□특집□

실시간 데이터베이스 시스템

김 영 국†

◈ 목 차 ◈

1. 서 론

실시간 데이타베이스 시스템(real-time database system 또는 RTDBS)은 트랜잭션이 마감시간(deadline)과 같은 시간적 제약조건을 가지는 데이타베이스 시스템이다. 시스템의 정확성(correctness)은 논리적인 결과뿐만 아니라 그 결과가 얻어진 시간에도 의존한다. RTDBS에서 트랜잭션들은 각각의 마감시간에 이르기 전에 완료될 수 있도록 스케쥴되어야만 한다. 예를 들어, 한 미사일의 궤적(tracking) 데이타에 대한 갱신(update)이나 질의(query)는 주어진 마감시간 안에 반드시 처리되어야만 한다[1,2].

RTDBS는 항공과 무기체계, 공장자동화, 로보틱스, 원자력발전소, 교통제어시스템과 같은 다양한 응용분야에서 점점 중요시되어지고 있다. 최근에 이 분야의 연구자들은 기존의 데이타 모델과 트랜잭션 모델이 시간적 제약을 가지는 응용분야에 적합하기 않기 때문에, 공유 데이타를 수집하고, 갱신하고, 검색하는 데 있어서의 시간적인 제약조건들을 만족시킬 수 있는 데이타베이스 시스템에 대한 기본적인 연구가 필요함을 인식하고, 많은 연구들을 진행하여 왔다[3].

실시간 데이타베이스 시스템에서의 트랜잭션은 크게 다음 두 가지로 구분될 수 있다. *Hard 실시간 트랜잭션*은 시간적 제약이 반드시 준수되어야 하는 경우를 말하며 만약 이러한 제약조건이 지켜지지 못할 경우에는 중대한 손실이 초래된다. 예를 들어, 비행물체를 추적하여 대공미사일을 발사하는 국방시스템은 자연히 이 경우에 속한다. 한편, *soft 실시간 트랜잭션*은 시간적 제약이 존재하기는 하지만, 이를 지키지 못하면서 트랜잭션을 완료하여도 어느 정도의 가치는 있는 경우를 말한다. 일반적으로 soft 실시간 트랜잭션의 마감시간을 지키지 못해도 중대한 손실은 초래하지 않는다. Soft 실시간 트랜잭션은 시간적 제약을 고려하여 스케쥴링을 하지만 마감시간 내 연산의 완료를 보장하지는 않는다. 이러한 트랜잭션의 구분은 각각의 응용분야에 의하여 대체로 결정되지만, 두 가지 종류의 트랜잭션을 공히 필요로 하는 실시간 시스템도 많이 있다.

기존의 데이타베이스 시스템은 다음 두 가지의 부적합성 때문에 실시간 응용 분야에서 일반적으로 사용되지 않고 있다. 하나는 *만족스럽지 못한 성능*이고 다른 하나는 *예측가능성(predicta-*

† 정회원 : 충남대학교 컴퓨터과학과 조교수

*bility)의 결여*이다. 기존의 데이타베이스 시스템에서는 트랜잭션 처리가 보조기억장치에 저장되어 있는 데이타베이스에 대한 액세스를 필요로 한다. 이로 인하여, 트랜잭션의 응답시간은 디스크 액세스 지연시간에 의하여 제약을 받는다. 이러한 데이타베이스는 수 초 정도의 응답시간이면 충분한 인간 사용자를 대상으로 하는 전통적인 응용분야를 위해서는 충분히 빠른 것이지만, 고성능을 요하는 실시간 응용분야를 위해서 충분할 만큼 빠른 응답시간을 제공할 수는 없다. 고성능을 얻을 수 있는 한가지 방법은 느린 장치들(예를 들면, 하드디스크)을 고속의 다른 장치들(예를 들면, 대용량의 RAM)로 대체하는 것이다. 다른 하나의 방법은 응용분야의 특성에 대한 지식을 이용해서 동시성(concurrency)의 정도를 높이는 방법이다. 예를 들어, 해당 응용분야의 트랜잭션과 데이타에 대한 의미(semantic) 정보를 파악함으로써 직렬성(serializability)이 아닌 다른 정확성(correctness) 기준을 사용할 수도 있을 것이다. 직렬성을 보장하는 것은 트랜잭션들의 동시수행을 제약 하기 때문에 시간적인 제약을 가지는 데이타베이스 시스템에서의 동시성제어를 위한 정확성 기준으로서는 적합하지 않을 수 있다. 필요하다면, 시간적 제약조건들을 만족하도록 하기 위해서, 데이타의 일관성을 어느 정도 포기할 수도 있을 것이다.

예측가능성이라는 측면에서, 기존의 데이타베이스 시스템은 응답시간 요구조건을 만족하도록 트랜잭션을 스케쥴하지 않으며, 일반적으로 일관성 보장을 위하여 데이타에 잠금(lock)을 건다. 잠금과 시간에 기반한 스케쥴링은 기본적으로 상충된다. 우선순위가 낮은 트랜잭션이 우선순위가 높은 트랜잭션을 블록할 수 있는데, 이는 우선순위가 높은 트랜잭션이 시간적 요구조건을 만족하지 못하도록 만들 수 있다. 결론적으로, 실시간 데이타베이스 시스템의 요구조건과 설계 목표는 기존의 데이타베이스 시스템과는 많이 다르며, 새로운 기술들이 데이타베이스의 일관성을 유지하기 위해서 필요하다. 그들은 시간에 기반한 스케쥴링과 양립할 수 있어야하며, 시스템의 응답시간 요구조건들과 시간적 일관성(temporal consistency) 요구조건들을 두루 만족해야한다. 이때 관건은 어떻게 기존의 데이타베이스 시스템을 수정해서 그 성능과 예측가능성이 실시간 응용분야에 적합하도록 만드느냐 하는 것이다.

RTDBS에서 트랜잭션의 적시성(timeliness)은 대개 트랜잭션의 우선순위를 결정할 때 그 트랜잭션의 중요도(criticality)와 함께 사용된다. 따라서, 실시간 트랜잭션 스케쥴링에서는 우선순위들의 적절한 관리와 충돌해결방안(conflict resolution)이 RTDBS의 예측가능성과 빠른 응답성을 위해 필수적이다.

데이타베이스 시스템에서의 동시성제어 이론과 실시간 태스크 스케쥴링 분야에 많은 진보가 있어온 반면에, 동시성제어 프로토콜과 실시간 스케쥴링 알고리즘 사이의 상호작용에 대해서는 거의 연구가 이루어지지 않아 왔다[4]. 데이타베이스 동시성제어에서는 마감시간을 만족시키는 것이 주의 깊게 다루어지지 않는다. 주요목적은 고도의 동시성을 제공함으로써 데이타의 일관성을 위반함이 없이 평균 응답시간을 빠르게 하는데 있다. 실시간 스케쥴링에서는 반대로 태스크들은 독립적이거나, 또는 공유데이타에 대한 액세스를 동기화하는데 보내는 시간은 실행시간에 비해서 무시할만하다고 가정하는 것이 보통이다. 여기에서의 목적은 시간적인 제약조건을 만족하는 범위 안에서 CPU와 같은 자원의 활용도(utilization)를 최대화하는 것이다. 데이타의 일관성은 실시간 스케쥴링에서는 고려 대상이 아니며, 따라서 공유데이타의 일관성을 보장하는 문제는 무시된다. 또한, 기존의 실시간 시스템들은 프로그램들에서 필요로

하는 자원들과 데이타 요구조건에 대한 사전지식을 갖는다고 가정한다.

RTDBS에서 해결되어져야 하는 어려운 문제들 중의 하나는 세 가지의 제약조건, 즉 데이타의 일관성, 트랜잭션의 정확성, 트랜잭션 마감시간을 만족시키면서 동시성과 자원에 대한 활용도를 최대화하는 실시간 스케쥴링과 동시성 프로토콜에 대한 통합된 이론을 만드는 것이다. 최근에 여러 개의 프로젝트들이 RTDBS에서 시간적 제약조건을 효율적이고 정확하게 관리할 수 있도록 데이타베이스 기술과 실시간 요구조건들을 통합하고 있다[5,6]. 이러한 통합을 하기 위해서는 여러 가지 어려움이 있는데, 예를 들어 데이타베이스 액세스 동작은 디스크 입출력, 로깅, 버퍼링 등이 요구되는지 아닌지에 따라서 수행시간에 많은 차이를 보인다. 또한, 동시성제어는 트랜잭션의 철회(aborts) 또는 무한정의 연기(delay)를 유발할 수 있다.

실시간 데이타베이스 시스템의 설계와 구현은 많은 새롭고 도전적인 문제들을 내포하고 있다. 즉, 실시간 트랜잭션과 데이타에 대한 적당한 모델은 무엇인가? 실시간 요구조건들을 명시하기 위해 사용될 수 있는 언어구조들은 무엇인가? 시스템의 예측가능성에 대한 측정기준은 무엇인가? 어떻게 트랜잭션들이 스케쥴되는가? 동시성제어에 있어서 실시간 제약조건의 영향은 무엇인가? 본 고에서는 이러한 문제들 중의 몇 가지에 대한 주요연구현황을 소개하고, 실제로 RTDBS를 설계하는데 있어서의 요구사항들에 대하여 기술하고자한다.

2. 트랜잭션 스케쥴링과 동시성제어

실시간 데이타베이스 시스템에서 스케쥴링의 목적을 두 가지 측면에서 보면, 하나는 시간적 제약을 만족하는 것이고, 다른 하나는 데이타의 일관성을 유지하는 것이다. 시간적 제약을 만족시키기 위해서 실시간 태스크 스케쥴링 방법을 실시간 트랜잭션 스케쥴링에 확장 적용하는 것이 가능하며, 데이타 일관성 유지를 위한 동작 스케쥴링을 위해서는 동시성제어 프로토콜이 필요하다. 실시간 데이타베이스 시스템에서 트랜잭션 스케쥴링과 동시성제어 프로토콜과의 통합은 쉽지 않다. 일반적으로, 실시간 데이타베이스 시스템에 사용되는 동시성 제어 프로토콜로는 기존에 있는 동시성 제어 프로토콜을 효율적으로 바꾸고, 트랜잭션 스케쥴링으로는 좀 더 임박한 트랜잭션을 먼저 수행할 수 있도록 시간적 제약을 가지는 트랜잭션을 스케쥴링하는 방법을 사용한다 [7].

동시성 제어는 데이타베이스의 일관성을 잃지 않도록 하는 방법으로서 동시에 존재하는 트랜잭션들의 제어를 말한다. 동시성 제어에서 가장 많이 사용되는 정확성 기준은 직렬성이다. 연속된 데이타베이스 오퍼레이션들의 결과가 트랜잭션들을 순차적으로 수행했을 때의 결과와 같으면 직렬적이라고 한다. 데이타의 유효시간이 일시적인 실시간 응용분야에서는 실시간 데이타베이스 시스템에서 성능을 향상하기 위해서 직렬성을 희생하도록 하는 방식을 따르기도 하고, 응용분야에 따라 데이타베이스 일관성을 유지하는데 직렬성을 따르기도 한다. 이때 동시성 제어에 대한 일반적인 접근방식은 잠금(locking)에 기반한 비관적 동시성 제어(Pessimistic Concurrency Control 또는 PCC) 기법들과 낙관적 동시성 제어 (Optimistic Concurrency Control 또는 OCC) 기법들이 있다[7].

먼저 PCC에 기반한 실시간 동시성 제어 기법들을 살펴보면, 기존의 데이타베이스 시스템에서 가장 일반적인 잠금기반 프로토콜인 이단계 잠금(2PL)기법은 우선순위 역전(priority inversion) 및 교착상태(deadlock)가 발생하는 문제점을 가지고 있다. 우선순위 역전은 우선순위가 높은 트랜잭션

이 우선순위가 낮은 트랜잭션에 의해 블럭되는 현상이다[8]. Wait Promote 방식은 이러한 문제를 해결하기 위한 하나의 방법으로 잠금을 소유하고 있는 트랜잭션의 낮은 우선순위를 잠금을 요청하는 트랜잭션의 높은 우선순위로 바꾸어주는 방법이다[9]. 이 방식은 우선순위 역전현상을 해결하는 반면, 교착상태가 발생하고, 연쇄 블로킹으로 인한 시스템 성능 저하를 가져올 수 있다. 다음으로 High Priority 방식은 우선순위가 낮은 트랜잭션을 취소하고 우선순위가 높은 트랜잭션을 실행하도록 허용하는 방식이다[8]. 이 방식은 우선순위 역전(priority inversion)문제를 해결하지만, 우선순위를 최소여유시간우선(Least Slack First) 방식으로 하는 결정하는 경우, 트랜잭션의 반복적인 철회를 일으키게 되는데, 이러한 반복적인 철회문제를 해결하기 위하여 수정된 High Priority 방법이 제안되었다[8]. 이러한 High Priority 방식은 간단한 반면, 교착상태를 발생하거나 시스템의 자원을 낭비하게 된다. Conditional Restart 방식은 이러한 자원의 낭비문제를 해결하기 위해서 데이타 충돌에 포함된 하위의 우선순위를 갖는 트랜잭션의 남은 수행 시간과 상위의 우선 순위를 갖는 트랜잭션의 여유시간을 고려하는 방식이다[8]. 이 방식은 자원의 낭비를 줄일 수는 있지만, 트랜잭션의 수행시간에 대한 정확한 예측이 필요하다는 단점이 있다.

다음으로 낙관적 동시성제어 방법을 살펴보면, 트랜잭션의 실행은 일반적으로 판독 단계(read phase), 검증 단계(validation phase), 기록 단계(write phase)의 3가지 단계로 나뉘어진다. OCC에 기반한 실시간 동시성 제어 기법들[11,12] 중에서 OPT-BC(Abort) 기법은 임의의 트랜잭션이 검증 단계에 도달했을 때 충돌이 일어난 트랜잭션들과의 우선 순위가 비교된다. 즉, 충돌된 트랜잭션들이 전부 검증 단계의 트랜잭션보다 우선순위가 높다면 검증 단계의 트랜잭션이 즉시 취소되면 그렇지 않으면 충돌된 트랜잭션이 모두 취소되는 두 가지 경우가 발생한다. OPT-Sacrifice 기법은 검증 단계에 있는 트랜잭션이 충돌이 일어난 트랜잭션들 중 우선순위가 더 높은 트랜잭션이 최소한 하나라도 존재하면 자신이 취소된다. 그렇지 않은 경우에는 검증 단계에 있는 트랜잭션이 기록 단계로 전환되며 현재 충돌중인 트랜잭션들을 모두 취소하게 된다. 결과적으로 OPT-Sacrifice 기법에서는 우선순위가 높은 트랜잭션이 성공적으로 종료되지 못하는 경우에는 낭비적인 취소가 된다. OPT-Wait 기법은 우선순위가 더 높은 트랜잭션에게 종료시한을 보장할 수 있는 기회를 먼저 부여하는 것이다. 이 방법은 검증 단계에 있는 트랜잭션을 즉시 취소하거나 종료시키는 대신 일단 지연시키는 것이다. Wait-50 기법은 OPT-Wait 기법의 문제점을 개선하고자 제안된 것으로, 이 기법은 OPT-Wait 기법에서 사용된 지연 기법의 잇점을 최대한 활용하였으며 실시간 시스템의 성능 저하 현상을 감소시키고자 하였다[12].

[11]에서 수행한 시뮬레이션 실험에서는 High Priority에 기반한 2PL 기법과 OPT-BC 기법을 비교하였다. 결과는 마감시간을 지난 트랜잭션을 수행하지 않는 방식에서 낙관적 동시성제어는 2PL 방식보다 좋은 성능을 보여준다. Huang과 Stankovic[13]에 의한 또 다른 연구에서도 낙관적 동시성 제어(OCCL_SVW)와 2PL을 비교하고 있다. 결과는 OCCL_SVW와 2PL의 성능의 차이는 데이타 경쟁(contention)의 정도에 영향을 많이 받는 한편, 입출력 자원의 경쟁에는 무관하다는 것이다. 특히 낙관적 동시성 제어는 데이타 경쟁이 낮을 때 2PL보다 성능이 우수한 반면 그렇지 않은 경우 2PL이 우수한 성능을 보여준다.

한편, PCC나 OCC 기반이 아닌 실시간 동시성 제어 기법들도 다수 소개되고 있다. [14]에서 Kim

과 Srivastava는 RTDBS에서 다중버전 2PL 동시성 제어에 기반한 다양한 프로토콜을 사용함으로써 잠재적인 성능의 향상을 시도하고 있다. [15,16]에서 Son 등은 OCC와 타임스탬프 순서방식을 통합한 혼성 프로토콜을 제안하고 있다. 이 프로토콜에서는 트랜잭션들의 정확한 직렬화 순서에 대한 결정이 가능한 한 뒤로 미루어져서 긴급한 트랜잭션들의 완료(commit)가 허용되도록 한다. 이는 타임스탬프 간격의 동적인 할당과 조정을 통하여 이루어진다. [17]에서 Bestavros와 Braoudakis는 Speculative Concurrency Control (SCC)을 제안하고 있는데, 이 방식에서는 특정 트랜잭션이 더 나은 결과를 얻기 위해서 더 많은 자원을 사용하도록 허용되며, 따라서 그 트랜잭션이 적시에 완료될 확률을 높이는 방식이다.

3. 시간적 일관성 (Temporal Consistency)

종종 실시간 데이타베이스 내의 상당수의 데이타 객체들은 극히 짧은 생명주기를 가진다. 따라서, RTDBS에서는 트랜잭션의 마감시간 이외에 다른 종류의 시간적 제약이 데이타와 연관되어 있을 수 있다. 예를 들면, 각각의 센서입력은 그것이 측정된 시간에 의해서 인덱스될 수 있다. 일단 데이타베이스에 저장된 이러한 데이타는 일정 시간 내에 갱신되지 않는다면 낡은 값이 되고 만다. 이러한 "age"의 개념을 정량화하기 위해서 실시간 데이타는 *유효간격(valid interval)*을 가질 수 있다. 유효간격 밖의 데이타는 해당 객체의 현재 상태를 나타내지 않는다. 한 트랜잭션이 유효간격 밖의 데이타를 액세스하려고 할 때 어떻게 대처할 것인가는 해당 응용에 있어서 데이타의 의미적 특성과 특정 시스템 요구조건에 달려있다.

실시간 트랜잭션은 그 트랜잭션에 의해서 액세스되는 데이타 값의 유효성을 명시하는 시간적 일관성 요구조건을 가질지도 모른다. 트랜잭션의 시간적 일관성 요구조건이 만족될 수 있는 한 그 시스템은 (최신은 아니더라도) 사용 가능한 정보를 이용해서 답을 제공할 수 있어야만 한다. 답은 유효간격이 시간에 따라 변함에 따라서 변할 수도 있다. 분산 데이타베이스 환경에서는, 센서 측정값들이 동시에 데이타베이스에 반영되지 않을 수도 있으며, 처리와 통신 지연 때문에 일관성 있게 반영되지 않을 수도 있다. 따라서, RTDBS를 위한 시간적 데이타 모델은 부분적이고 낡은 정보도 수용할 수 있어야 한다. RTDBS에서의 시간적 데이타모델을 기존의 데이타베이스 시스템의 데이타 모델과 구분짓는 한가지 측면은 RTDBS에서의 데이타 값들은 항상 정확성을 유지하지는 않을 수도 있다는 것이다. 따라서, 시스템은 데이타 값을 해석하는데 있어서 신중을 기해야만 한다. [18,19,20,21]

4. 주기억 상주 데이타베이스 시스템

실시간 트랜잭션의 시간 제약 조건을 만족시키기 위한 방법으로 기존 디스크 상주 데이타베이스 시스템에 실시간 처리 특성을 추가하는 방법과 주기억장치에 전체 데이타베이스를 상주시키는 방법이 있지만, 전자의 방법은 디스크 입출력으로 인한 트랜잭션 처리 지연을 피할 수 없기 때문에 실시간 트랜잭션 처리에 적합하지 않다고 볼 수 있다. 따라서, 주기억상주 데이타베이스 시스템(Memory Resident Database System 또는 MRDBS)을 실시간 시스템에 적용하는 실시간 주기억장치 데이타베이스 시스템(Real-Time Main Memory Database System)이 연구의 초점이 되고 있다[1].

RTDBS를 설계하는데 발생하는 주요 난제 중의 하나는 디스크를 액세스하는데 걸리는 지연시간이 느리고 예측하기 어렵다는 것이다. 메모리의 가격이 계속해서 낮아짐에 따라 떠오른 대안이

모든 데이타를 메모리에 넣고, 그럼으로써 I/O 액세스를 제거하는 것이다. 디스크에 비하여, 주메모리 액세스 시간은 매우 빠르고(1,000~10,000배), 디스크 탐색 시간이 없으므로 좀더 예측 가능하다. 이러한 특징은 RTDBS에서 매우 바람직하며, 만일 트랜잭션이 극도로 타이트한 시간 제약 조건을 가진다면 꼭 필요할 수도 있다.

그러나, 모든 데이타를 메모리에 올린다고 해서 단점이 없는 것은 아니다. 무엇보다도, MRDBS는 디스크 기반의 시스템보다 비싸다. 반도체의 가격이 계속해서 떨어지고 집적도는 계속 증가한다 하더라도, 메모리에 모든 데이타를 상주시키기에는 여전히 한계가 있다. 커다란 데이타베이스의 경우, 주메모리에 데이타를 저장하는 것은 선택적으로 해야 한다. 실시간 환경에서, 만일 트랜잭션의 데이타 요구가 상대적으로 안정적이고 알려져 있다면, 높은 우선순위를 가진 트랜잭션에 의해 참조된 데이타 아이템은 메모리에 상주하는 데 있어서 낮은 우선순위의 트랜잭션에 의해 요구된 데이타 아이템보다 우선 순위를 가진다.

주메모리를 사용하는 데 생기는 또 하나의 문제는 메모리가 휘발성이라는 것이다. 주메모리에 저장된 데이타는 보통 CPU 고장이나 정전 시에 보관되지 않는다. 그러므로, MRDBS는 안정적인 백업 저장소로서 여전히 디스크를 필요로 한다. 전체 데이타베이스의 디스크 백업본을 메모리로 로딩한 후, 데이타베이스를 최근의 데이타로 갱신하기 위하여 트랜잭션 로그를 적용하는 기존의 회복 기법은 실시간 응용에 쓰이기에는 너무 느리다. 빠른 재시작 및 회복 시에도 부분적이나마 동작하는 데이타베이스를 제공하는 기법들이 요구된다.

MRDBS에서의 세 번째 논점은 기존의 디스크 기반의 시스템과는 설계 목적이 다르다는 것이다. 기존의 데이타베이스 시스템에서의 데이타 구조와 질의처리 알고리즘은 디스크 액세스 횟수를 최소화하고, 데이타 집적도를 향상시키는데 적합하도록 설계되었다. 이러한 목적은 MRDBS에서는 더 이상 필요치 않다. 데이타가 메모리에 상주하고 있을 경우, 질의 최적화와 데이타구조는 CPU 처리시간을 최소화하고, 메모리 공간을 최소한으로 사용하도록 설계되어야 한다. 따라서, 기존의 액세스 방법과 데이타베이스 구조는 수정되어야 한다. 예를 들어, B-tree 방식은 MRDBS의 인덱스 탐색을 위한 해싱 기법보다 덜 효과적이다. B-tree의 경우는 모든 키와 포인터를 저장해야 할 필요가 있으므로 추가적인 스페이스가 필요하기 때문이다. 디스크 I/O 횟수를 최소화하기 위해 설계된 sort-merge join 알고리즘의 경우는 성능 면에서 메모리가 충분할 때의 hash-merge 알고리즘보다 낮은 것으로 알려져 있다[23].

5. 향후 RTDBS 연구 및 개발 방향

지금까지 RTDBS의 개념과 필요성, 연구분야 및 기존의 연구결과들에 대하여 간략하게 살펴보았다. 향후의 RTDBS의 연구 및 개발은 다양한 실시간 응용분야와 운영 환경에 적응할 수 있도록 다음과 같은 방향으로 이루어져야 할 것으로 생각된다.

● 융통성 있는 시스템 구조 (Flexible System Architecture)

각각의 실시간 응용 분야들은 트랜잭션과 데이타에 대하여 서로 상이한 특성과 성능 요구조건을 가지고 있기 때문에, 고정된 시스템 구조와 운영환경을 가지는 하나의 RTDBS가 모든 종류의 실시간 응용에 적용될 수 없다. 따라서, 기존의 대부분의 RTDBS는 특정 응용분야를 염두에 두고 특정한 운영환경을 위해 개발되었기 때문에, 다른 응용분야나 다른 운영환경에서 사용될 수 없었다. 앞으로의 RTDBS는 응용분야의 특성에 따라서 쉽

게 커스터마이즈되고, 목표 운영환경에 쉽게 이식될 수 있도록 융통성있는 시스템 구조를 가져야 한다. 즉, 시스템의 각 기능들이 서로 독립적인 모듈들로 설계 및 구현되어 특정 응용분야에서 필요한 기능 모듈들만을 취사 선택하여 시스템을 구성할 수 있어야 한다.

● 다양하고 선택 가능한 기능 (Functionality) 제공

응용분야의 성격에 따라 선택적으로 적용될 수 있도록 다음과 같은 다양한 트랜잭션 스케쥴링 알고리즘이 제공되어져야 한다.

1) Rate-Monotonic 또는 Deadline-Monotonic: 모든 트랜잭션들의 도착주기와 수행시간을 예측할 수 있는 경우
2) Earliest-Deadline-First: 트랜잭션들이 임의로 도착하며, 수행시간을 예측할 수 없는 경우 (실시간운영체제의 지원 필요)
3) User-Defined Fixed-Priority: 트랜잭션들이 임의로 도착하며, 수행시간을 예측할 수 없는 경우 (우선순위는 마감시간, 중요도 등을 고려하여 결정)

또한, 응용분야의 성격에 따라서 선택적으로 사용할 수 있도록 다음과 같이 다양한 동시성제어 기법을 제공해야 한다.

1) 선입선출(First Come First Serve): 경우에 따라서 트랜잭션들의 순차적인 수행이 가장 좋은 성능을 낼 수 있다.
2) 잠금기반 실시간 동시성제어 (Lock-based Real-Time Concurrency Control): 대부분의 시스템에서 사용하는 기법. 일반적으로 데이타 충돌이 빈번할 경우 좋은 성능을 보인다.
3) 낙관적 실시간 동시성제어 (Optimistic Real-Time Concurrency Control): 성능과 구현상의 여러가지 난점들로 인하여 범용 DBMS에서는 거의 사용되어지지 않는 기법이나 주기억장치 상주형의 RTDBS 환경하에서는 구현과 성능 상에 여러 가지 장점을 갖는 것으로 보고되어지고 있다.

기존의 디스크 기반 데이타베이스에서는 주데이타베이스가 디스크에 존재하고 필요에 따라 해당데이타를 주기억장치로 옮긴다. 반면에 주기억장치 상주형 데이타베이스는 주데이타베이스가 메모리에 상주하며 필요에 따라 데이타를 디스크로 옮긴다. 이러한 구조상의 차이로 인해 이미 디스크 상주형 데이타베이스를 위해 개발된 회복기법들은 그대로 적용될 수 없다. 또한 비휘발성 메모리의 사용으로 인한 새로운 회복방법의 연구개발이 필요하다. 한편, 외부 환경을 모니터링하는 응용 (즉, 센서로부터 주기적으로 데이타를 받아 저장하는 경우) 또는 부착형 데이타베이스로 활용되는 경우에는 회복이 필요없기 때문에 회복관리모듈을 제외할 수 있어야 한다.

● 실시간 성능 (Real-Time Performance)

- 실시간성: 실시간 DBMS의 운영 환경에서 여러가지 트랜잭션 유형들에 대하여 종료시한 내에 처리를 보장하는 비율이 X % 이상이어야 한다. (여기서 X 값은 응용에 따라 다르게 정의된다.)
- 처리율(throughput) 및 응답시간(response time): 일반적으로 초당 100 ~ 1,000개 정도의 TPC-B 타입의 트랜잭션을 처리해야하며, 95%의 TPC-B 타입 트랜잭션에 대하여 수십 msec 이내의 응답시간을 만족해야 한다.

● 다양한 운영환경 (Operating Environments) 지원

트랜잭션들이 엄격한 시간제약을 가지는 실시간 응용을 위해서는 실시간 스케쥴링 등을 지원하는 실시간 커널 위에 RTDBS를 개발할 필요가 있다. 그러나, 관대한 시간 제약을 갖거나, 빠른 응답시간만을 요구하는 대부분의 실시간 응용을

위해서는 범용 운영체제 환경 하에서 동작하는 이식성 좋은 RTDBS가 필요하다. 이를 위해서는 POSIX.4와 같은 표준에 의거해 코딩이 이루어져야 하며, EDF와 같은 마감시간 스케쥴링의 효율적인 에뮬레이션이 필요하다.

6. 결 론

실시간 데이타베이스 시스템은 정보통신망 서비스, 공정 제어 등 넓은 영역에 걸쳐서 그 중요성이 부각되고 있다. 범용 데이타베이스 시스템은 예측성의 결여와 성능 저하라는 두 가지 요인으로 인하여 실시간 데이타베이스 응용에 적합하지 않다. 실시간 데이타베이스 시스템에 대한 요구사항과 설계는 전통적인 데이타베이스 시스템과 크게 다르다. 실시간 데이타베이스 시스템의 요구사항은 실시간 데이타베이스 응용에 따라서 크게 다를 수 있으며, 전체 시스템의 구성과 밀접하게 관련되어 있으므로 별도로 데이타베이스 관리 시스템에 대한 요구사항만을 도출하는데 상당히 많은 어려움이 있다.

본 고에서는 실시간 데이타베이스 시스템의 개념과 필요성, 연구 분야와 현황에 대하여 살펴보고, 실시간 데이타베이스 시스템이 사용될 수 있는 응용분야의 특성을 파악한 다음, 실용적인 실시간 데이타베이스 시스템의 요구사항과 개발 방향에 대하여 고찰하여 보았다.

참고문헌

[1] Ben Kao and Hector Garcia-Molina, "An Overview of Real-Time Database Systems", In Sang H. Son, editor, Advances in Real-Time Systems, chapter 19. Prentice Hall, 1995.

[2] Krithi Ramamritham, "Real-Time Databases", International Journal of Distributed and Parallel Databases, 1(2), 1993.

[3] G. Ozsoyoglu and R. Snodgrass, "Temporal and Real-Time Databases: A Survey", IEEE Transactions on Knowledge and Data Engineering, 7(4):pp.513--532, August 1995.

[4] J. A. Stankovic, "Real-Time Computing Systems : The Next Generation", Tutorial Hard Real-Time Systems, pp.14-37, 1988.

[5] Matt Lehr, Young-Kuk Kim, and Sang H. Son, "Managing Contention and Timing Constraints in a Real-Time Database System", In Proceedings of the 16th Real-Time Systems Symposium, pp.332-341, Pisa, Italy, December 1995.

[6] A. Buchmann et al. "Time-Critical Database Scheduling: A Framework for Integrating Real-Time Scheduling and Concurrency Control", In Proceedings of the 5th International Conference on Data Engineering. IEEE, February 1989.

[7] P. Yu, K. Wu, K. Lin and S. H. Son, "On Real-Time Databases: Concurrency Control and Scheduling", Proceedings of IEEE, Special Issue on Real-Time Systems, 82(1):pp.140-157, January 1994.

[8] R. Abbott and H. Garcia-Molina, "Scheduling Real-Time Transactions: A Performance Evaluation", ACM Transactions on Database Systems, 17(3):pp.513--560, September 1992.

[9] L. Sha, R. Rajkumar, J. P. Lehoczky, "Priority Inheritance Protocols: An Approach to Real-Time Synchronization", IEEE Transaction on Computers, Vol. 39, number 9, pp.1175-1185, 1990.

[10] J. Huang, J. Stankovic, "On Using Priority Inheritance In Real-Time Databases", IEEE

Real-Time Systems Symposium, pp.210-221, 1991.

[11] J. Haritsa, M. Carey, and M. Livny, "On Being Optimistic About Real-Time Constraints", In Proceedings of the ACM Symposium on Principles of Database Systems, April 1990.

[12] J. Haritsa, M. Carey, and M. Livny, "Dynamic Real-Time Optimistic Concurrency Control", In Proceedings of the 11th Real-Time Systems Symposium, Orlando, FL, pp.94-103, December 1990.

[13] J. Huang, A. Stankovic, "Experimental Evaluation of Real-Time Concurrency Control Schemes", Proceedings of the 17th VLDB Conference, pp. 35-46, 1991.

[14] Woosaeng Kim and Jaideep Srivastava, "Enhancing real-time DBMS performance with multiversion data and priority based disk scheduling", In Proceedings of the 12th Real-Time Systems Symposium, December 1991.

[15] Y. Lin and S. H. Son, "Concurrency Control in Real-Time Databases by Dynamic Adjustment of Serialization Order", In Proceedings of the 11th Real-Time Systems Symposium, pp.94--103, Orlando, FL, December 1990.

[16] S. H. Son, Juhnyoung Lee, and Yi Lin, "Hybrid Protocols using dynamic adjustment of serialization order for real-time concurrency control. The Journal of Real-Time Systems, 4:269-276, 1992.

[17] Azer Bestavros and Spyridon Braoudakis, "Value cognizant speculative concurrency control for real time databases", Information Systems Journal:Special Issue on Real Time Database Systems, 21(1):75-101, March 1996.

[18] K. J. Lin, "Consistency Issues in Real-Time Database Systems", In Proc. 22nd Hawaii International Conference on System Sciences. pp.654-661, Jan 1989.

[19] Tei-Wei Kuo and Aloysius K. Mok, "SSP: A Semantics-Based Protocol for Real-Time Data Access", In Proceedings of the 14th Real-Time Systems Symposium, pp.76-86, Raleigh-Durham, NC, December 1993.

[20] X. Song and J.W.S. Liu, "Maintaining Temporal Consistency: Pessimistic vs. Optimistic Concurrency Control", IEEE Transactions on Knowledge and Data Engineering, 7(5):pp.786--796, October 1995.

[21] M. Xiong, et al. "Scheduling Transactions with Temporal Constraints: Exploiting Data Semantics", In Proceedings of the 17th Real-Time Systems Symposium, pp. 240-251, Washington, DC, December 1996.

[22] Hector Garcia-Molina, "Memory Resident Database Systems: An Overview", *IEEE Trans. on Knowledge and Data Engineering*, Vol.4, No.6, pp.509-516, 1992.

[23] T. J. Lehman and M. J. Carey, "A Study of Index Structure for Main Memory Database Management Systems", In *Proc.Intl.Conf. on Very Large DataBases*, pp.294-303, May 1986.

김 영 국

1985년 서울대학교 자연과학대학 계산통계학과 (학사)
1987년 서울대학교 자연과학대학 계산통계학과 (석사)
1995년 University of Virginia, 컴퓨터과학과 (박사)
1995년-1996년 핀란드 VTT, 노르웨이 SINTEF 연구소 객원연구원
1996년-현재 충남대학교 컴퓨터과학과 조교수
관심분야 : 실시간 시스템, 데이타베이스 시스템, 이동 및 분산시스템