분산 실시간 데이터베이스 시스템을 위한 효율적인 동시성제어 기법

이종설, 신재룡, 유재수
충북대학교 정보통신공학과
e-mail:leejs98@pretty.chungbuk.ac.kr

An Efficient Concurrency Control Scheme for Distributed Real-time Database Systems

Jong-Sul Lee, Jae-Ryong Shin, Jae-Soo Yoo
Dept of Computer & Communication Eng., Chungbuk National University

요 약

본 논문에서는 분산 실시간 데이터베이스 시스템을 위한 효율적인 동시성 제어 기법을 제안한다. 제안하는 기법은 분산 실시간 환경에서 완료준비 단계에 도달한 트랜잭션의 우선 순위를 상승시킴으로써 트랜잭션의 재시작에 의한 낭비를 줄이고, 트랜잭션의 완료를 최대한 보장하며, 잠금 지연 시간을 최소화하는 장점을 갖는다. 또한, 제안하는 기법은 우선 순위가 상승된 트랜잭션의 완료를 보장하며 데이터 차용(borrowing)을 통해 다른 트랜잭션의 지연시간을 줄여줌으로써 전체적인 시스템 성능을 향상시킨다.

1. 서 론

컴퓨터 네트워크의 발달에 따라 통신 시스템과 군사 시스템 그리고 전자상거래와 같은 실시간 데이터베이스 시스템이 분산환경으로 확장되어졌다. 이와 같이 분산 환경으로 확장됨에 따라 동시성제어와 회복 기법 또한 더욱 복잡해졌으며 실시간 데이터베이스의 직렬성 보장과 시스템 성능 향상을 위해 기존의 실시간 동시성제어 기법과 완료 프로토콜을 분산환경에 적용하기 위한 다양한 연구들이 진행되어왔다[1,8,6].

실시간 동시성제어 기법을 분산환경에 적용하기 위한 연구로는 크게 비관적 접근 방법과 낙관적 접근 방법을 분산환경으로 확장하는 것이다. 비관적 접근 방법은 잠금을 이용하는 것으로 분산 동시성제어 방법인 D2PL(Distributed Two Phase Locking)과 DO2PL(Distributed Optimistic Two Phase Locking)을 실시간 동시성제어 기법으로 확장하는 방법과 MIRROR(Managing Isolation in Replicated Real-time Object Repositories) 등이 있다. 낙관적 접근 방법으로는 OCC-Sacrifice, OCC-Wait, Wait 50 등을 분산환경으로 확장하는 방법이 있다[1]. 또한, 분산환경의 완료 프로토콜을 개선한 OPT 방법이 있다[4].

본 논문에서는 새로운 분산 동시성제어 기법인 우선 순위 상승 프로토콜을 제안한다. 제안하는 프로토콜은 분산 실시간 환경에서 완료준비 단계에 도달한 트랜잭션의 우선 순위를 상승시킴으로써 트랜잭션의 재시작에 의한 낭비를 줄인다. 뿐만 아니라 트

본 연구는 과학재단 특정기초과제(과제번호:"1999-1-303-007-3") 연구비 지원에 의하여 수행되었음.

랜잭션의 완료를 최대한 보장하며, 잠금 지연 시간을 최소화하는 장점을 갖는다.

또한 트랜잭션의 완료를 최대한 보장하기 때문에 점유하고 있는 데이터가 정상적으로 데이터베이스에 반영될 것이다. 따라서 다른 트랜잭션이 해당 데이터를 필요로 하는 경우 데이터 차용을 통해 다른 트랜잭션의 지연시간을 줄여줌으로써 전체적인 시스템 효율 향상이 가능하다.

논문의 구성은 다음과 같다. 2장에서는 분산 실시간 동시성제어 기법과 분산 실시간 완료 프로토콜에 대한 기존 연구를 살펴보고, 3장에서는 새롭게 제안하는 우선 순위 상승 프로토콜에 대해 기술한다. 4장에서는 기존 프로토콜과의 성능평가를 통해 제안한 동시성 제어 기법의 우수성을 입증한다. 마지막 5장에서는 결론을 맺고 향후 연구 방향을 제시한다.

2. 관련 연구

일반적으로 분산환경에서는 트랜잭션이 제시된 주 사이트에서 실행되는 master 트랜잭션(M)과 다른 여러 사이트에서 실행되는 cohort 트랜잭션(C_i)들로 트랜잭션이 구성된다. 데이터는 중복될 수 있으므로 데이터를 변경하는 모든 cohort는 다른 사이트에 대해 중복된 데이터를 변경하는 하나 이상의 remote update 트랜잭션(U_{ij})들을 갖는다. 따라서 본 논문에서도 중복을 허용하는 분산환경을 고려하였다. 특히 cohort는 데이터베이스 시스템의 동시성 보장을 위해 완료준비 단계의 시작 부분에서 prepare 메시지와 변경할 데이터의 사본을 remote update에 전달한다.

이와 같은 분산환경의 동시성제어를 위해 사용되는 여러 기법들을 살펴보면 다음과 같다.

2.1 DO2PL(Distributed Optimistic Two-Phase Locking)

DO2PL은 읽기 단계, 완료 준비 단계, 결정 단계의 세 단계로 이루어진다[2].

읽기 단계에서 cohort는 여러 사이트에 존재하는 데이터 사본 중에서 하나의 사본에 대해 읽기 잠금을 설정한다. 트랜잭션의 실행이 완료된 각 cohort는 해당 master에게 트랜잭션 완료를 통보한다.

완료 준비 단계에서 master의 prepare 메시지가 도착하면 각 cohort는 변경될 데이터 항목과 prepare 메시지를 해당 remote update에 전달한다. 각 remote update에서는 변경할 데이터에 대해서 쓰기 잠금을 설정한 뒤 prepared 상태가 되며 자신의 cohort에게 prepared 메시지를 전달한다. 모든 remote update에서 보내온 prepared 메시지를 받은 cohort는 prepared 상태가 되고, 자신의 master에게 prepared 메시지를 전달한다.

마지막 결정 단계에서 master는 cohort에게 commit 메시지를 전달하고, cohort는 변경할 데이터를 데이터베이스에 반영한다.

DO2PL을 실시간 환경으로 확장하기 위해 실시간 충돌 해결 방법을 적용한다. 실시간 충돌 해결 방법으로 PA(Priority Abort)를 적용할 경우 우선 순위가 낮은 완료 단계의 트랜잭션이 철회될 수 있고, PB(Priority Blocking)를 적용할 경우 완료준비 단계의 장기 트랜잭션에 의해 대기 시간이 길어질 수 있으며, PI(Priority Inheritance)를 적용할 경우에는 너무 복잡해지는 단점이 있다.

2.2 MIRROR(Managing Isolation in Replicated Real- time Object Repositories)

DO2PL을 기반으로 한 MIRROR의 가장 큰 특징은 트랜잭션의 상태에 따라 데이터의 충돌에 대한 해결 방법을 다르게 적용하는 것이다.

cohort와 remote update의 상태는 경계점으로 구분된다. cohort의 경계점은 자신의 master로부터 prepare 메시지를 받았을 때이며, remote update의 경계점은 모든 변경할 데이터 항목에 대한 쓰기 잠금을 획득하였을 때이다.

MIRROR에서는 충돌 해결을 위해 트랜잭션 상태에 따라 경계점 이전에는 PA 방법을 적용하고, 경계점 이후에는 PB 방법을 적용한다.

MIRROR의 특징은 상태 정보를 얻기 위해 사이트간의 추가적인 통신이나 동기화가 불필요하고, 완료 프로토콜의 경우 2단계 완료 프로토콜을 변경할 필요가 없다는 것이다. 따라서 보통의 데이터 변경이 발생하는 중복 분산환경에서 우수한 성능을 보인다.

그러나 경계점 이후에서 PB를 적용함으로써 완료 단계의 기간이 길어질 경우 우선 순위 역전현상이 발생하고, 트랜잭션의 대기 시간이 길어지며 교착

상태가 발생할 수 있다는 단점이 있다.

2.3 OPT(OPTimistic commit protocol)

2단계 완료 프로토콜을 개선한 OPT 프로토콜은 높은 우선 순위의 트랜잭션 T_j가 필요로 하는 데이터가 완료준비 상태인 낮은 우선 순위 트랜잭션 T_i에 의해 점유되었을 경우 트랜잭션 T_j가 사용중인 데이터에 대해서 트랜잭션 T_j에게 해당 데이터의 사본을 빌려준다[4]. 이때 트랜잭션 T_i, T_j를 각각 대여자(lender), 차용자(borrower)라 한다.

OPT 프로토콜은 완료준비 단계의 낮은 우선 순위 트랜잭션의 정상적인 완료를 보장하며, 대부분의 대여자 트랜잭션이 완료된다고 가정했을 때 준비상태인 트랜잭션이 점유한 데이터를 필요로 하는 다른 높은 우선 순위 트랜잭션에게 빌려줌으로써 대기 시간을 줄일 수 있는 장점을 갖는다.

3. 제안하는 동시성제어 기법

3.1 개요

실시간 데이터베이스 시스템에서는 자원에 대한 충돌 해결과 트랜잭션의 시간제약 조건을 만족시키기 위해 각 트랜잭션에 우선 순위를 할당한다. 본 논문에서는 Firm 실시간 데이터베이스를 대상으로 하며 우선 순위 할당 방법으로 마감 시간에 가장 근접한 트랜잭션에 높은 우선 순위를 할당하는 EDF기법을 사용한다. 또한 충돌 해결을 위해 PA와 2PL을 사용하며, 분산 완료 프로토콜인 이 단계 완료 프로토콜을 이용한다.

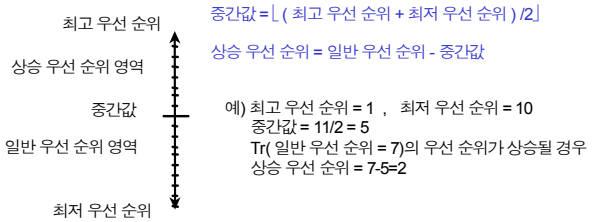
본 논문에서 제안하는 동시성제어 기법은 DO2PL_PA 프로토콜을 기반으로 재시작에 의한 낭비를 줄이기 위해 완료준비 단계인 트랜잭션의 완료를 최대한 보장한다. 이를 위해 우선 순위의 영역을 일반 우선 순위 영역과 상승 우선 순위 영역으로 구분한다.

처음 시작되는 트랜잭션은 일반 우선 순위를 할당받고 읽기 단계에 들어간다. 읽기 단계에서 데이터에 대한 충돌이 발생할 경우 데이터를 점유한 트랜잭션이 상승 우선 순위이면 해당 데이터를 차용하여 처리를 계속한다. 그러나, 데이터를 점유한 트랜잭션이 일반 우선 순위인 트랜잭션일 경우에는 PA를 적용하여 두 트랜잭션간의 충돌을 해결한다. 트랜잭션의 실행이 끝나면 완료 준비 단계에 들어간다. 완료 준비 단계에서 트랜잭션은 완료를 최대한 보장받기 위해서 상승 우선 순위를 재 할당받는다. 상승 우선 순위를 재 할당받은 트랜잭션은 모든 하위 트랜잭션에 대해서 처리 완료 여부를 확인하게 된다. 전체 하위 트랜잭션의 완료 여부가 결정되면, 최종 결정 단계로 들어간다. 결정 단계에서는 전체 트랜잭션의 완료 여부를 판단하여 트랜잭션의 완료 및 철회를 최종적으로 결정하고 그 결과를 각 하위 트랜잭션에게 전달한다.

3.2 우선 순위 영역 분할

우선 순위 할당은 (그림 1)과 같이 중간 값을 기준으로 상승 우선 순위 영역과 일반 우선 순위 영역으로 구분하여 적용한다.

처음 시작되는 트랜잭션의 경우 일반 우선 순위 영역에 할당되고, 트랜잭션이 prepare 메시지를 전달받은 뒤에 완료준비 단계로 들어갈 경우 상승 우선 순위 영역에 새로운 우선 순위를 할당받게 된다. 새롭게 할당받는 우선 순위는 최초 할당받은 일반 우선 순위에서 중간 값을 뺀 값이 된다.

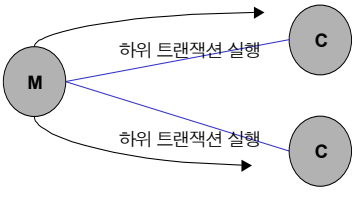


(그림 1) 우선 순위 영역 분할

3.3 읽기 단계

일반 우선 순위를 할당받은 트랜잭션 T_i는 각 cohort들에 대해서 하위 트랜잭션을 실행한다. 트랜잭션 T_i가 읽기 잠금을 요구하는 데이터가 사용되지 않을 경우에는 해당 데이터를 사용하지만, T_j가 필요로 하는 데이터가 일반 우선 순위를 가지는 트랜잭션 T_i에 의해 사용될 경우, 두 트랜잭션에 대해서 PA를 적용하여 충돌을 해결한다. 만약 데이터를 사용중인 트랜잭션 T_j가 상승 우선 순위인 경우에

는 트랜잭션 T_j가 점유한 데이터에 대해 차용한 트랜잭션이 있는지를 확인하고, 이미 데이터를 차용한 트랜잭션이 존재할 경우에는 해당 트랜잭션과 T_i는 PA를 적용하여 충돌을 해결한다. 그러나, 데이터를 차용한 트랜잭션이 존재하지 않을 경우, 트랜잭션 T_i는 완료준비 단계에서 상승 우선 순위를 갖는 트랜잭션 T_j가 점유한 데이터를 차용하여 트랜잭션을 수행한다. 이때 데이터를 차용한 트랜잭션을 차용자 트랜잭션이라고 하고 데이터를 대여한 트랜잭션을 대여자 트랜잭션이라고 한다.




```
if (data items is not used)
    use data;
else if (Normal priority Tj is using data items)
    PA between Ti and Tj ;
else if (Upper priority Tj is using data items) {
    if ( Tj is lending data items ) {
        if ( borrower is waiting ) abort Ti ;
        else PA between Ti and borrower ;
    }
    else Ti borrows data items from Tj ;
}
```

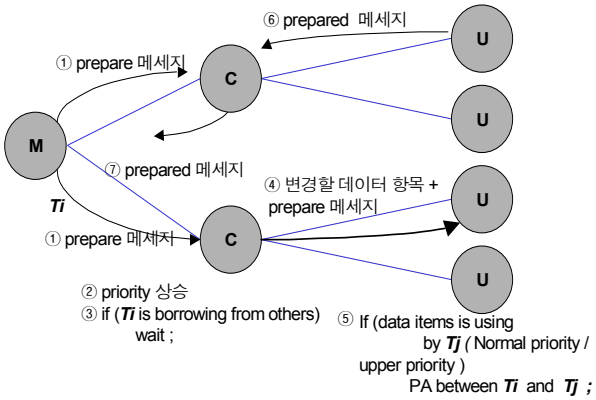
(그림 2) 읽기 단계

3.4 완료 준비 단계

완료 준비 단계에서 일반 우선 순위를 갖는 cohort는 (그림 3)과 같이 prepare 메시지를 전달받은 후, 상승 우선 순위를 재 할당받는다. 상승 우선 순위를 할당받은 cohort는 차용한 데이터의 대여자 트랜잭션의 완료가 결정되지 않았을 경우에는 일정 기간동안 대기하게 된다. 대여자 트랜잭션이 완료된 경우에는, 대기중인 차용자 트랜잭션은 변경할 데이터 항목과 prepare 메시지를 remote update에 전달한다. 만약, 대여자 트랜잭션이 철회되었을 경우에 차용자 트랜잭션은 master 트랜잭션에 abort 메시지를 전달한다. 차용한 데이터가 없을 경우, 변경할 데이터 항목과 prepare 메시지를 remote update에 전달한다.

prepare 메시지를 전달받은 remote update는 변경할 데이터에 대해 쓰기 잠금을 설정한다. 이때 충돌이 발생할 경우 PA를 적용하여 충돌을 해결한다. 모든 데이터에 대해 잠금을 획득한 remote update는 자신의 cohort에게 prepared 메시지를 전달한다.

모든 remote update에서 prepared 메시지를 전달받은 cohort는 master에게 prepared 메시지를 전달한다.

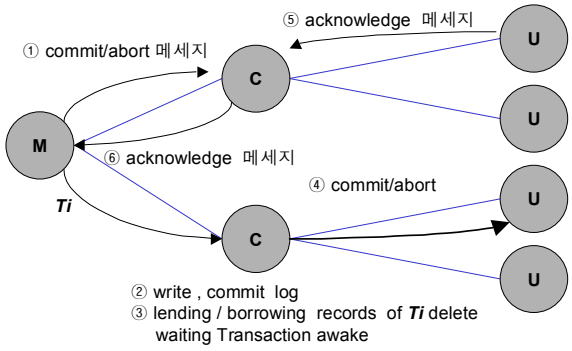


(그림 3) 완료 준비 단계

3.5 결정 단계

완료준비 단계에서 검증을 성공적으로 마친 master는 commit 메시지를 cohort에게 전달한다. 그리고 cohort는 변경할 데이터를 데이터베이스에 반영하고 commit 로그 레코드를 추가한다. 이때 해당되는 대여 및 차용 기록을 삭제하고 대기중인 차용자 트랜잭션을 실행시킨다.

commit 메시지를 전달받은 remote update는 변경된 내용을 데이터베이스에 반영하고 commit 로그 레코드를 추가한 후에 cohort에게 확인 메시지를 전달한다. 모든 remote update에게서 확인 메시지를 전달받은 cohort는 master에게 확인 메시지를 전달한다.



(그림 4) 결정 단계

4. 성능 평가

4.1 시뮬레이션 모델

본 논문에서 제안한 우선 순위 상승 프로토콜은 DO2PL_PA와 MIRROR를 대상으로 성능 평가를 실시하였다. 성능 평가에 사용된 시뮬레이션 모델은 MIRROR[8]에서 적용한 모델을 기반으로 하였다.

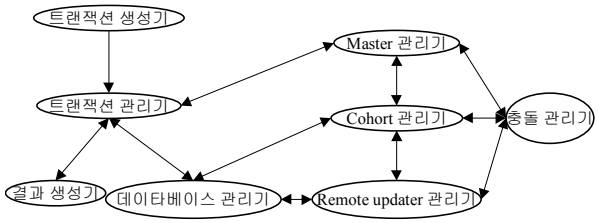
(표 1) 시스템 파라미터

파라미터	내용	설정값
Numsites	전체사이트의 수	9
DBSize	데이터베이스 크기	1000 pages
ReplDegree	데이터의 중복 정도	100 page
Deadline	트랜잭션의 마감시간	Trans/Second
ArrivalRate	트랜잭션의 도착 비율	트랜잭션의 도착시간 + 자원점유시간*SlackFactor
TransSize	트랜잭션당 page 수	16*(0.5 ~ 1.5)

성능 평가에 사용된 파라미터는 (표 1)과 같다. 시뮬레이션 모델은 1000개의 페이지를 갖는 9개의 사이트로 구성하였으며, 각 데이터베이스는 100개 페이지의 중복 데이터를 갖는다. 또한 한 트랜잭션은 9 ~ 24개의 페이지를 갖는다. 트랜잭션의 마감시간은 (식 1)과 같으며, 트랜잭션의 도착 비율은 초당 도착하는 트랜잭션의 개수로 설정하였다.

마감 시간 = 자원점유시간 * Slack_Factor + 도착시간 ..... (식 1)

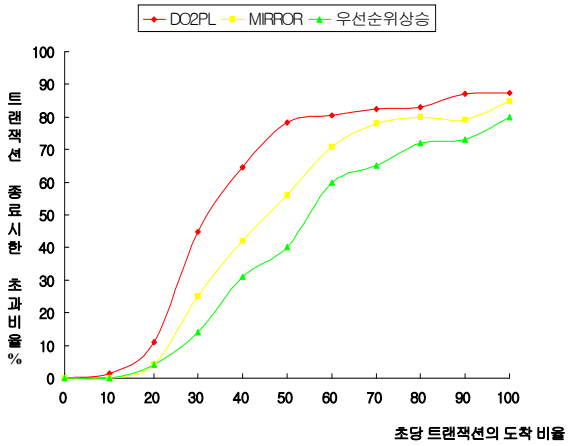
시뮬레이션 모델의 각 구성 요소는 (그림 5)와 같다. 트랜잭션 생성기에서는 시스템 파라미터를 바탕으로 시뮬레이션에 사용될 트랜잭션을 생성한다. 데이터베이스 관리기에서는 데이터베이스를 생성하고 각 데이터베이스를 관리한다. 트랜잭션 관리기는 트랜잭션 생성기에서 생성한 트랜잭션에 대한 master 트랜잭션을 생성하고 실행한다. Master 관리기, Cohort 관리기, Remote updater 관리기에서는 각 트랜잭션에 의해 생성되는 하위 트랜잭션의 생성, 실행, 완료를 관리한다. 트랜잭션간에 충돌 해결 방법으로는 충돌 관리기에서 제공하는 PA 및 PB를 사용한다.



(그림 5) 시뮬레이션 모델의 각 구성요소

4.2 실험 결과

트랜잭션 도착 비율에 따른 트랜잭션의 마감시간 초과 비율을 나타낸 것이 (그림 6)이다. 실험 결과에서와 같이 제안하는 우선 순위 상승 프로토콜이 DO2PL_PA나 MIRROR에 비해서 마감시간 초과 비율이 낮아진 것을 알 수 있다. 이것은 완료 준비 단계의 트랜잭션에 대해서 우선 순위를 상승시킴으로써 트랜잭션의 완료를 최대한 보장하고 트랜잭션의 데이터 점유 시간을 줄일 수 있기 때문이다. 부가적으로, 상승 우선 순위를 갖는 트랜잭션이 점유한 데이터에 대한 차용을 허용함으로써, 트랜잭션의 낭비되는 재시작을 감소시킬 수 있다.



(그림 6) 트랜잭션 도착 비율에 따른 종료시한 초과 비율

5. 결 론

본 논문에서는 분산 실시간 데이터베이스 시스템을 위한 효율적인 동시성 제어 기법인 우선 순위 상승 프로토콜을 제안했다. 새롭게 제안된 우선 순위

상승 프로토콜은 트랜잭션의 상태에 따라 우선 순위를 상승시킴으로써 트랜잭션의 재시작에 의한 낭비를 줄인다. 뿐만 아니라 트랜잭션의 완료를 최대한 보장하며, 잠금 지연 시간을 최소화하는 장점을 갖는다. 또한 다른 트랜잭션이 해당 데이터를 필요로 하는 경우 데이터 차용을 통해 다른 트랜잭션의 지연시간을 줄여줌으로써 전체적인 시스템 성능을 향상시킨다.

트랜잭션의 상태 정보는 MIRROR 프로토콜과 마찬가지로 사이트간의 추가적인 통신을 필요로 하지 않는다. 그리고 트랜잭션의 충돌 해결 방법은 MIRROR와 달리 PA만을 적용함으로써 구현이 쉬운 장점을 갖는다.

향후, 우선 순위 상승 프로토콜을 낙관적 동시성 제어 기법인 OCC-Sacrifice, OCC-Wait, Wait 50 등에 적용하여 성능평가를 수행할 계획이다.

참고 문헌

[1] Baothman, F., Sarje, A.K. and Joshi, R.C. "On optimistic concurrency control for RTDBS," TENCON '98. 1998 IEEE Region 10 International Conference on Global Connectivity in Energy, Computer, Communication and Control Volume: 2 , Page(s): 615 -618 vol.2 1998.

[2] Carey, M., and Linvy, M., "Conflict Detection Tradeoffs for Replicated Data," ACM Transactions on Database Systems, Vol. 16, Page(s): 703-746, 1991.

[3] C. Mohan, B. Lindsay and R. Obermarck, "Transaction Management in the R* Distributed Database Management System" , ACM TODS, 11(4), 1986.

[4] Haritsa, J.. R. and Carey, M.J.; Livny, M. "Dynamic real-time optimistic concurrency control," Real-Time Systems Symposium, Proceedings., 11th , 1990 , Page(s): 94 -103, 1990.

[5] Michael J. Carey. and Miron Livny. "Distributed Concurrency Control Performance: A Study of Algorithm, Distribution, and Replication," Proceedings of the 14th VLDB Conference, Los Angels, California Page(s): 13 -25. 1988.

[6] Son, S., "Advances In Real-Time Systems," Prentice Hall, 1995.

[7] Thomasian, A. "Distributed optimistic concurrency control methods for high-performance transaction processing," Knowledge and Data Engineering, IEEE Transactions on Volume: 10 1 , Jan.-Feb., , 1998.

[8] Xiong, M., Ramamritham, K., Haritsa, J., and Stankovic, J., "MIRROR: A State-Conscious Concurrency Control Protocol in Replicated Real-time Database," Technical Report 98-36, Department of Computer Science, University of Massachusetts at Amherst, 1998 (http://wwwccs.cs.umass.edu/rtdb/publications.html).