데이타베이스 시스템에서는 일반적으로 모든 트랜잭션이 동일한 조건하에서 수행되지만 실시간 데이타베이스의 트랜잭션들은 각각 만족해야 하는 데드라인이 다르기 때문에 동일한 조건으로 모든 트랜잭션을 수행할 필요가 없다. 예를 들어, 두 트랜잭션 T_1과 T_2를 수행하는데에 있어서 T_1의 데드라인보다 T_2의 데드라인이 선행한다면 실시간 데이타베이스 환경에서는 T_1보다는 T_2를 더 유리한 조건으로 수행시킴으로써 트랜잭션 T_2가 T_1보다 먼저 수행을 완료할 기회를 가질 수 있도록 한다.

데이타베이스 시스템에서 스케쥴링의 대상이 되는 것은 CPU, 버퍼, 디스크, 트랜잭션 등이 될 수 있다. 일반 데이타베이스 시스템의 스케쥴링 기법들은 모든 트랜잭션을 동일한 조건으로 취급하기 때문에 각 트랜잭션의 데드라인을 고려해야 하는 실시간 데이타베이스 시스템에는 적합하지 않다. 이와같은 CPU, 버퍼, 디스크, 트랜잭션 등에 대한 실시간 데이타베이스 시스템의 스케쥴링 기법들은 많이 연구되어 왔다. 실시간 데이타베이스 시스템의 스케쥴링 기법들은 각 트랜잭션에 할당된 우선 순위에 따라서 스케쥴링을 하게 한다. 각 트랜잭션에 할당되는 우선 순위는 트랜잭션의 데드라인, 데드라인까지 남은 여유시간, 트랜잭션의 중요도 등 트랜잭션에 관련된 여러가지 요소들을 포함한다. 우선 순위를 기반으로 하는 CPU 스케쥴링은 운영체제 분야에서 많은 연구가 이루어

* 준회원
** 종신회원

졌다. CPU 스케쥴링 기법들 중 하나로 상위의 우선 순위를 갖는 프로세스가 하위의 우선 순위를 갖는 프로세스를 preemption함으로써 상위의 프로세스가 빨리 수행을 완료하도록 하는 우선 순위 기반 preemptive 스케쥴링 기법이 있다[3,4]. 또한, 프로세스들 간의 preemption을 허용하지 않고 각 프로세스에게 주어진 time-slice단위로 context 전환이 이루어지는 우선 순위 기반 non-preemptive 스케쥴링 기법도 많은 연구가 이루어지고 있다.

데이타 버퍼링은 디스크에 저장되어 있는 데이타베이스의 일부를 비휘발성 메모리에 유지하여 디스크 접근 횟수를 감소시킴으로써 트랜잭션의 응답시간을 단축시키도록 한다. 그러나, 기존에 연구되어진 버퍼링 기법들은 각 트랜잭션의 우선 순위가 아니라 각 버퍼에 대한 접근 성향만으로 버퍼를 스케쥴링하였다. 그러나, 버퍼에 대한 스케쥴링을 각 트랜잭션의 우선 순위에 따라서 수행함으로써 데드라인내에 수행을 완료하는 트랜잭션의 수를 증가시킬 수 있다. 이와 같은 우선 순위를 기반으로 하는 버퍼링 기법은 기존의 LRU 버퍼링 기법에서 버퍼를 소유하는 트랜잭션들의 우선 순위에 따라서 버퍼를 대체함으로써 상위의 우선 순위를 갖는 트랜잭션이 필요로 하는 버퍼를 메모리에서 제거하지 않도록 한다[3,12].

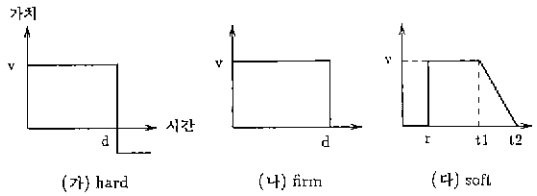
디스크를 접근하는 시간은 CPU의 실행시간에 비하여 상대적으로 상당히 길기때문에 실시간 데이타베이스 시스템에서 디스크 스케쥴링은 중요한 연구 대상이다. 운영체제 분야에서 이미 많은 연구가 되었던 First Come First Serve(FCFS), Shortest Seek-time First(SSF), 엘리베이터 알고리즘 등은 평균적인 탐색거리를 단축시키는 것을 목표로 한다. 그러나, 실시간 데이타베이스 시스템에서 디스크 스케쥴링은 요청된 디스크 섹터와 현재의 디스크 암사이의 거리뿐 아니라 디스크 섹터를 요청한 트랜잭션의 우선 순위가 고려된다[3,6,12]. 많은 우선 순위 기반 디스크 스케쥴링 기법들은 디스크 섹터 요청큐를 우선 순위에 따라서 유지하고 임의의 우선 순위를 갖는 디스크 섹터들 사이에서 FCFS, SSF, 엘리베이터 기법 등의 알고리즘을 적용하는 방법을 사용한다.

실시간 데이타베이스 환경에서는 CPU, 버퍼, 디스크 등의 스케쥴링보다는 트랜잭션 스케쥴링에 더 많은 중점을 두고있다. 실시간 데이타베이스 시스템의 트랜잭션 스케쥴링 기법은 기존의 실시간 시스템에서 이미 많이 연구된 실시간 타스크 스케쥴링 기법을 응용하여 연구가 되었다. 실시간 트랜잭션들을 스케쥴링하는데에 있어서, 공유되는 데이타에 대한 충돌 해결 방법인 동시성 제어 기법의 연구는 일반 데이타베이스 시스템의 동시성 제어 기법을 기반으로 많은 연구가 진행되고 있다[1,5].

본 원고는 이와 같은 실시간 데이타베이스 시스템에서 시간 제약 모델링과 트랜잭션들에 대한 다양한 동시성 제어 기법을 알아본다. 2장에서는 실시간 데이타베이스에서 중요한 개념인 시간제약의 모델링에 대해서 기술하고 3장에서 실시간 트랜잭션들에 대한 다양한 동시성 제어 기법을 알아보고 4장에서 결론으로 끝을 맺고자 한다.

II. 시간제약의 모델링

실시간 트랜잭션이 일반 트랜잭션과 구별될 수 있는 가장 큰 특징은 트랜잭션의 수행에 시간제약이 포함되어야 한다는 점이다. Stankovic은 실시간 트랜잭션의 시간 제약을 표현하기 위해서 가치함수(value function)를 사용하였다[1]. 시간에 따라서 결과가 결정되는 가치함수의 값은 트랜잭션을 완료한 후에 실시간 데이타베이스 시스템에게 기여한 이윤의 정도를 표현한다. 따라서, 시간 t에서 임의의 트랜잭션 T에 대한 가치함수의 결과는 시간 t까지 트랜잭션 T가 시스템에게 기여한 이윤의 정도를 나타낸다. 일반적으로 시간 t에서 가치함수의 값이 클수록 시스템에 많은 이윤을 제공한 것으로 볼 수 있다. 실시간 데이타베이스 시스템의 트랜잭션은 hard, firm, soft 등으로 구분될 수 있다. hard 실시간 트랜잭션은 데드라인내에 수행을 완료하지 않는 경우 큰 피해를 낼 수 있는 트랜잭션에 해당하며, firm 실시간 트랜잭션은 데드라인을 지나서도 수행을 완료하지 못한 트랜잭션은 시스템에서 더 이상 수행될 수 없는 것이다. Soft 실시간 트랜잭션은 데드라인에 대한 제약조건을 완화시킴으로써 데드라인을 지나서까지 수행될 수 있다. 그러나 데드라인을 많이 넘을수록 soft 실시간 트랜잭션의 가치는 점점 감소하게 된다. 이와같은 hard, firm, soft 실시간 트랜잭션들은 각각 (그림 1)의 (가), (나)와 (다)와 같은 가치함수로써 표현될 수 있다. (그림 1)의 (가)는 hard 실시간 트랜잭션에 대한 가치함수를 표현한 것이다. 이때, d는 데드라인을 나타내고 v는 데드라인 이내에 트랜잭션이 완료한 경우에 시스템에 제공하는 이윤을 나타낸다. 그러나, d 이후까지 트랜잭션이 수행을 한다면 시스템에 음수의 가치를 제공함으로써 시스템에 피해를 주게된다. (그림



(그림 1) 실시간 트랜잭션의 구분

1)의 (나)는 firm 실시간 트랜잭션에 대한 가치함수이며 이는 (가)와 유사하지만 d 이후에 트랜잭션이 완료한다면 음수의 가치보다는 0의 가치를 표현함으로써 시스템에 어떤 이윤도 제공하지 못함을 나타낸다. 따라서, firm 실시간 트랜잭션이 데드라인을 지났다면 시스템에 어떤 이윤도 제공하지 않기 때문에 더 이상 시스템내에서 수행할 필요가 없다. Soft 실시간 트랜잭션의 가치함수는 (그림 1)의 (다)와 같이 표현할 수 있다. r은 트랜잭션의 수행이 시작된 시간을 나타낸다. Soft 실시간 트랜잭션은 $r \le t \le t_1$의 시간에는 v의 가치를 시스템에게 제공하지만 $t_1 < t \le t_2$의 시간에는 t가 증가됨에 따라서 트랜잭션의 가치가 감소된다. 또한, 트랜잭션의 수행시작 시간에도 제약을 가함으로써 t>r의 시간에 수행하는 경우 0의 가치를 제공하는 것으로 표현하였다. 그림에서 soft 실시간 트랜잭션을 $r \le t \le t_1$에 수행을 완료하는 경우에 시스템에 가장 최대의 가치인 v를 제공하는 것으로 가정하였다.

III. 실시간 동시성 제어 기법

실시간 트랜잭션들에 대한 동시성 제어는 일반 데이타베이스 시스템에서 많이 연구되어 왔던 2PL(2 Phase Locking)이나 낙관적인(optimistic) 방법 등을 기반으로 많은 연구가 진행되었다. 이와같은 동시성 제어 기법들은 공유되는 데이타를 접근하는 트랜잭션들간의 데이타 충돌을 해결하기 위하여, 2PL인 경우 블록킹을 사용하며, 낙관적인 방법의 경우 데이타 충돌에 포함된 트랜잭션들을 롤백시키는 방법을 사용한다. 일반적인 데이타베이스 시스템의 동시성 제어 기법에서 트랜잭션들간의 데이타 충돌을 해결하는데 있어서 각 트랜잭션들이 충돌을 일으키는 연산의 수행순서가 중요한 요인이 된다. 그러나, 실시간 데이타베이스 시스템에서 동시성 제어 기법들은 각 트랜잭션의 수행순서뿐 아니라 우선 순위도 고려한다. 본 장은 기존의 데이타베이스 시스템에서 이미 많은 연구가 이루어졌던 로킹 기법, 낙관적인 기법, 다중 버젼 제어 기법 등을 기반으로 하는 여러가지 실시간 동시성 제어 기법에 대해서 알아본다.

3.1 로킹 기법 기반 실시간 동시성 제어 기법

일반 데이타베이스 시스템에서 많이 사용되고 있으며 일반적인 상황에서 성능이 우수한 것으로 알려진 2PL을 실시간 트랜잭션의 동시성 제어 기법에 적용시킨 연구로는 Abbott[12]의 2PL-HP 알고리즘과 2PL에 Ordered Sharing이라는 기법을 적용시킨 Agrawal[2]의 2PL-OS 알고리즘 등이 있다.

Abbott의 동시성 제어 알고리즘은 우선 순위를 결정하기 위하여 수행할 트랜잭션에 대한 정보를 이용한다. 동시성 제어에 이용할 트랜잭션에 대한 정보는 트랜잭션의 수행 시작 시간인 r, 데드라인을 나타내는 d, 트랜잭션의 수행 예상 시간을 나타내는 E 등으로 구성된다. 일반적으로 트랜잭션의 수행 예상 시간인 E는 사실상 정확히 구할 수 없지만 동일한 트랜잭션을 계속 반복 수행하는 경우에는 트랜잭션의 수행 예상 시간을 비교적 정확히 측정할 수 있다. Abbott의 동시성 제어 기법은 트랜잭션들의 E를 사용하기 때문에 동시성 제어 기법의 성능은 E의 값에 많은 영향을 받는다. Abbott의 동시성 제어 기법은 실시간 데이타베이스 시스템을 주기억 장치 데이타베이스로 가정하며 데이타에 대한 연산은 수정 연산만을 고려한다. Abbott의 연구는 트랜잭션에 우선 순위를 할당하는 방법으로 First Come First Serve(FCFS), Earliest Deadine(ED), Least Slack(LS) 등의 방법을 사용한다.

(1) First Come First Serve

트랜잭션의 수행시작 시간이 작을수록 높은 우선 순위를 할당한다. 즉, 시스템에서 먼저 수행된 트랜잭션이 높은 우선 순위를 갖는다. 따라서, 이 방법은 시스템에서 최근에 수행을 시작한 트랜잭션 보다는 데드라인까지는 아직 여유가 있지만 시스템에서 수행을 먼저 시작한 트랜잭션을 선호하게 된다.

(2) Earliest Deadine

데드라인에 근접할수록 높은 우선 순위를 할당한다. 트랜잭션의 데드라인을 우선 순위에 고려하기 때문에 FCFS 방법보다는 데드라인이내에 수행을 완료할 가능성이 높다. 그러나, 데드라인을 이미 넘은 트랜잭션이나 거의 데드라인에 근접한 트랜잭션이 높은 우선 순위를 할당받기 때문에 데드라인이내에 수행을 완

료할 수 없는 트랜잭션이라도 높은 우선 순위을 할당받게 된다. 따라서, 데드라인을 만족할 수 있는 다른 트랜잭션들의 수행을 방해할 수 있다는 문제점을 가지고 있다.

(3) Least Slack

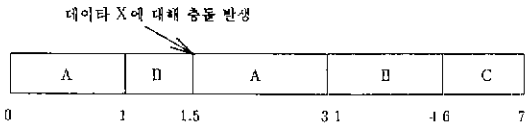
현재시간을 t, 지금까지 트랜잭션에 할당된 시간을 P로 할 경우 각 트랜잭션에 대하여 여유시간(slack time) S를 d-(t+E-P)로 정의하며, S가 적을수록 높은 우선 순위을 할당한다. S는 현재의 시간을 기준으로 데드라인까지 여유시간을 의미한다. 따라서, S≥0인 경우 트랜잭션 T는 정상적으로 시스템의 서비스를 받는다면 데드라인을 만족할 수 있음을 의미한다. 그러나, S<0이라면 트랜잭션 T는 이미 데드라인을 지났기 때문에 시스템으로부터 적당한 조치를 받아야 한다.

이와같은 방법으로 각 트랜잭션에 우선 순위를 할당하는 경우 Abbott의 실시간 동시성 제어 기법은 High Priority 알고리즘과 High Priority 알고리즘을 개선한 Conditional Restart 알고리즘으로 구분된다. (이후로 데이타에 대해 로크를 소유한 트랜잭션을 T_H로 표시하고 같은 데이타에 대해서 로크를 요구한 트랜잭션을 T_R라고 표시하기로 한다.)

3.1.1 High Priority 알고리즘

이 알고리즘은 충돌이 발생한 경우 상위의 우선 순위를 가지는 트랜잭션을 선호하는 방법을 사용한다. 즉, 트랜잭션 T_R이 상위의 우선 순위를 가지면 트랜잭션 T_R이 상위의 우선 순위를 가지면 트랜잭션 T_H를 철회시킨다. 역으로 트랜잭션 T_H가 상위의 우선 순위를 가지는 경우 하위의 우선 순위를 가지는 트랜잭션 T_R은 트랜잭션 T_H가 로크를 해제할 때까지 기다린다. 그러나, 이 기법은 Least Slack 방법으로 우선 순위를 할당하는 경우 트랜잭션들의 반복적인 철회를 일으킬 수 있다. 예를 들어, 트랜잭션 T_R이 상위의 우선 순위를 갖고 트랜잭션 T_H가 하위의 우선 순위를 가진 경우 트랜잭션 T_R에 의해서 트랜잭션 T_H가 철회되고 재수행된다. 이 경우 트랜잭션 T_H가 시스템으로부터 서비스를 받지 않았기 때문에 여유 시간 S=d-(t+E-P)에서 P 값이 0가 된다. 결국 트랜잭션 T_H가 현재 로크를 소유하고 있는 트랜잭션 T_R보다 상위의 우선 순위를 가질 수 있으므로 이번에는 트랜잭션 T_R이 T_H에 의해서 철회되어 재수행될 것이다. 마찬가지로 트랜잭션 T_R이 재수행됨으로써 트랜잭션 T_R의 우선 순위가 현재 로크를 소유하고 있는 트랜잭션 T_H보다 상위의 우선 순위를 가질 수 있다. 결국, 이러한 현상으로 트랜잭션 T_R과 T_H 사이에는 반복적인 철회가 발생할 수 있다. 다음은 반복적인 철회문제를 해결하기 위해서 High Priority을 수정한 알고리즘이다. 알고리즘에서 $p(T_H)$는 트랜잭션 T_H의 우선 순위를 표시하며 $p(T_H)^A$는 S의 계산에 P 값을 0으로 함으로써 트랜잭션 T_H가 철회되었다고 가정한 경우의 우선 순위를 나타낸다.

트랜잭션	r	E	d	갱신될 데이타
A	0	2.6	5	X
B	1	2	4	X
C	2	2.1	8	Y



(그림 2) 스케쥴링의 예

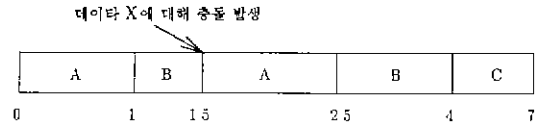
```
if p(T_H)<p(T_R) and p(T_H)^A<p(T_R) then
    T_H를 철회시킨 후에 T_R을 수행시킨다.
else
    T_R의 수행을 중지시키고 T_H의 수행을 계속한다.
```

3개의 트랜잭션 A, B, C에 대하여 Least Slack 방법으로 우선 순위를 할당하고 수정된 High Priority 알고리즘에 의하여 생성된 스케쥴은 (그림 2)와 같다. 1.5의 시간에서 트랜잭션 A가 데이타 X에 대한 로크를 소유한 후에 트랜잭션 B가 같은 데이타에 대해서 로크를 요청했기 때문에 트랜잭션 A와 B는 서로 충돌이 발생하였다 p(B)가 1이고 $p(A)^A$는 0.9이기 때문에 B가 A가 데이타 X에 대한 로크를 해제할 때까지 기다려야 한다. 그러나, High Priority 알고리즘은 데이타 충돌시의 해결방안으로 블럭킹이나 트랜잭션의 철회를 발생하기 때문에 교착 상태를 발생하거나 시스템의 자원을 낭비할 수 있다.

3.1.2 Conditional Restart 알고리즘

High Priority 알고리즘은 우선 순위가 낮은 트랜잭션을 철회하는 경우 지금까지 수행해 온 작업을 상실함으로서 발생하는 자원의 낭비 문제를 가지고 있다. Conditional Restart 알고리즘은 이와같은 자원의 낭비를 줄이기 위해서 데이타 충돌에 포함된

트랜잭션	r	E	d	갱신될 데이타
A	0	2	5	X
B	1	2	4	Y
C	2	3	8	Y



(그림 3) 스케쥴링의 예

하위의 우선 순위를 갖는 트랜잭션의 남은 수행 시간과 상위의 우선 순위를 갖는 트랜잭션의 여유 시간을 고려한다. 다음은 Conditional Restart 알고리즘을 나타낸다.

if $p(T_H)<p(T_R)$ and $p(T_H)^{\wedge}<p(T_R)$ then
  if $E_H-P_H \le S_R$ then
    T_R의 수행을 중지시키고 T_H의 수행을 계속한다.
  else
    T_H를 철회시킨 후에 T_R을 수행시킨다.
else
  T_R의 수행을 중지시키고 T_H의 수행을 계속한다.

알고리즘에서 E_H-P_H은 T_H가 앞으로 수행을 완료하기 위하여 남은 예상 시간이며, S_R은 T_R의 여유 시간을 나타낸다. 즉 T_R의 여유 시간내에 T_H가 수행을 완료할 수 있으면 T_R의 우선 순위가 높아도 T_H가 수행을 완료할 때까지 기다린다. (그림 3)은 트랜잭션들에 대하여 Earliest Deadine 방법으로 우선 순위를 할당하는 경우 Conditional Restart 알고리즘이 생성한 스케쥴의 예를 나타낸다. 1.5의 시간에서 트랜잭션 A가 데이타 X에 대한 로크를 소유한 후에 트랜잭션 B가 같은 데이타에 대해서 로크를 요청했기 때문에 트랜잭션 A와 B는 서로 충돌이 발생하였다. 이 스케쥴에서 B의 여유 시간 S=4−1.5−1.5=1로 A의 남은 수행 시간과 동일하기 때문에 하위의 우선 순위를 갖는 A가 계속 수행을 하고 B는 데이타 X에 대한 로크가 해제될 때까지 기다린다. 이 예에서 모든 트랜잭션이 데드라인내에 수행을 완료한 것을 알 수 있다. Conditional Restart 알고리즘은 트랜잭션의 수행 예상 시간인 E의 값에 종속되기 때문에 E값이 정확하여야 데드라인 이내에 완료할 트랜잭션의 수가 증가될 수 있다.

Agrawal의 동시성 제어 기법인 2PL-OS는 기존의 공유(shared) 로크와 비공유(nonshared) 로크에 ordered-shared라는 새로운 로크를 추가하여 트랜잭션의 블록킹을 줄이는 동시성 제어 기법이다. ordered-shared 로크는 트랜잭션들간에 ordered-shared 로크를 소유한 순서대로 ordered-shared 로크에 대응하는 연산을 수행해야 하는 제약조건을 가진다. 실시간 동시성 제어를 위한 2PL-OS 알고리즘은 ordered-shared 로크를 사용하기 위한 ordered-shared rule에 트랜잭션의 데드라인에 대한 제약을 첨가시켜서 2PL-OS를 수정한 실시간 동시성 제어 알고리즘이다. Agrawal의 성능비교 결과는 실시간 2PL-OS 알고리즘이 2PL-HP 알고리즘보다 우수하다는 것을 보였다[2].

2PL-HP나 2PL-OS 등의 실시간 동시성 제어 기법은 트랜잭션들간의 데이타 충돌을 해결하기 위해서 로크를 사용하기 때문에 실시간 데이타베이스 시스템에서 중요한 문제점중에 하나인 priority inversion이 발생할 수 있다. priority inersion 문제는 상위의 우선 순위를 가지는 트랜잭션이 하위의 우선 순위를 가지는 트랜잭션을 기다리는 상황이다. 또한 이는 데드라인을 만족하는 트랜잭션들의 수를 감소시켜 시스템의 성능을 저하시키는 중요한 요인이 될 수 있다.

3.2 낙관적인 실시간 동시성 제어 기법

Haritsa[5]은 기존의 낙관적인 동시성 제어 기법에 시간 제약을 포함하여 실시간 트랜잭션을 위한 동시성 제어 기법을 제시했다. Haritsa의 낙관적인 기법은 트랜잭션이 수행하는 중에 데드라인을 넘으면 시스템이 그 트랜잭션의 수행을 중지하는 firm 실시간 트랜잭션 시스템을 가정한다. 일반 데이타베이스 시스템을 위한 동시성 제어 기법으로는 보편적으로 사용되며 그 성능이 우수한 것으로 나타나 있는 2PL이다. 그러나, Haritsa의 연구에서는 firm 실시간 트랜잭션 환경을 가정하는 경우 2PL보다 낙관적인 기법이 더 우수하다는 것을 시뮬레이션으로 보였다[5]. Haritsa는 낙관적인 기법에 시간 제약을 포함하여 OPT-SACRIFICE, OPT-WAIT, WAIT-50 등의 세 가지의 알고리즘을 제안하였다.

3.2.1 OPT-SACRIFICE 알고리즘

이 알고리즘은 데이타 충돌에 포함된 하위의 우선 순위를 가지는 트랜잭션들을 철회시키기 때문에 하

위의 우선 순위를 가지는 트랜잭션들을 희생시키게 된다. 다음은 OPT-SACRIFICE를 위한 트랜잭션 T의 validation 단계를 알고리즘으로 나타낸 것이다. CHP (Conflicting Higher Priority)는 validation을 하고 있는 트랜잭션과 충돌이 발생한 상위의 우선 순위를 갖는 트랜잭션들을 나타내며 CLP(Conflicting Lower Priority)는 validation을 하고있는 트랜잭션과 충돌이 발생한 하위의 우선 순위를 갖는 트랜잭션들을 나타낸다.

(1) 만일 데이타 충돌에 포함된 트랜잭션들내에 CHP에 포함된 트랜잭션이 있으며 현재 validation을 하는 트랜잭션을 철회시켜라.

(2) (1)의 경우가 아니라면 데이타 충돌에 포함된 모든 트랜잭션들을 철회시키고 commit하라.

이 알고리즘은 두 가지의 문제점이 있다. 첫번째는 validation중인 트랜잭션 T가 CHP에 포함되는 트랜잭션들에 의해서 철회된 후에 CHP에 포함된 트랜잭션들이 모두 철회된다면 트랜잭션 T는 불필요하게 철회되었기 때문에 T가 지금까지 수행한 작업이 상실된다. 결국, 트랜잭션 T는 시스템의 자원을 낭비한 결과가 된다. 두번째는 트랜잭션의 우선 순위가 트랜잭션이 수행되는 중에도 변화하는 경우 트래잭션의 반복적인 철회가 발생할 수 있다.

3.2.2 OPT-WAIT 알고리즘

이 알고리즘은 validation 단계에서 데이타 충돌에 포함된 상위의 우선 순위를 가지는 모든 트랜잭션이 완료할 때까지 기다리게 함으로써 불필요하게 시스템의 자원을 낭비하지 않도록 한다. validation 단계의 알고리즘은 다음과 같다.

```
while (충돌중인 트랜잭션내에 CHP에 포함된 트랜잭션이 있다) do
    wait;
restart 충돌에 포함된 트랜잭션들;
commit validation을 하고있는 트랜잭션;
```

이 알고리즘은 while 문장에서 CLP에 포함된 트랜잭션들이 수행을 완료할 때까지 기다린다. 그 후에 validation을 하고있는 트랜잭션은 수행을 계속한다. 그러나, CHP에 포함된 트랜잭션이 while 문장 후에 CLP에 포함된 트랜잭션들을 철회시키기 때문에 결국은 모든 CLP에 포함된 모든 트랜잭션들을 철회시키게 된다. OPT-WAIT는 다음과 같은 문제점들을 가지고 있다. 트랜잭션이 while문장에서 기다린 후에 CLP에 포함된 트랜잭션들을 모두 철회시키기 때문에 OPT-SACRIFICE에서도 문제가 되었던 시스템의 자원을 낭비하게 된다. 또한 CHP에 포함된 트랜잭션들이 수행을 완료할 때까지 기다리기 때문에 충돌하고 있는 트랜잭션들의 수를 증가시킴으로써 결국 더 많은 철회를 발생시킬 수 있다.

3.2.3 WAIT-50 알고리즘

이 알고리즘은 OPT-WAIT를 수정하여 데이타 충돌에 포함된 트랜잭션들중의 50% 이상이 CHP에 포함된 트랜잭션들이라면 OPT-WAIT에서 처럼 기다린다. HPpercent를 충돌중인 트랜잭션들 중 CHP에 포함된 트랜잭션들의 비율이라고 정의한 경우 WAIT-50의 validation 알고리즘은 다음과 같다.

```
while (충돌중인 트랜잭션내에 CHP에 포함된 트랜잭션이 있고 HPpercent≥50) do
    wait;
restart 충돌에 포함된 트랜잭션들;
commit validation을 하고있는 트랜잭션;
```

CHP에 포함된 트랜잭션들의 수가 너무 적으면 하위의 우선 순위를 갖는 트랜잭션이라도 CHP에 포함된 트랜잭션들을 철회시킬 수 있다. 따라서, OPT-SACRIFICE나 OPT-WAIT는 하위의 우선 순위를 갖는 트랜잭션들을 항상 철회시키지만 WAIT-50은 하위의 우선 순위를 갖는 트랜잭션이라도 항상 철회시키지는 않는다. WAIT-50을 일반화시키면 X를 HPpercent의 제한값으로 정의할 때 WAIT-X가 된다. 따라서, OPT-WAIT, WAIT-50 등의 알고리즘은 각각 X의 값이 0과 50의 값을 갖는 WAIT-X의 특수한 경우가 된다.

낙관적인 기법과 로킹 제어 기법의 개념을 함께 고려하는 Lin[7]의 알고리즘은 기존의 낙관적인 기법과 같이 트랜잭션이 3단계로 수행되며 데이타의 일관성을 유지하기 위해서 로킹을 사용한다. 2PL은 트랜잭션들이 동시에 수행될 때 직렬화 스케쥴(serializable schedule)이 유지되어야 한다. 일반적으로 2PL로 트랜잭션들의 데이타 충돌을 해결한다면 트랜잭션들의 수행 순서는 트랜잭션들의 데이타 충돌이 발생한 순간에 결정되고 그 순서는 트랜잭션이 수행을 완료할 때까지 변경될 수 없다. 따라서, 우선 순위가 서로 다른 두 트랜잭션이 충돌한 경우 상위의 우선 순위를 갖는 트랜잭션을 선호하기 위해서는 하위의 우선 순위를 갖는 트랜잭션을 철회시켜야 한다.

이와 같은 문제점을 고려할 때 Lin의 알고리즘은

다음과 같은 장점을 가지고 있다. 상위의 우선 순위를 갖는 트랜잭션은 아직 수행을 완료하지 않은 하위의 우선 순위를 갖는 트랜잭션에 의해서 절대로 철회되지 않으며, 하위의 우선 순위를 갖는 트랜잭션이라도 상위의 우선 순위를 갖는 트랜잭션에 의해서 항상 철회되지는 않는다. 또한, 2PL과는 달리 트랜잭션의 수행 순서가 처음으로 데이타 충돌을 일으킨 순서에 따라서 고정되지 않고 트랜잭션의 수행에 따라서 동적으로 변경될 수 있다. 예를 들어, A가 상위의 우선 순위를 가진 트랜잭션인 경우 트랜잭션 A가 데이타 X를 읽기 전에 트랜잭션 B가 데이타 X를 수정했다면 트랜잭션의 수행 순서가 B→A이기 때문에 트랜잭션 A는 B가 데이타 X에 대한 로크를 해제할 때까지 기다리거나 트랜잭션 A를 계속 수행시키려면 트랜잭션 B를 철회할 수 밖에 없다. 그러나, Lin의 알고리즘은 이런 데이타 충돌이 발생한 경우 상위의 우선 순위를 갖는 트랜잭션 A가 수행될 수 있도록 트랜잭션 수행 순서를 A→B로 조정하기 때문에 트랜잭션 A가 기다리거나 트랜잭션 B가 철회될 필요가 없다.

3.3 그외의 실시간 동시성 제어 기법

본 절에서는 로킹을 기반으로 하는 동시성 제어 기법이나 낙관적인 제어 기법과는 다른 방식을 사용하는 실시간 동시성 제어 기법에 대해서 알아보기로 한다. 먼저 Stankovic[11]의 연구는 soft 실시간 데이타베이스 환경하에서 트랜잭션의 데이타 충돌을 해결하기 위해서 각 트랜잭션 마다 가상 시계(virtual clock)를 두었다. 가상 시계의 진행속도가 각 트랜잭션의 우선 순위에 따라서 다르기 때문에 가상 시계의 값이 트랜잭션의 철회나 블럭킹 등을 결정하는 요인이 된다. 실시간 데이타베이스 환경에서 트랜잭션들이 충돌을 한 경우 어느 트랜잭션을 철회나 블럭킹을 시킬것인가는 트랜잭션의 데드라인, 긴급한 정도, 시스템에 있었던 시간, 현재까지 시스템으로부터 받은 서비스의 양, 앞으로 수행을 완료하기까지의 여유 시간 등과 같은 요인에 의해서 결정될 수 있다.

Stankovic의 연구는 가상 시계를 이용한 동시성 제어를 위하여 각 트랜잭션은 긴급한 정도에 따라서 n개의 그룹으로 구분되며 트랜잭션 T에 대한 그룹 번호가 작을수록 트랜잭션 T의 긴급한 정도는 감소된다. 각 트랜잭션 T_i는 고유의 가상 시계를 가지고 있으며, $VC(T_i)$를 트랜잭션 T_i에 대한 가상 시계의 값으로 표기한다. 트랜잭션 T_i가 수행을 시작할 때 $VC(T_i)$는 시스템의 현재 시간이 t로 초기화된다. 그 후로부터 $VC(T_i)$의 값은 트랜잭션 T_i의 그룹 번호를 k라고 할 때 트랜잭션의 그룹에 대해서 다음과 같은 조건을 가지는 B_k의 비율에 따라서 증가한다.

$$B_1 \le B_2 \le B_3 \le \cdots \le B_n$$

즉, 트랜잭션 T에 대한 가상 시계의 값은 T의 긴급한 정도가 클수록 빨리 증가한다. 현재 시간을 t라고 하고 트래잭션 T_i의 수행 시작 시간을 ts_i라고 할 때, 주어진 시간 $t(t \ge ts_i)$에서 가상시계의 값은 다음과 같이 정의된다.

$$VC(T_i) = ts_i + B_k(t - ts_i)$$

트랜잭션 T_H가 데이타 X에 대한 연산을 하려고 할 때 데이타 X에 대한 연산을 하고 있는 트랜잭션이 없으면 T_H는 데이타 X에 대한 연산을 할 수 있다. 그러나, t 시간에 다른 트랜잭션 T_R가 트랜잭션 T_H가 사용하고 있는 데이타 X에 대해서 충돌을 일으키는 연산을 수행하려 한다면 다음과 같은 알고리즘에 따라서 데이타 충돌을 해결한다.

```
if (T_R의 데드라인>T_H의 데드라인) then
  트랜잭션 T_R를 기다리게 한다.
else
  if(VC(T_H)<T_H의 데드라인) then
    트랜잭션 T_H를 철회시킨다;
    트랜잭션 T_R를 수행시킨다;
  else
    트랜잭션 T_R를 기다리게 한다;
```

이 알고리즘은 트랜잭션 T_H가 데이타 X에 대한 연산을 끝낸 후 데이타 X를 사용하기 위해서 기다리고 있는 트랜잭션들 중 가장 작은 데드라인을 가지는 트랜잭션에게 데이타 X에 대한 접근을 허용하는 Least Deadline 방법을 이용한다.

가상 시계를 이용한 동시성 제어 알고리즘은 트랜잭션의 데드라인, 긴급한 정도, 현재까지 시스템에 머무른 시간 등을 고려하였다. 다음은 같은 그룹에 속한 두 트랜잭션 T_H와 T_R가 충돌하는 경우 B_k의 값을 변화시킴으로써 Stankovic의 알고리즘이 어떻게 반응하는가를 나타낸다. 시간 t에 T_H가 데이타 X를 접근하고 있는 중에 T_R가 데이타 X에 대한 접근을 요청한다고 가정하자.

1. $B_k=0$일 때, $VC(T_i)=ts_H : ts_H$는 항상 T_H의 데

드라인보다 작기 때문에 T_H가 이미 데드라인을 넘었어도 만일 T_H의 데드라인보다 T_R의 데드라인이 더 작으면 T_H가 철회된다. 이 경우에 가상 시계 알고리즘은 같은 그룹에 속한 트랜잭션들 간의 데이타 충돌을 Least Deadline 방식으로 해결한다.

2. $B_k=1$일 때, $VC(T_H)=t:T_R$의 데드라인이 T_H의 데드라인보다 작고 T_H가 아직 데드라인을 넘지 않았으면 T_H가 철회된다. 만일, 데드라인을 넘지 않았으면 Least Deadine 방식으로 데이타 충돌을 해결하지만 일단 데드라인을 넘은 트랜잭션은 결코 철회되지 않는다.

3. $B_k=\infty$일 때, $VC(T_H)=\infty$: 이 경우는 $VC(T_H)$가 항상 T_H의 데드라인보다는 크기 때문에 T_H의 데드라인에 상관없이 T_H는 결코 철회되지 않는다. 즉, non-preemptive 방식이 된다.

이와같이 가상 시계 알고리즘은 B_k의 값을 변화시킴으로써 다양한 방식으로 충돌을 해결할 수 있다. 결론적으로 가상 시계 알고리즘은 다음과 같은 특징을 가진다.

1. 트랜잭션이 시스템에서 받은 서비스의 양인 $(t-(ts_i))$을 데이타 충돌 해결시에 사용한다. 만일 트랜잭션이 아직 시스템으로부터 많은 서비스를 받지 않았다면 철회될 가능성도 높아지게 된다. 결국 트랜잭션이 지금까지 수행한 작업의 양이 묵시적으로 고려된 것이다.

2. 트랜잭션의 데드라인이 고려되기 때문에 데이타 충돌시에 현재 데이타를 접근하고 있는 트랜잭션의 데드라인이 더 클때만 철회될 가능성이 있다. 즉, 데드라인이 클수록 철회될 가능성이 증가하게 된다.

3. 현재 데이타를 접근하고 있는 트랜잭션 T_H의 긴급한 정도가 데이타 충돌을 해결하는데 사용된다. 트랜잭션의 긴급한 정도가 증가할수록 $VC(T_H)$의 값이 증가하는 속도가 빨라진다. 따라서 긴급한 트랜잭션일수록 $VC(T_H)$의 값이 데드라인을 빨리 넘기 때문에 그만큼 철회될 가능성이 감소된다.

4. 이 알고리즘은 데이타 충돌을 해결할 때 데이타의 접근을 요구하는 트랜잭션 T_R의 긴급한 정도는 전혀 고려하지 않는다. 즉, 현재 데이타를 사용하고 있는 트랜잭션 T_H보다 데이타 접근을 요구하는 트랜잭션이 더 긴급하더라도 트랜잭션 T_H가 데이타의 사용을 마칠때까지 기다려야 한다.

Kim[6]의 연구에서는 기존에 일반 데이타베이스 환경을 위한 다중 버전 동시성 제어 기법을 실시간 데이타베이스 환경에 적합하도록 수정한 기법을 제안하였다. 본 원고에서는 각 데이타에 2개의 버전만을 허용하는 실시간 트랜잭션을 위한 이중 버전 2PL 제어 기법을 살펴보기로 한다. 이중 버전 2PL은 read, write, certify 로크 등 세 가지의 로크를 이용하여 데이타 충돌을 해결한다. (그림 4)는 이들 로크에 대한 호환성표를 나타낸다. read와 write 로크는 기존의 2PL 제어 기법과 동일한 의미를 가진다. 트랜잭션의 판독 연산은 항상 commit된 트랜잭션이 기록한 데이타에 대해서만 수행되기 때문에 연속적인 철회(cascading abort) 문제를 발생하지 않게 된다. 트랜잭션의 commit시에 그 트랜잭션이 소유한 모든 write 로크를 certify 로크로 변환시켜야 한다. 그러나, 이미 다른 트랜잭션이 read 로크를 소유하고 있는 데이타에 대해서 로크 변환을 하려면 read 로크가 해제될 때까지 기다려야 한다. 즉, 트랜잭션이 certify 로크를 소유하기 위해서는 현재 데이타에 대한 판독 연산을 수행하고 있는 트랜잭션이 존재하지 않을때까지 기다려야 한다. 결국 이와 같은 상황은 실시간 데이타베이스 환경에서 해결해야 될 문제인 priority inversion을 발생시킨다. 이중 버전 2PL 제어 기법이 실시간 데이타베이스 환경에서 직접 사용된다면 priority inversion 문제를 일으킬 수 있는 여러가지 상황이 발생한다.

T_R가 로크를 요청

T_H가 로크를 소유	read	write	certify
read	Y	Y	N
write	Y	N	–
certify	N	N	–

(그림 4) 이중 버전 2PL을 위한 로크 호환성표

예를 들어, 상위의 우선 순위를 가지는 트랜잭션 T_H가 데이타 A에 대해서 write 로크를 소유한 후에 하위의 우선 순위를 가지는 트랜잭션 T_L이 같은 데이타에 대해서 read 로크를 요청한다고 하자. T_H의 write 로크와 T_L의 read 로크는 서로 호환하기 때문에 T_L의 read 로크는 허용된다. T_H가 commit하는 경우 T_L이 read 로크를 해제할 때까지 기다려야 한다. 따라서, T_H는 T_L의 수행이 완료할 때까지 기다려야 하기 때문에 데드라인 이내에 수행을 완료하지 못할수도 있다. 또 다른 예로써, T_L이 데이타 A에 대해서 read 로크를 소유한 후에 T_H가 write 로크를 요청한다고 하자. T_L의 read 로크와 T_H의 write 로크가 서로 호

T_R가 로크를 요청

T_H이 로크를 소유	read	write	certify
read	Y	A	–
write	Y	A	–
certify	$\bar{Y}$	A	–

(가) $P(T_R) > P(T_H)$

T_R

T_H	read	write	certify
read	Y	Y	N
write	N	N	–
certify	N	N	–

(나) $P(T_R) < P(T_H)$

(그림 5) 실시간 이중 버전 2PL을 위한 로크 호환성표

환하므로 T_H의 write 로크는 허용된다. 앞의 예와 동일하게 T_H가 commit하기 위하여는 T_L이 read 로크를 해제할 때까지 기다려야 한다.

이중 버전 2PL 제어 기법의 이와 같은 priority inversion 문제는 트랜잭션 T_H가 T_L보다 상위의 우선 순위를 가질 때, T_H가 T_L이 소유한 로크와 호환하지 않는 로크를 요청한다면 T_L을 철회시킴으로써 해결될 수 있다. 그러나, 어떤 상황에서는 하위의 우선 순위를 가지는 트랜잭션 T_L을 철회시킬 필요가 없을수도 있다. 예를 들어, T_L이 데이타 A에 대해서 certify 로크를 소유했고 같은 데이타에 대해서 T_H가 read 로크를 요청했다고 하자. 이런 경우 T_L을 철회시키기 보다는 T_L의 certify 로크를 write 로크로 변환시킨다면 T_H는 데이타 A에 대한 read 로크를 소유할 수 있으며 하위의 우선 순위를 가지는 T_L을 철회시킬 필요가 없다. (그림 5)는 실시간 동시성 제어를 위한 이중 버전 2 PL의 로크 호환성표를 나타낸다. (그림 5)의 (가)는 상위의 우선 순위를 가지는 트랜잭션 T_R가 로크를 요청하는 경우이고 (나)는 하위의 우선 순위를 가지는 트랜잭션 T_R가 로크를 요청하는 경우를 나타낸다. 로크 호환성표에서 A는 로크를 소유하고 있는 트랜잭션이 철회되는 것을 나타내며 Y는 트랜잭션이 소유한 로크를 로크 변환이전의 로크로 역변환됨을 나타낸다. 로크 호환성표에서 나타나있는 것처럼 상위의 우선 순위를 가지는 트랜잭션 T_R가 write 로크를 요청하는 경우 하위의 우선 순위를 가지는 트랜잭션 T_H이 어떤 로크를 소유하고 있더라도 철회된다. 또한, T_H이 certify 로크를 소유하고 있고 T_R가 read 로크를 요청하는 경우 T_H가 소유한 certify 로크는 write 로크로 역변환되어 T_R가 read 로크를 소유할 수 있도록 한다.

(그림 5)의 로크 호환성표에 따라서 동시성 제어를 하는 이중 버전 2PL 기법은 priority inverison 문제를 해결했지만 시스템의 자원을 낭비할 수 있다. certify 로크를 소유하고 있는 하위의 우선 순위를 가지는 트랜잭션 T_L은 commit단계를 수행하고 있음을 알 수 있다. 그러나, (그림 5)의 로크 호환성표 (가)에 의해서 상위의 우선 순위를 가지는 트랜잭션 T_R는 write 로크를 소유하기 위해서 T_H을 철회시킨다. 이와 같은 경우 T_H이 철회되기 전까지 수행한 작업이 상실되기 때문에 시스템의 자원을 낭비한 결과가 된다. 만일, 하위의 우선 순위를 가지는 트랜잭션 T_H이 certify 로크를 소유하고 있다면 상위의 우선 순위를 가지는 트랜잭션 T_R가 T_H를 철회시키지 않고 T_H의 수행이 완료할 때까지 기다릴 수 있다면 시스템의 자원을 낭비하지 않을 수도 있다. T_R가 T_H을 기다리는 상황은 priority inversion 문제를 발생시키지만 T_H이 commit 단계에 있기 때문에 T_R가 기다리는 시간을 길지 않게 된다.

IV. 결 론

실시간 동시성 제어 기법은 기존의 일반 데이타베이스 시스템에서 많이 연구되어진 2PL과 낙관적인 기법을 기반으로 연구되었다. 특히, 실시간 트랜잭션은 데드라인 이내에 수행을 완료해야 하는 제약 조건으로 인해서 기존의 동시성 제어 기법과는 다른 요인을 가진다. 이는 실시간 데이타베이스의 성능은 트랜잭션의 평균 응답 시간보다는 데드라인 이내에 수행을 완료한 트랜잭션들의 수로써 평가되기 때문에 각 트랜잭션의 데드라인이 동시성 제어에 중요하게 고려되어야 한다. 실시간 동시성 제어 기법은 각 트랜잭션에 대한 정보를 우선 순위로 표현하여 이를 트랜잭션들간의 데이타 충돌을 해결하는데 이용하여야 한다. 실시간 동시성 제어 기법에서 고려되어야 할 사항들을 요약하면 다음과 같다.

1. 실시간 트랜잭션들간에 데이타 충돌을 해결하기

위해서 priority inversion 문제는 실시간 데이타베이스 시스템의 성능을 저하시키므로 실시간 동시성 제어 기법에서 중요하게 고려되어야 한다.

2. 실시간 데이타베이스 환경에서 트랜잭션의 수행은 시스템의 자원을 사용하게 되기 때문에 트랜잭션의 철회는 시스템의 자원을 낭비하는 결과가 된다. 따라서, 만일 하위의 우선 순위를 가지는 트랜잭션의 수행 완료가 얼마 안남았다면 하위의 우선 순위를 갖는 트랜잭션을 철회시키기 보다는 상위의 우선 순위를 갖는 트랜잭션을 기다리게 하는 것이 바람직할 수 있다.

3. 데이타의 일관성을 유지하기 위해서 로크를 사용하는 경우 트랜잭션들간의 수행 순서는 최초로 데이타 충돌을 발생한 순간에 결정되어 트랜잭션의 수행이 완료할 때까지 고정된다. 따라서, 하위의 우선 순위를 가지는 트랜잭션의 수행 순서가 상위의 우선 순위를 가지는 트랜잭션보다 선행한다면 상위의 우선 순위를 가지는 트랜잭션은 데드라인 이내에 수행을 완료하지 못할수도 있다. 이와 같은 경우에 트랜잭션들간의 수행 순서를 동적으로 변화시킴으로써 상위의 우선 순위를 가지는 트랜잭션이 하위의 우선 순위를 가지는 트랜잭션보다 선행하여 수행시킬 수 있다면 데드라인 이내에 완료되는 트랜잭션의 수를 증가시킬 수 있다.

참 고 문 헌

1 Abbott, R. and H. Garcia-Molina, "Scheduling Real-Time Transactions with Disk Resident Data," Proc. of the 17th Int'l Conf. on Very Large Data Bases, Amsterdam, Aug., 1989.

2. Agrawal, D., A. E. Abbadi and R. Jeffers, "Using Delayed Commitment in Locking Protocols for Real-Time Databases," Proc. of ACM-SIGMOD Int'l. Conf. on Management of Data, 1992.

3. Carey, M. J., R. Jauhari and M. Livny, "Priority in DBMS Resource Scheduling," Proc. of the 15th Int'l Conf. on Very Large Data Bases. Amsterdam, Aug., 1989.

4. Furht, B., *et al.*, "Real-Time UNIX Systems Design and Application Guide," Kluwer Academic Publishers, 1991.

5. Haritsa, J. R., Carey, M. J. and Livny, M. "Dynamic Real-Time Optimistic Concurrency Control," Proc. of the 11th IEEE Real-Time Systems Symposium, Florida, Dec., 1990.

6. Kim, W. S. and Srivastava, J. "Enhancing Real-Time DBMS Performance with Multiversion Data and Priority Based Disk Scheduling," Proc. of the 12th IEEE Real-Time Systems Symposium, Texas, Dec., 1991.

7. Lin, Y. and Son, S. H. "Concurrency Control in Real-Time Databases by Dynamic Adjustment of Serialization Order," Proc. of the 11th IEEE Real-Time Systems Symposium, Florida, Dec., 1990.

8. Sighal, M., "Issues and Approaches to Design of Real-Time Database Systems," SIGMOD Record, Vol. 17, No. 1, March. 1988.

9. Son, S. H. and Kang, H. "Approaches to Design of Real-Time Database Systems," Proc. of Int'l Symposium on Database Systems for Advanced Applications, Seoul, Korea, April, 1989.

10. Son. S. H., "Real-Time Database Systems: A New Challenge." Data Engineering, Vol. 13, No. 4, 1990.

11. Stankovic, J. A., "On Real-Time Database Systems: A New Challenge," Data Engineering, Vol. 13, No. 4, 1990.

12. Stankovic, J. A., K. Ramamritham and D. Towsley, "Scheduling in Real-Time Transaction Systems," Foundations of Real-Time Computing: Scheduling and Resource Management, Kluwer Academic Publishers, 1991.

홍 석 희

1989 홍익대학교 공과대학 전자계산학과 졸업
1991 한국과학기술원 전산학과 석사학위 취득
1991 ~현재 한국과학기술원 전산학과 박사과정 재학중
관심 분야 : 실시간 데이타베이스 시스템, 주기억 장치 데이타베이스 시스템, 트랜잭션 처리 시스템

김 명 호

1982 서울대학교 공과대학 전자계산기공학과 졸업
1984 서울대학교 공과대학 전자계산기공학과에서 석사학위 취득
1989 미시간 주립대학교에서 박사학위 취득
1992 ~현재 한국과학기술원 전산학과 조교수
관심 분야 : 데이타베이스 시스템, 실시간 데이타베이스, 병렬처리와 분산 시스템 등

이 윤 준

1977 서울대학교 계산통계학과 졸업
1979 한국과학기술원 전산학과 석사학위 취득
1983 France, INPG-ENSIMAG 박사학위 취득
1983 ~1984 France, IMAG 연구원
1984 ~현재 한국과학기술원 전산학과 부교수
1989 MCC(미) 초빙연구원
1990 CRIN(불) 객원교수

관심 분야 : 정보검색, 데이타베이스 시스템, 실시간 데이타베이스, ODA 등