방송환경에서 질의 거래를 위해 직렬화 그래프에 기반을 둔 동시성 제어 기법

(Concurrency Control based on Serialization Graph for Query Transactions in Broadcast Environment : CCSG/QT)

이 욱 현 † 황 부 현 ††

(Ukhyun Lee) (Buhyun Hwang)

요 약 방송환경은 서버(server)와 클라이언트(client)간 대역폭이 서버에서 클라이언트쪽으로는 크고 클라이언트에서 서버쪽으로의 대역폭은 상대적으로 많이 작은 비대칭적(asymmetric) 특수한 환경이다. 또한 대부분의 방송 환경 응용 시스템들은 클라이언트측에서 발생한 주로 주식 데이타, 교통 정보와 새로운 뉴스와 같은 여러 가지 다양한 정보를 검색하는 읽기전용 즉 질의 거래들을 허락한다. 그러나, 기존의 여러 가지 동시성 제어 기법들은 이러한 특수성을 고려하지 않음으로써 방송 환경에 적용될 때 거래들의 불필요한 철회를 일으킨다. 이 논문에서는 방송환경에서 질의 거래를 위한 효율적인 동시성 제어 기법을 제안한다. 이 기법은 서버에 의해 관리 유지되고 클라이언트에 의해 읽혀지는 데이타의 상호 일치성과 데이타의 현재성을 만족시키기에 가장 적절한 정확성 검증 기준인 약한 일치성(weak consistency)을 채택하였다. 또한, 그것을 효율적으로 실행할 수 있도록 직렬화 그래프(serializability graph)를 이용하였다. 그 결과, 전역적 직렬화를 적용할 때 발생되는 질의 거래의 불필요한 철회 및 재시작의 횟수를 줄임으로써 성능향상을 도모하였다.

키워드 : 방송환경, 이동 클라이언트/서버, 비대칭적 대역폭, 질의 거래, 동시성 제어, 약한 일치성, 직렬화 그래프

Abstract The broadcast environment has asymmetric communication aspect that is typically much greater communication bandwidth available from server to clients than in the opposite direction. In addition, most of mobile computing systems allow mostly read-only transactions from mobile clients for retrieving different types of information such as stock data, traffic information and news updates. Since previous concurrency control protocols, however, do not consider such a particular characteristics, the performance degradation occurs when previous schemes are applied to the broadcast environment. In this paper, we propose the efficient concurrency control for query transaction in broadcast environment. The following requirements are satisfied by adapting weak consistency that is the appropriate correctness criterion of read only transactions: (1) the mutual consistency of data maintained by the server and read by clients (2) the currency of data read by clients. We also use the serialization graph scheme to check the weak consistency efficiently. As a result, we improved a performance by reducing unnecessary aborts and restarts of read-only transactions caused when global serializability was adopted.

Key words : broadcast environment, mobile client/server, asymmetric bandwidth, query transaction, concurrency control, weak consistency, serialization graph

1. 서 론

많은 데이타베이스 응용분야에서 특히, 동시에 수행하는 엄청나게 많은 수의 클라이언트들을 가진 응용에서 데이타 전달을 위해 기술이 요구되어진다. 예를 들어, 경매와 같은 전자상거래는 단지 소수에게 낙찰될지라도

† 정 회 원 : 전남대학교 전산학과
uhlee@inha.ac.kr
†† 종신회원 : 전남대학교 전자계산학과 교수
bhhwang@chonnam.chonnam.ac.kr
논문접수 : 2001년 10월 29일
심사완료 : 2002년 11월 25일

수 만 명의 사용자들이 참여한다. 낙찰이 이루어질 때의 갱신 데이타는 신속하고 일관성 있게 전파되어져야 한다. 다행히 데이타베이스의 상대적으로 작은 부분 즉, 경매의 현재 상황과 같은 데이타만이 방송을 요구한다. 그러나, 클라이언트가 서버와 통신하기 위해 필요한 통신 대역폭이 상당히 제한되어 있다. 그러므로, 갱신에 대해 통신하도록 허락되어지는 시간에 클라이언트가 작은 대역폭의 무선망을 사용하여 경매의 현재 상태를 보내기 위해 방송 매체를 사용하는 것이 일반적이다. 즉, 방송환경은 서버와 클라이언트간 대역폭이 서버에서 클라이언트쪽으로는 크고 클라이언트에서 서버쪽으로의 대역폭은 상대적으로 많이 작은 비대칭적(asymmetric)인 특수한 환경이다[1].

무선환경 및 이동 컴퓨팅 환경을 위한 여러 가지 동시성 제어 기법들이 국내외에서 제안되었다. 이전 연구 노력의 대부분이 주로 이동 컴퓨팅 환경은 통신 비용이 비싸고 통신자체가 안전하지 못하다는 특성을 갖는다는 것에만 기반을 두어 방송환경의 특수성을 제대로 고려하지 않았다. 클라이언트가 자체에서 발생한 거래를 제어 완료하지 못하고 서버에게 상당히 많이 의존하는 형태라면 클라이언트측에서 서버측으로 정보 요구가 자주 생기게 된다. 이것은 각 클라이언트에게 주어진 짧은 시간 안에 상대적으로 작은 대역폭을 가지고 윗방향(uplink) 통신을 해야하는 방송환경에서는 거래 실행이 지연되어 전체 성능면에서 단위 시간당 처리율이 떨어진다. 그러므로 비대칭적 방송환경 특수성을 감안하여 클라이언트가 서버로의 윗방향 정보 요구는 가능하면 줄이는 거래 동시성 제어 기법이 필요하다.

또한 기존의 제안된 기법들은 대부분 질의 거래에 대해서는 특별히 고려하지 않았다. 대부분의 이동 컴퓨팅 시스템에서의 거래들은 주로 클라이언트측에서 발생되는 읽기전용(read_only) 거래들이다. 이러한 거래들은 주식 정보, 교통 정보와 새로운 뉴스 등 여러 가지 다양한 종류의 정보를 요구 검색하는 것들이다. 읽기전용 거래는 읽기 연산으로만 이루어진 것으로 이후 질의(query) 거래라 칭한다. 실제적으로 날씨 예보와 교통 정보 시스템과 같은 대부분의 많은 방송 시스템에서 갱신 거래가 질의 거래보다 발생하는 확률이 훨씬 낮다. 이러한 실제 상황을 고려하더라도 질의 거래를 갱신 거래 방해 없이 즉, 질의 거래와 갱신 거래의 충돌연산으로 인한 철회 없이 질의 거래를 최대한 많이 실행 완료시키는 동시성 제어 기법이 연구되어야 한다.

마지막으로 기존의 제안된 기법들은 정확성 검증 기준으로써 직렬화(Serialization)을 사용하는데 그것은 질의 거래가 많은 방송환경에서 매우 비용이 많이 들고 제한적이며 불필요하다[1]. 직렬화[2]은 데이타베이스시스템에서 거래를 위한 가장 공통적인 정확성 검증 기준이다. 그러나, 이것은 본질적으로 포괄적(global)인 성질을 가지고 있다. 즉, 여러 클라이언트에서 동시 수행중인 모든 거래들로 인한 영향이 그 거래들이 직렬 순서로 수행된 결과와 같아야 한다. 그 결과 다음과 같은 상황이 발생한다. 첫째 로크와 같은 것을 얻기 위해 분산된 개체들 사이에 과도한 통신이 요구된다. 둘째 그것을 기반으로 한 기법은 극도로 보수적인 성격을 요구하므로 그것과는 다른 관점에서 볼 때 올바르다고 판단되는 실행이 허락되지 않을 수 있다. 위의 첫째 요인으로 인하여 클라이언트가 서버와 통신할 수 있는 대역폭이 제한적인 방송환경에서는 통신비용이 매우 비싸진다. 또한, 둘째 요인으로 인해 불필요한 질의 거래 철회를 양산하여 바람직하지 않은 상태가 초래된다. 그러므로 이러한 문제점을 보완할 수 있는 정확성 검증 기준 채택 및 그것을 기준으로 하는 효율적인 동시성 제어 기법이 연구되어야 한다.

본 논문의 목적은 다음과 같다. 이 논문에서 우리는 방송환경의 특수성을 극복하고 성능 향상에 기여할 수 있는 효율적인 동시성 제어 기법을 제안하고 평가한다. 우리의 기법은 (1) 상호 일관성과 (2) 현재성이라는 두 가지 특성을 만족한다. 상호 일관성은 (a) 서버가 상호적으로 일치하는 데이타를 유지하고 (b) 클라이언트들은 상호적으로 일치하는 데이타를 읽을 수 있다. 현재성은 클라이언트들이 항상 방송 주기 시작 시점의 최근 데이타를 보는 것을 보장한다.

우리는 실제 방송 시스템의 거래 형태를 고려하여 방송환경에서의 질의 거래 관리를 위해 약한 일관성(weak consistency)[3]이라 불리는 정확성 검증 기준을 적용한다. 그럼으로써, 상호 일관성을 보장하고 질의 거래의 완료율을 최대한 높여 거래 처리율을 향상시키는 방법을 고안하고자 한다. 제안한 기법은 이동 클라이언트에서 수행중인 질의 거래가 항상 서버와 접촉 없이 가장 최근의 그리고 일관성을 만족하는 데이타를 읽도록 허락한다.

2. 관련연구

방송환경에서 거래 개념을 도입한 연구 논문[8,9]도 발표되었으나 이러한 논문들은 리얼타임(real-time)의 쟁점이 없어 방송 기법의 효율성을 살리지 못했다는 문제점이 있다. [8]에서는 이동 클라이언트/서버 컴퓨팅 환경에서 거래들의 직렬화(serialization) 검증 부분을

이동 클라이언트에게 일부분 허락하는 방송 기술을 사용하여 거래들이 실행되어진다. 이 접근법의 초점은 방송 기술의 사용과 검증 작업을 이동 클라이언트에게 부분적으로 맡기는 것과 철회(abort)되어질 거래들을 일찍 알아낼 수 있다는 것이다. 그러나, 이 접근법은 이미 완료된 거래들과 충돌이 일어나는 현재의 거래들은 그것이 어떤 거래일지라도 완료되어질 수 없다. 이 결과 매우 낮은 거래 처리율을 보였다. [9]에서는 거래 관리에 있어서 방송 기술이 push 기반 데이타 전달을 위해 채택되었다. 예를 들어, 서버는 클라이언트의 구체적인 요구 없이 각 데이타 항목의 여러 버전을 버전넘버와 함께 반복적으로 방송한다. 그리고 이 논문에서는 읽기 전용 거래들의 일관성과 현재성을 보장하는 문제가 언급되어진다. 그러나, 이 방법은 한 거래가 수행되는 데 필요한 방송 주기 횟수 증가와 그럼으로써 상당한 응답 시간(response time)의 증가를 가져온다. 이것은 매우 제한적이며 융통성이 부족하다.

그 후 방송 기법을 도입한 OCC-UTS(Optimistic Concurrency Control with Update TimeStamp)[10]도 국내에서 발표되었으나 질의 거래가 많은 방송환경의 특수성을 제대로 감안하지 않아 성능 감소의 문제점을 내포하고 있다. [8]에 비해 거래 처리율이 향상되었고 이동 거래 처리에서 요구되는 무선 통신 횟수를 감소시켰다. 그러나, 이 논문은 질의 거래가 많은 방송환경에 적용될 때 갱신 거래와의 연산 충돌로 인해 많은 질의 거래들의 철회가 일어나며 그 결과 성능 감소가 나타난다[11].

우리와 비슷한 관심사로 동기가 되어 발표된 데이타 주기(datacycle) 접근법[12]은 방송환경에 초점을 맞춘 최초의 동시성 제어 기법이나 무선 환경에 적용하기는 성능상 문제점이 있다. 이 데이타주기 구조는 유선망(network) 분산 데이타베이스시스템의 처리율을 높혔다. 상당히 대역폭이 큰 채널을 가진 유선망에서 데이타베이스의 전체 내용을 클라이언트로 반복적으로 방송한다. 이것은 클라이언트와 서버에서 수행하는 모든 거래들의 전역적 직렬화를 보장한다. 그러나, 서론에서 언급한 것처럼 직렬화는 그것을 보장하기 위해 서버와 무수한 통신이 필요하므로 불안정한 무선 채널을 가진 방송환경에 적용하기는 매우 비용이 많이 들기 때문에 상당한 성능 감소를 유발한다. 우리가 직렬화를 보장하기 위해 필요한 서버와의 상호작용으로 야기된 통신 비용을 피하기 위해서는 클라이언트들은 보수적이어야 하며, 이것은 불필요한 거래의 철회를 발생시킨다.

직렬화 순서 그래프 검사(Serialization Graph Testing: SGT)에 기반을 둔 논문 [13]도 발표되었으나 각 클라이언트의 질의 거래가 직렬화 그래프상 전역적 직렬화를 만족할 때만 수행이 허락되므로 질의 거래가 많은 방송 환경의 동시성 제어 기법으로는 적절하지 않다. 이 논문에서 제시된 접근법은 다음과 같이 작동된다. 각 클라이언트는 직렬화 그래프의 복사본을 지역적으로 유지한다. 서버에서의 직렬화 그래프는 서버에서 완료된 거래들만 포함한다. 또한, 클라이언트측 지역 복사본은 그 사이트에서 재기된 수행중인 질의 거래들을 포함한다. 각 방송 주기에서 서버는 직렬화 그래프상의 모든 갱신된 것들을 방송한다. 갱신 내용을 받자마자 클라이언트는 그것들을 그래프 지역 복사본에 반영한다. 클라이언트측 읽기 연산은 단지 그것이 지역 직렬화 그래프에서 순환을 생성하지 않는다면 실행된다. 그러나, 이 기법은 클라이언트와 서버에서 수행하는 모든 거래들의 정확성 검증 기준으로 직렬화를 택하여 불필요한 질의 거래의 철회를 발생시킨다.

갱신 일관성(update consistency)[3]이라 불리는 새로운 검증 기준을 적용한 논문[1]의 동시성 제어 기법에서는 방송환경에서 질의 거래들이 서버와 접촉 없이 현재성과 일관성을 만족하는 데이타를 읽도록 하였으나 제어 정보를 유지하는데 계산비용이 많이 드는 등 여러 문제점을 가진다. 이 논문에서는 직렬화를 정확성 검증 기준으로 채택하지 않아 위에서 언급한 통신비용과 성능적 측면에서 상당히 우수함을 보인다. 또한, 이 논문에서는 이러한 개념을 바탕으로 새로운 기법인 F-Matrix와 R-Matrix를 선보였다. 그러나, 이 기법들은 앞서 언급한 논문[12]에서의 데이타주기 접근법보다 더 좋은 성능을 보였으나 제어 정보를 유지하기 위해 계산비용이 상당히 비싸고 높은 대역폭을 요구하여 시스템이 알고리즘을 처리하는데 상당한 부담이 있다.

3. 방송환경에서 질의 거래를 위한 직렬화 그래프에 기반을 둔 동시성 제어 기법

새로이 제안하고자 하는 기법은 방송환경에서 질의 거래를 위한 직렬화 그래프에 기반을 둔 동시성 제어 기법(Concurrency Control based on Serialization Graph for Query Transaction : CCSG/QT)이라 칭한다.

이동 거래가 완료되면 거래에 의해 접근된 데이타 항목에 대한 거래 일관성이 보장되어야 한다. 이 논문에서는 이동 갱신 거래들을 검증하기 위해 거래를 우선 수행하고 나중에 유효화 단계를 거치는 낙관적 동시성 제어 기법[14]을 사용한다. 낙관적 기법이 이동 클라이언트들과 서버 사이에 주고받는 메시지 수를 줄여주기 때

문이다. 낙관적 동시성 제어 기법을 사용할 때 서버는 매 거래 완료 시점에서 그 거래를 포함하는 실행이 직렬화를 만족하는지 아닌지를 검사할 필요가 있다. 직렬화를 만족하지 않으면 완료를 하려는 거래는 철회되고 그렇지 않으면 완료를 무사히 마친다. 낙관적 동시성 제어에서 거래를 유효화하는데 있어서 후방향과 선방향의 두 가지 유효화 과정 형태가 있다[15]. 후방향 유효화는 완료하려는 거래가 최근에 완료된 다른 거래에 의해 무효화되어지지 않는다는 것을 보장하는 것을 검사한다. 선방향 유효화는 완료하려는 거래가 다른 어떤 활동중인 거래와 충돌하지 않는다는 것을 보장하는 것을 검사한다.

이동 클라이언트/서버 컴퓨팅 환경에서 이동 클라이언트측 거래는 서버가 실행중인 이동 거래들에 대해 잘 모르기 때문에 후방향 유효화 방법을 선택한다. 클라이언트측은 유효화 될 거래의 읽기 연산 항목 집합이 이미 완료된 다른 거래의 쓰기 연산 항목 집합과 비교되는 순수한 후방향 유효화 방법을 사용한다. (또한, 우리는 윗방향 통신 대역폭의 사용을 줄이기 위해 기본 낙관적 기법과 달리 이동 클라이언트에서 어떤 데이타 충돌이 발견되자마자 해당 거래를 재시작시키는 방법을 사용한다.) 클라이언트측 거래의 유효화 과정 대부분을 그 클라이언트내에서 자체적으로 해결하도록 함으로써 유효화 과정이 진정으로 분산되어 행해질 수 있다.

질의 거래를 위한 본 논문의 동시성 제어 기법에서는 정확성 검증기준으로 직렬화를 바탕으로 하는 엄격한 일관성(strict consistency)[3]을 채택하지 않고 더 느슨한 약한 일관성(weak consistency)[3]을 적용시켜 질의 거래의 철회를 줄이고 데이타의 현재성을 만족시킨다. 약한 일관성이란 직렬화 그래프상 갱신 거래들만으로 이루어진 순환(*cycle*)과 하나의 질의 거래와 갱신 거래들로 이루어진 순환은 허용이 안되고 그 외 순환은 허용하는 일관성 형태를 말한다. 즉, 질의 거래는 갱신 거래들과의 직렬화 순서를 유지하되 동시에 수행 중인 다른 질의 거래들은 그것과 같은 순서를 반드시 유지하지는 않는다.

즉, 약한 일관성은 각 질의 거래가 각각 다르게 갱신 거래들 집합과 직렬가능하도록 허락함으로써 엄격한 일관성을 느슨하게 한다. 질의 거래는 갱신 거래들과 직렬가능하나 다른 질의 거래들과는 직렬가능하지 않다면 약한 일관성을 만족한다고 말한다. 중복 데이타베이스에서의 사용을 위해 [16]에서 처음 소개된 이 형태의 일관성은 질의들이 거래 일관성 있는 데이타를 보는 것을 보장한다. 그러나, 이것은 직렬화 그래프에서 여러 질의 거래와 하나 또는 그 이상의 갱신 거래들을 포함하는 순환(다중 질의 순환)을 허락한다. 하나의 질의와 하나 또는 그 이상의 갱신 거래들을 포함한 순환(단일 질의 순환)과 단지 갱신 거래들만을 포함하는 순환(갱신 거래 순환)은 엄격한 일관성에서처럼 여전히 금지된다. 다중 질의 순환에서 각 질의 거래는 순환 내 갱신 거래들의 상호 비일관적인 순서를 유지한다. 즉, 하나의 질의 거래는 다른 질의 거래들과 비교할 때 갱신 거래들의 다른 직렬 순서를 인지할 것이다. 두 갱신 거래들의 상대적인 순서는 그것들이 어떤 충돌 연산도 발생시키지 않는다면 위치가 바뀌어 질 수 있다.

직렬화를 바탕으로 하는 엄격한 일관성을 정확성 검증 기준으로 채택할 때의 문제점은 관련연구에서 살펴본 논문[1]에 잘 나타나 있다. 첫째, 각각 다른 클라이언트에서 수행 중인 모든 질의 거래들이 동시에 수행중인 갱신 거래들과의 직렬화 순서를 똑같이 유지해야함으로써 불필요한 질의 거래 철회가 많이 일어난다는 것이다. 둘째, 각 질의 거래는 그 질의 거래가 읽은 데이타에 어떤 영향도 주지 않는 갱신 거래를 포함하여 동시에 수행 중인 모든 갱신 거래들과의 사이에 직렬 순서를 유지해야함으로써 또한 불필요한 질의 거래 철회가 발생한다는 것이다.

질의 거래를 위한 정확성 검증기준으로 약한 일관성을 채택할 때의 장점은 실세계 응용 시스템 중 서버 정보가 이동 클라이언트를 통해 주기적으로 바뀌는 교통정보 시스템, 항해 시스템 또는 기상 예측 시스템과 같은 응용에서 잘 나타난다. 이러한 응용 시스템의 서버 정보는 여러 장소에 배치된 이동 클라이언트가 정해진 시간 간격을 두고 보낸 현장 데이타에 의해 기록 유지된다. 따라서, 실시간의 정보와는 약간의 오차가 있을 수 있다. 이러한 응용 시스템에서는 질의 거래의 최대한의 정확한 정보가 사용자에게 빠르게 전달되는 것이 관건이다. 그러므로, 약한 일관성을 적용한 동시성 제어 기법은 이와 같은 응용 시스템에 효과적으로 활용된다.

CCSG/QT의 시스템 모델은 다음과 같다. 이동 클라이언트의 사용자들은 읽기와 쓰기 연산으로 이루어진 이동 거래에 의해 데이타베이스에 자주 접근할 것이다. 모든 거래는 클라이언트에서만 발생하고 한 이동 클라이언트에 의해 어느 시각에 단지 하나의 거래만 실행할 수 있다고 가정한다. 즉, 클라이언트는 한 거래가 완료된 후에야 다른 거래를 발생시킬 수 있다. 또한, (각 이동 거래는 읽기 연산 대상 데이타를 서버에게 요구하여 다음 방송주기에서 최신의 값을 얻는다.) 각 이동 거래는 먼저 읽기 연산에 의해 읽혀진 데이타 항목들의 부

분집합만을 갱신할 수 있고 하나의 데이타 항목은 질의 거래에서 단지 한 번 읽힌다고 가정한다. 데이타베이스는 서버에 의해 저장되고 유지되는 데이타 항목들의 집합이다. 갱신은 이동 클라이언트에 의해 발생되지만 궁극적으로 데이타베이스에 반영되어야 한다. 클라이언트는 서버에게 데이타를 요구하고 서버는 주로 클라이언트들에게 해당 데이타와 제어 정보를 방송하기 위해 방송매체를 사용한다. 또한, 거래의 실행은 원자적(atomic)이다. 즉, 거래는 완료되거나 철회된다.

클라이언트측 거래 일관성을 보장하기 위해 서버는 주기적으로 제어정보를 방송하고 모든 실행중인 이동 클라이언트들은 이 정보를 기다려 그 내용에 따라 그들 거래의 유효화 과정을 진행한다. 서버측의 제어정보테이블(Control Information Table)에는 가장 최근 마지막 방송주기에서 완료된 각 갱신 거래 T와 그것에 의해 갱신된 데이타 항목 집합 $WS(T)$ 및 그 거래에 의해 읽혀진 데이타 항목 집합 $RS(T)$를 유지한다. 유효화 과정의 자세한 사항은 3장 알고리즘에서 언급한다.

이동 거래들의 약한 일관성을 성취하기 위한 이동 클라이언트측 알고리즘은 다음과 같다. 각 이동 클라이언

```
1. On receiving r_T(d) {
        go uplink to the server for d;
        if first operation of query(read_only) transaction T {
                create node T in SG(T) ;
        }
        if T is query(read_only) transaction {
                if ( U_i of DataitemTable[ d, W ] is node of SG(T) )
                        abort T ;
                else
                        copy T to DataitemTable[ d, R ];
        }
        }
2. On receiving commit_T request {
        if T is update transaction {
             send a RequestCommit message, along with T_id, RS(T_id), WS(T_id) and C_i;
        }
        else {commit T; }
        }
3. On receiving the control information table CIT(C_i) {
        if T_id appears in CommitList { commit T; }
        if T_id appears in AbortList { abort T; }
        if WS(any U_j ) ∩ RS(T) ≠ {} {
                ( U_j ∈ all the transactions in CommitList )
                if T is update transaction { abort T; }
        }
        else {
                 if T is update transaction {
                        record C_i ;
                        T is allowed to continue;
                 }
        }
        for ( ∀ U_j ) ( U_j ∈ all the transactions in CommitList ) {
                for (∀ a ∈ RS(U_j)) {
                    if there exists U_i of DataitemTable[a, W] in SG(T) {
                                create node U_j in SG(T) ;
                                create edge from U_i to U_j in SG(T) ;
                    }
                }
                for (∀ a ∈ WS(U_j)) {
                    if there exists U_i of DataitemTable[a, R] in SG(T) {
                                create node U_j in SG(T) ;
                                create edge from U_i to U_j in SG(T) ;
                    }
                }
                copy U_j to DataitemTable[a, R]: (∀ a ∈ RS(U_j))
                update U_j in DataitemTable[a, W]: (∀ a ∈ WS(U_j))
        }
        for ( ∀ U_n ) ( U_n ∈ created node in SG(T) ) {
                for ( ∀ U_m ) ( U_m ∈ all the transactions in CommitList ) {
                    if RS(U_n) ∩ WS(U_m) ≠ {} ( n ≠ m ) {
                                create node U_m in SG(T) ;
                                create edge from U_n to U_m in SG(T) ;
                    }
                }
        }
        }
```

알고리즘 3.1 이동 거래를 처리하기 위한 클라이언트측 알고리즘

트들은 각 거래가 갱신 거래인지 질의 거래인지를 거래 수행 시작시점에서 안다고 가정한다. 각 이동 클라이언트들은 하나의 질의 거래Q가 시작되면 거래 완료시점까지 약한 일관성을 만족하지 않는지 즉, 단일 질의 순환이 발생하는지를 조사하기 위해 Q의 수행시작 이후에 방송 받은 갱신 거래들과의 충돌관계를 표현하는 Q를 포함하는 직렬화 그래프인 $SG(Q)$를 유지하고 순환(cycle)을 검사한다. $SG(Q)$의 정의는 다음과 같다.

[정의] Q를 포함하는 직렬화 그래프 $SG(Q)$

$SG(Q)$의 노드의 집합은 하나의 질의 거래 Q와 Q의 수행시작 이후에 방송 받은 완료된 갱신 거래들로 구성된다. 노드사이의 에지(edge)는 다음의 에지 생성 규칙에 의하여 생성이 된다.

[에지 생성 규칙]

1. For any T committed and broadcasted,
   if $(RS(Q) \cap WS(T) \neq \emptyset)$
   an edge $Q \rightarrow T$ is inserted into $SG(Q)$;
2. For any T committed and broadcasted and any T' in $SG(Q)$,
   if $(RS(T') \cap WS(T) \neq \emptyset$ or $WS(T') \cap RS(T) \neq \emptyset)$
   an edge $T' \rightarrow T$ is inserted into $SG(Q)$; □

정의의 에지 생성 2번 규칙에서 $WS(T') \cap WS(T) \neq \emptyset$는 고려할 필요가 없다. 왜냐하면, 위에서 언급한 시스템 모델에서 각 이동 거래는 먼저 읽기 연산에 의해 읽혀진 데이타 항목들의 부분집합만을 갱신할 수 있다고 했기 때문이다. 위의 규칙에 의해 생성된 $SG(Q)$는 질의 거래 수행시작 때 생성이 되고 질의 거래 완료시까지 유지된다. $SG(Q)$의 노드가 된 거래들은 Q와 관련이 있음을 나타내는 태그(tag)를 가진다. 또한, 그 질의 거래와 동시에 수행되는 갱신 거래들과의 충돌(conflict)을 감지하여 노드들간 에지(edge)를 표현하는데 요구되는 정보를 저장하기 위하여 *DataitemTable*을 유지한다. *DataitemTable*의 각 행은 데이타항목과 그 항목을 읽은 거래들의 리스트인 R리스트 그리고 그 항목을 마지막으로 갱신한 거래에 대한 정보인 W를 가지고 있다. *DataitemTable*은 각 방송 주기 제어정보테이블의 내용인 완료된 갱신 거래 T의 $WS(T)$와 $RS(T)$을 바탕으로 구성이 된다.

각 제어정보테이블이 도착할 때 각 이동 클라이언트는 접근된 데이타에 대한 거래 일관성을 체크하고 알고리즘 3.1과 같이 수행한다.

알고리즘 3.1의 1번 두 번째 if문에서처럼 읽기 연산을 수행하는 과정 중 특히 질의 거래의 읽기 연산은 다음과 같이 처리된다. 그 읽기 연산 대상 데이타가 $SG(T)$ 노드인 임의의 거래가 쓰기 연산한 항목이라면 $SG(T)$에 순환이 발생하므로 그 시점에서 질의 거래는 철회되고 아니라면 *DataitemTable*의 해당 데이타 항목의 R리스트에 그 질의 거래를 명시한다. 첫 번째 if문에서처럼 만약 첫 번째 읽기 연산이면 그 질의 거래를 $SG(T)$의 시작 노드로 생성한다.

마지막 읽기 연산까지 철회 즉, 재시작이 발생되지 않으면 그 질의 거래는 무사히 완료된다. 이와 같이 서버에게 어떤 완료요구 없이 각 클라이언트가 갱신 거래들과의 직렬성을 위배하지 않으며 질의 거래를 독립적으로 처리할 수 있도록 한다. 또한, 질의 거래의 각 읽기 연산시마다 그 동안 완료된 갱신 거래들과의 사이에 직렬성을 만족하지 않을 가능성이 있다면 발견 즉시 그 거래를 철회, 재시작하기 때문에 이른 데이타 충돌 탐지가 가능하다.

알고리즘 3.1의 2번처럼 갱신 거래 T에 대한 완료를 해야할 때 거래의 구분자 T_{id}, $RS(T_{id})$, $WS(T_{id})$ 그리고 해당 방송 주기 C_i와 함께 *RequestToCommit*메시지를 서버에게 보낸다.

그 후에 알고리즘 3.1의 3번처럼 각 제어정보테이블이 방송될 때마다 거래 T에 대한 완료를 요구했던 이동 클라이언트는 접근된 데이타에 대한 거래 일관성을 체크하기 위해 $CIT(C_i)$ 즉, 제어정보테이블과 함께 실려온 *CommitList*와 *AbortList*를 주시한다. 그리고, 해당 T_{id}가 이 두 개의 리스트 중 어느 곳에 있는지를 검사한다. 첫 번째와 두 번째 if문에서처럼 T_{id}가 *CommitList*에 있다면 T는 무사히 완료된다. 그러나, T_{id}가 *AbortList*에 나타나면 T는 철회된다.

알고리즘 3.1의 3번 세 번째 if, else문과 같이 모든 이동 갱신 거래들은 요구하여 방송되어진 데이타 항목을 접근하기 앞서 방송주기 시작시점에서 자체내 부분 유효화 과정이 행해진다. 그것은 마지막 방송 주기에서 완료된 거래들에 대해서 이루어진다. 완료된 거래들은 직렬화 순서에서 유효화 단계의 거래를 앞서기 때문에 유효화 단계의 갱신 거래의 읽기 연산 집합과 완료된 거래들의 갱신된 항목 집합을 비교함으로써 데이타 충돌이 감지된다. 그러한 데이타 충돌이 일어나면 유효화 과정의 갱신 거래를 철회하고 재시작시킴으로써 갱신 거래들의 직렬화 순서가 보장된다.

즉, 각 방송주기 시작시점에서 마지막 방송 주기 Ci에서 갱신 완료된 거래들을 가진 *CommitList*내 임의의 거래 U_j의 쓰기 연산 집합인 $WS(U_j)$와 유효화 과정에 있는 거래 T의 $RS(T)$ 사이에 공통 데이타 항목이 존재

한다면 다음 단계와 같은 과정을 밟는다. T가 갱신 거래일 경우는 갱신 거래들 사이에 비직렬화 가능성으로 인해 철회된다. 그러나, $WS(U_j)$와 $RS(T)$ 사이에 공통의 데이타 항목이 존재하지 않고 T가 갱신 거래일 경우에 무사히 부분 유효화 과정을 마친 후, 서버에게 완료를 요구할 때 사용하기 위해 방송 주기 C_i를 저장한 후 수행을 계속한다.

그리고, 알고리즘 3.1의 3번 첫 번째 for문에서처럼 제어정보테이블이 방송될 때마다 $SG(T)$를 갱신한다. 제어정보테이블과 함께 방송된 *CommitList*내의 완료된 임의의 갱신 거래 U_j의 읽기 연산 집합인 $RS(U_j)$ 중 하나의 항목에 해당하는 *DataitemTable*의 W의 값이 그 시점의 $SG(T)$ 노드 중 하나인 임의의 갱신 거래 U_i라면 또는 U_j의 쓰기 연산 집합인 $WS(U_j)$ 중 하나의 항목에 해당하는 *DataitemTable*의 R 리스트내의 거래가 그 시점의 $SG(T)$ 노드 중 하나인 임의의 갱신 거래 U_i라면 U_i와 U_j사이에 직렬 순서가 존재함으로 다음과 같은 과정을 행한다. $SG(T)$에 거래 U_j 노드를 만들고 U_i에서 U_j로의 에지를 추가한다. 그런후, *CommitList*내의 완료된 임의의 갱신 거래 U_j의 읽기 연산 집합인 $RS(U_j)$의 모든 데이타 항목에 해당하는 *DataitemTable*의 R 리스트에 U_j를 추가하고 U_j의 쓰기 연산 집합인 $WS(U_j)$의 모든 데이타 항목에 해당하는 *Dataitem Table*의 W의 값으로 U_j를 명시한다. 이와 같은 과정을 방송 완료된 모든 거래 Uj에 대해 반복 수행한다.

또한, 알고리즘 3.1의 3번 두 번째 for문에서처럼 한 방송주기 시점에서 함께 방송된 갱신 거래들과의 사이에 직렬 순서가 존재하는지를 검사한다. 즉, $SG(T)$ 노드에 먼저 추가된 갱신 거래 Um의 읽기 연산 집합인 $RS(U_m)$과 $SG(T)$ 노드가 아닌 어떤 방송된 거래 U_n의 쓰기 연산 집합인 $WS(U_n)$인 사이에 공통의 항목이 존재한다면 U_m에서 U_n으로의 직렬 순서가 존재한다. 그러므로, $SG(T)$에 거래 U_n 노드를 만들고 U_m에서 U_n으로의 에지를 추가한다. 이러한 과정을 모든 U_m과 U_n에 대해 반복 수행한다.

서버가 처리해야 할 자세한 내용은 알고리즘 3.2에 나타나 있다.

이동 거래들의 일관성을 유지하기 위해 서버는 이동 클라이언트들로부터 *RequestToCommit*메시지들을 받아 큐(Queue)를 유지한다. 각 *RequestToCommit*메시지 m은 $m.T_{id}$, $m.RS(T_{id})$, $m.WS(T_{id})$, $m.C_i$와 같은 항목필드들을 갖는다.

서버측 유효화 과정은 이동 클라이언트에서 행해진 부분 유효화 과정 때까지는 완료 방송되지 않고 부분 유효화 과정 이후에 완료된 거래들이 있을 수 있기 때문에 필요하다. 그러므로, 이동 클라이언트에서 행해진 마지막 유효화 과정의 방송 주기가 최종 유효화 과정을 위해 서버로 보내진다. T_{id}를 유효화 단계 거래라 하고 C_i는 T_{id}와 함께 딸려온 방송주기 값이라 하자. 즉, T_{id}는 방송주기 C_i때 이동 클라이언트에서 부분 유효화 처리되었다. 이동 거래는 자체 부분 유효화 과정 후 최종 유효화 과정을 위해 서버로 보내어지기 전에 그 거래가 갱신한 데이타 항목을 다른 거래들이 읽거나 쓸 수 있다. $T_c(c = 1, 2,..., m)$를 C_i 이후 완료된 거래들이라 하자. 클라이언트측 알고리즘에서 언급한 것처럼 거래 T_{id}의 읽기 연산 집합과 쓰기 연산 집합을 각각 $RS(T_{id})$와 $WS(T_{id})$로 표기한다.

알고리즘 3.2의 1번처럼 최종 후방향 유효화 과정은 다음과 같다. 서버에서 행해지는 최종 후방향 유효화 과정은 이동 클라이언트에서 행해진 마지막 부분 유효화

```
1. dequeue a message m from Queue {
        vaild = true;
        while ( exist T_c (c = 1, 2, ..... m) ) {
                if WS(T_c) ∩ m.RS(T_id) ≠ {} { valid = false; }
                if not valid { while exit; }
        }
        if not valid {
                put m.T_id in the next AbortList report;
        }
        else {
                put m.T_id in the next CommitList report;
                commit the transaction in the server;
                (i.e., install the values in the m.WS(T_id) into the database)
                record m.T_id, m.WS(T_id) and m.RS(T_id) into the control information table;
        }
        }
2. If it is time to broadcast CIT(C_i) {
        piggyback CommitList and AbortList with CIT(C_i);
        broadcast CIT(C_i);
        broadcast requested data;
        }
```

알고리즘 3.2 이동 갱신 거래를 처리하기 위한 서버측 알고리즘

과정 이후 완료된 갱신 거래들이 있을 수 있기 때문에 반드시 필요하다. 그러므로, 클라이언트에서 행해진 마지막 부분 유효화 과정의 방송 주기 값이 최종 유효화 과정을 위해 서버에 보내져야 한다. T_{id}의 $RS(T_{id})$와 임의의 T_c의 $WS(T_c)$의 교집합이 공집합이 아니면 즉, T_{id}보다 T_c가 먼저 같은 데이타 항목을 갱신하고 완료하였다면 T_{id}는 철회된다. 무사히 완료된 T_{id}의 $WS(T_{id})$는 서버측 데이타베이스에 반영된다. 그 다음 T_{id}, $WS(T_{id})$와 $RS(T_{id})$를 제어정보테이블에 기록한다. 이러한 내용의 제어정보테이블이 다음 방송주기에 각 이동 클라이언트에게 방송되어진다. 클라이언트측 알고리즘에서 언급했듯이 이러한 제어정보테이블은 이동 클라이언트가 자체 내 부분 유효화 과정을 행하는데 그리고 질의 거래의 약한 일관성 검사를 위해 사용된다.

이러한 이동 거래의 완료 또는 철회 결과가 각각 *CommitList*와 *AbortList*에 기록되어 알고리즘 3.2의 2번처럼 각 방송 주기마다 제어정보테이블을 방송할 때 함께 실어 보내진다. 또한, 클라이언트에서 읽기 연산을 위해 요구한 데이타들도 보내진다. 이와 같이, 각 클라이언트에서 수행되는 거래들은 항상 방송 주기 시작 시점의 최근 데이타를 보는 것이 보장된다.

CCSG/QT에서는 질의 거래가 마지막 읽기 연산 시점까지 다른 갱신 거래들과의 비직렬화 가능성이 없는 즉, 거래 일관성이 만족되는 경우에는 다른 갱신 거래들의 완료와 상관없이 무사히 완료된다. 이러한 사항이 예 3.1에서 자세히 설명되어 있다.

[예 3.1] CCSG/QT를 질의 거래가 많은 방송 환경에 적용했을 경우 질의 거래 완료:

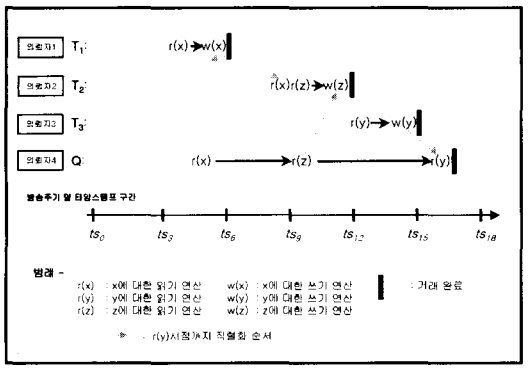

그림 3.5 CCSG/QT를 적용했을 경우 질의 거래의 완료

질의 거래 Q와 갱신 거래 T_1, T_2 및 T_3가 이동 클라이언트측에서 각각 실행하고 있다고 가정한다. 또한, 매3초마다 제어정보테이블을 방송한다고 가정한다. 기존의 동시성 제어 기법이라면 Q는 T_1이 완료되는 시점에서 Q→T_1→Q 또는 T_2가 완료되는 시점에서 Q→T_2→Q의 비직렬화 가능성으로 인해 철회되었을 것이다(그림 3.5).

질의 거래 Q의 첫 읽기 연산 r(*x*)시점에서 *SG*(*Q*)의 시작노드로 Q가 만들어지고 *DataitemTable*의 *x* 항목 R리스트에 Q를 명시한다. r(*x*)시점 이후 T_1은 *x*를 갱신하고 완료한 후 시점 ts_6에서 방송되었다. 이 때, T_1의 쓰기 연산 대상 데이타 항목 *x*에 해당하는 *DataitemTable*의 *x* 항목 R리스트에 Q가 명시되고 Q는 *SG*(Q)의 노드이므로 *SG*(Q)에 노드 T_1이 만들어지고 Q에서 T_1으로의 에지가 생성된다. 그리고, *DataitemTable*의 *x*항목 R리스트에 T_1을 추가하고 W값으로 T_1을 명시한다. r(*z*)시점에서 *DataitemTable*의 *z* 항목의 R리스트와 W값이 비어 있으므로 *SG*(*Q*)에 별다른 변화 없이 수행은 계속된다. *DataitemTable*의 *z* 항목 R리스트를 Q로 명시한다. r(*z*) 이후 T_2가 완료되어 방송되면 T_2의 쓰기 연산 대상 데이타 항목 *z*에 해당하는 *DataitemTable*의 *z* 항목 R리스트에 Q가 명시되고 Q는 *SG*(*Q*)의 노드이므로 *SG*(*Q*)에 노드 T_2가 만들어지고 *Q*에서 T_2으로의 에지가 생성된다. 또한, T_2의 읽기 연산 대상 데이타 항목 중 x에 해당하는 *DataitemTable*의 *x* 항목 W값으로 T_1이 명시되어 있고 T_1은 *SG*(*Q*)의 노드이므로 T_1에서 T_2으로의 에지도 동시에 생성된다. T_2의 읽기 연산 항목이 *x*와 *z*이므로 *DataitemTable*의 *x* 항목 R리스트에 T_2를 추가하고 *z* 항목 R리스트에도 T2를 명시하며 또한 *z* 항목 W값으로 T_2를 명시한다. 그 후, T_3가 ts_{15}시점에서 완료 방송된다. 그 때, T_3가 읽고 쓰기 연산한 데이타 항목 *y*에 해당하는 *DataitemTable*의 *y* 항목 R리스트와 W값에 어떤 거래도 명시되어 있지 않으므로 *SG*(*Q*)에는 별다른 변화가 일어나지 않고 *DataitemTable*의 *y*항목 R리스트에 T_3를 추가하고 W값으로 T_3을 명시한다. Q의 마지막 읽기 연산 r(*y*)시점에서 *DataitemTable*의 *y* 항목의 W값이 T_3이나 T_3는 *SG*(*Q*)의 노드가 아니므로 *SG*(*Q*)에는 별다른 변화가 일어나지 않는다. 질의 거래 Q의 마지막 연산 시점까지 *SG*(*Q*)는 순환을 포함하지 않으므로 Q는 동시에 수행 중인 갱신 거래들 T_1, T_2 및 T_3와의 사이에 약한 일관성이 보장되어 무사히 완료된다. □

4. 정확성 검증

이동 클라이언트측 질의 거래는 어느 읽기 연산 시점에서든지 거래 일치 상태에 있다. 이것은 서버가 단지

갱신 거래들의 직렬가능한 수행만 허락하고 제어정보테이블을 이용하여 거래적으로 일치하도록 함으로써 항상 일관성 있는 데이타베이스 상태를 유지한다. 이동 클라이언트에 의해 접근된 데이타에 대한 일관성 검사가 다음의 정리 4.1에 기반을 둔다.

정리 4.1 이동 클라이언트에서 수행 중인 질의 거래 Q에 의한 읽기 연산 시점 ts_i에서 접근된 어떤 데이타 항목에 해당하는 *DataitemTable*내 W값으로 표현된 거래가 그 시점의 전역적 직렬화 그래프인 $SG(Q)$의 노드에 속하지 않는다면 Q를 포함한 실행은 ts_i시점까지 약한 일관성을 만족한다.

증명: Q를 포함한 실행은 tsi까지 약한 일관성을 만족하지 않는다고 가정하자. 이것은 Q가 거래 시작과 tsi사이 어느 시점에서 $SG(Q)$에 약한 일관성을 위배하는 단일 질의 순환을 포함하고 있다는 것을 의미한다. 즉, Q가 처음 읽기 연산한 데이타 항목을 갱신하고 가장 최근에 완료한 갱신 거래를 T_2라 하고 ts_i시점에서 Q가 읽은 데이타 항목을 가장 최근에 갱신하고 완료한 갱신 거래를 T_3라 하면 $Q \rightarrow T_2 \rightarrow .. \rightarrow T_3 \rightarrow Q$가 존재한다는 것이다. T_2는 Q가 처음 읽기 연산한 데이타 항목을 갱신하고 가장 최근에 완료한 갱신 거래이기 때문에 그 거래를 포함한 *CommitList*와 함께 제어정보테이블이 방송될 때 *DataitemTable*내 해당 데이타 항목의 W값으로 T_2가 보관된다. 그 후 직렬 순서 $T_2 \rightarrow .. \rightarrow T_3$가 존재한다는 것은 거래 T_2가 쓰기 연산한 어떤 데이타 항목을 어떤 임의의 거래가 읽기 연산하거나 또는 T_2가 읽기 연산한 어떤 데이타 항목을 어떤 임의의 거래가 쓰기 연산하며 그 임의의 거래와 T_3 사이에도 같은 관계가 성립함을 의미한다. 그리고, 임의의 거래가 완료되어 방송되는 시점에서 임의의 거래가 읽은 모든 데이타 항목에 해당하는 *DataitemTable*의 R리스트에 임의의 거래가 포함되고 임의의 거래가 갱신한 모든 데이타 항목에 해당하는 *DataitemTable*의 W값으로 임의의 거래가 명시된다. 그 후 T3가 완료되어 방송되는 시점에서도 T_3가 읽은 모든 데이타 항목에 해당하는 *DataitemTable*의 R리스트에 T_3가 포함되고 T_3가 갱신한 모든 데이타 항목에 해당하는 *DataitemTable*의 W값으로 T_3가 명시된다. 또한, $T_3 \rightarrow Q$가 존재한다는 것은 ts_i시점에서 Q의 읽기 연산의 대상 데이타 항목에 해당하는 *DataitemTable*의 W값으로 T_3가 명시되었음을 의미한다. 이것은 Q에 의한 ts_i에서 접근된 어떤 데이타 항목에 해당하는 *Dataitem able* 내 W값의 거래가 그 시점의 $SG(Q)$의 노드에 속하지 않는다는 가정과 모순이다. 그러므로 현재 수행중인 질의 거래 T_1을 포함한 갱신 거래들의 실행은 ts_i시점까지 약한 일관성을 만족한다. □

정리 4.2 (CCSG/QT의 정확성): CCSG/QT는 질의 거래들의 약한 일관성을 만족하는 수행을 보장한다.

증명: 완료하려는 이동 질의 거래를 T_n, 이미 완료된 갱신 거래들을 T_1, T_n-1 등이라 하고 우리는 n>1일 때 $T_n \rightarrow T_1 \rightarrow .. \rightarrow T_n-1 \rightarrow T_n$이 CCSG/QT에 의한 $SG(Q)$에 존재하지 않음을 보일 것이다. 먼저 약한 일관성을 만족하지 않아 그러한 순환이 있다고 가정하자. 즉, T_n이 아직 완료된 상태는 아니고 완료하려는 시점 즉, 마지막 읽기 연산 시점에서 $SG(Q)$에 약한 일관성을 위배하는 순환이 존재한다는 것이다. 이것은 우리가 증명한 정리 4.1에 대하여 모순이다. 즉, 질의 거래는 읽기 연산으로만 이루어진 거래이므로 어느 읽기 연산 시점에서라도 정리 4.1에 의해 다른 갱신 거래들과의 관계에서 약한 일관성을 만족하므로 마지막 읽기 연산 시점 즉, 완료 시점까지 $SG(Q)$는 약한 일관성을 위배하는 순환을 포함하지 않음을 의미한다. 그러므로, CCSG/QT는 약한 일관성을 만족하는 수행을 보장한다. □

5. 성능 평가

이 장에서는 모의 실험을 통해 본 논문에서 제안한 CCSG/QT를 [13]에서 제시한 기존 기법인 SGT(Serialization Graph Testing)의 성능과 비교하여 우리의 기법이 더 우수함을 보이고자 한다. CCSG/QT가 방송 환경을 배경으로 하는 응용 시스템에서 질의 거래를 위한 효율적인 동시성 제어 기법인지를 판단하기 위해서는 클라이언트측 질의 거래의 읽기 연산 수, 클라이언트의 읽기와 서버의 갱신 패턴사이의 겹침(overlap) 및 갱신 수 등의 변수를 이용하여 모의 실험을 통한 검증 작업이 필요하다.

5.1 성능 모델

우리의 성능 모델은 [17]에서 제시된 것과 유사하게 설정한다. 서버는 1에서 방송크기(BroadcastSize) 내의 데이타 항목들의 집합을 주기적으로 방송한다. 우리는 간편하게 서버가 항목들의 집합을 순환적으로 방송하는 평범한 방송 구조를 가정한다. 클라이언트는 방송된 항목들의 부분집합인 범위 1에서 읽기범위(ReadRange)까지의 항목들을 접근한다(단, 읽기범위 ≤ 방송크기). 이 범위 안에서 접근 가능성은 비균일성(non-uniform) 접근을 모델화하기 위한 매개변수 세타(θ)를 가진 Zipf 분포를 따른다. 접근 패턴은 θ가 증가함에 따라 점차적으로 경사가 급해진다. 클라이언트는 쉬는 시간(Think Time)을 기다린 후 다음 읽기 요구를 하게 한다. 그것과 유사하게 서버에서의 갱신은 Zipf 분포를 따라 처리

표 5.1 성능 모델 매개변수

D (방송크기)	1000	c (갱신 거래 당 연산 수)	(u+4*u) / N
갱신범위	500	k (주요 필드의 크기)	1 unit
θ(zipf 분포 매개변수)	0.95	d (다른 필드들의 크기)	5 * k
변위(갱신과 질의 거래 읽기 접근 편차)	0 - 250 (100)	b (버켓 크기)	d units
갱신거래 읽기 범위	1000	질의거래 읽기 범위	250
N(갱신 거래들의 수)	10	쉬는 시간(방송 단위에서 클라이언트 읽기 간격 시간)	2
변위(갱신과 갱신 거래 읽기 접근 편차)	0	질의 거래 당 읽기 수	5 - 50 (10)
u (갱신들의 수)	50 -500 (50)	S(거래 기간)	다양

된다. 쓰기 분포는 범위 1에서 갱신범위(UpdateRange)까지이다. 우리는 클라이언트 접근 패턴과 서버 갱신 패턴사이의 불일치를 모델화하기 위해 변위(offset)이라 불리는 매개변수를 사용한다. 변위가 0일 때 두 분포사이의 겹치는 부분이 최대 즉, 클라이언트의 최대 빈발 접근 페이지(page)가 또한 가장 자주 갱신되어지는 페이지인 경우이다. k 변위는 클라이언트에게 관심이 덜한 갱신 분포 k 항목들을 이동시킨다. 버켓(bucket) 크기에서 버켓이란 방송의 가장 작은 논리적 단위를 말한다.

우리는 각 방송 주기동안 N 거래들이 서버에서 완료되어진다고 가정한다. 모든 서버에서 완료되는 거래들은 모두다 같은 수의 갱신과 읽기 연산을 갖는다. 단, 한

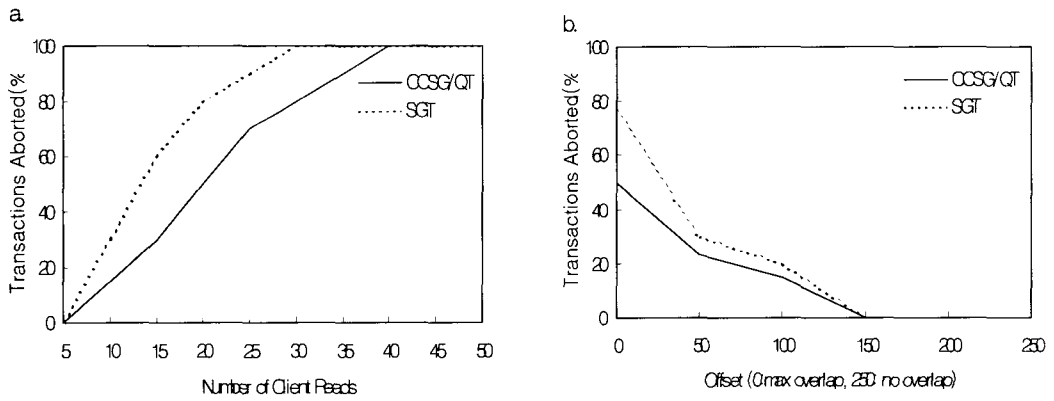

그림 5.1 a. 질의 거래 당 연산 수 b. 변위에 대한 철회율

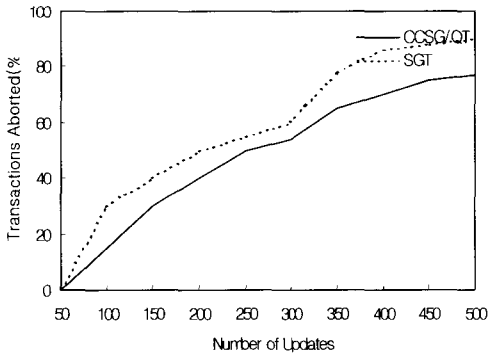

그림 5.2 갱신 수에 대한 철회율

거래당 읽기 연산들은 갱신보다 4배나 더 자주 일어난다. 서버에서 완료되는 거래의 읽기 연산들은 범위 1에서 방송크기까지이고 Zipf 분포를 따르며 서버측 갱신 집합과 0 변위를 가진다. 표 5.1은 매개변수들을 요약한다. 가로 안의 값은 기본 값이다.

5.2 성능 비교

5.2.1 동시성

서버에서의 갱신은 질의 거래들에 의해 읽혀진 데이타 값들을 무효화시키고 다시 새롭게 되도록 한다. 그림 5.1은 CCSG/QT와 SGT기법의 철회율을 보여준다. 질의 거래를 몇 퍼센트(percentage) 완료시키는지 보기 위해 우리는 먼저 질의 거래 당 읽기 연산 수를 다양하게 실험하였다(그림 5.1.a). 전반적으로 CCSG/QT는 SGT보다 완료율이 높으며 읽기 연산 수 20에서 두 기법 사이에 가장 많은 차이를 보인다. 이것은 일관성의 느슨함으로 인해 CCSG/QT의 질의 거래 완료율이 점차 커져 두 기법간의 철회율이 점점 더 큰 차이를 보인다. 그러다가 읽기 연산 수가 20 기준점을 넘어서면 그 질의 거래와 동시에 수행 중인 거래들간에 단순 질의 순환의 가능성도 커짐으로 인해 CCSG/QT의 질의 거래 철회율도 약간 증가함을 알 수 있다.

철회율은 또한 갱신율에 의존하고 클라이언트 읽기와 서버의 갱신 패턴사이의 겹침(overlap)에 의존한다. 그러므로, 우리는 그 겹침을 고려한 실험을 하였다(그림 5.1.b). 예상대로 겹침이 최대일 때 즉, 클라이언트의 빈발 데이타가 가장 자주 갱신되는 데이타일 때 두 기법들의 철회율이 가장 높다. 겹침이 작을(50% 이하) 때 CCSG/QT와 SGT 기법은 모든 거래를 완료시킨다. 그림 5.1.a에서와 마찬가지로 CCSG/QT의 철회율은 전 구간에서 더 낮으며 겹침이 더 클수록 엄격한 일관성을 적용하는 SGT의 기법에서 질의 거래의 완료율은 현저히 떨어지므로 상대적으로 CCSG/QT의 우수성이 더 잘 나타난다.

마지막으로 우리는 갱신 수를 고려한 실험을 하였다(그림 5.2). 갱신의 수 값이 상당히 클 때 철회율의 증가폭이 크다. 이것은 직렬화 그래프에서 순환의 가능성

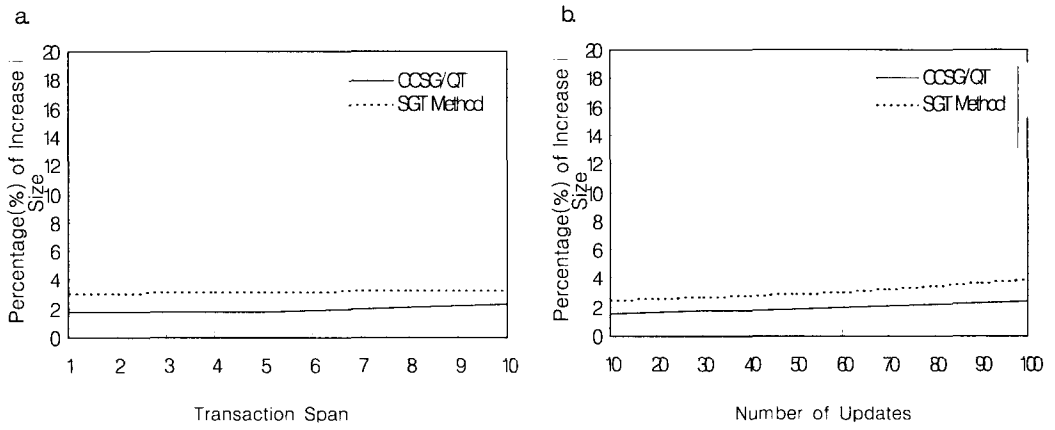

그림 5.3 a. 거래 수행기간(방송주기당 U = 50) b. 갱신 수(수행기간 = 3)에 대한 방송 크기 증가

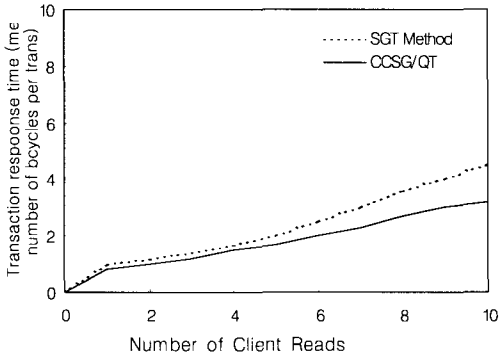

그림 5.4 질의 거래 당 읽기 연산 수에 대한 거래 존속 기간

이 서버에서 완료되는 거래의 연산 수와 함께 증가하기 때문이다. 그러므로, 서버에서 완료되는 거래 연산 수가 작을 때는 완료되는 질의 거래들의 수가 많으나 서버에서 완료되는 거래 연산 수가 많을 때는 완료되는 거래의 증가가 크게 감소됨을 보여준다. 갱신 연산 수가 증가함에 따라 순환의 가능성이 크므로 두 기법 모두 철회율이 증가하나 전반적으로 다중 질의 순환이 허락되는 CCSG/QT가 더 높은 완료율을 보임을 알 수 있다.

5.2.2. 방송 크기

방송내용 크기의 증가는 내용 전달이 대역폭을 소비하기 때문에 제안된 기법들의 효율성의 중요한 척도이다. 더욱이, 방송 데이타의 양은 질의 거래들의 응답 시간에 영향을 준다. 이것은 질의 거래들의 가장 핵심적인 관건이다. 데이타에 대한 접근이 순차적이기 때문에 방송 양이 크면 클수록 클라이언트가 관심을 갖는 데이타가 채널에 나타나는지를 기다리는 시간이 길어진다. 그림 5.3은 [13]에서 개발된 공식을 사용하여 최대 거래 수행기간(span)과 갱신 수(U)의 함수에 따른 방송 크기의 증가를 보인다. SGT 기법은 서버가 완료된 갱신 거래들의 직렬화 그래프를 유지하고 각 방송 주기마다 직렬화 그래프상의 모든 갱신된 내용을 방송해야 하므로 CCSG/QT보다 더 많은 대역폭이 필요함을 볼 수 있다.

5.2.3 지연

우리는 질의 거래 당 평균 방송주기(bcycle)들의 수로서 질의 거래들의 평균 존속 기간인 지연정도를 측정한다. 그림 5.4.는 질의 거래 당 연산 수에 대한 지연을 보여준다. 두 기법 모두 읽기 연산 수가 많으면 많을수록 응답 시간은 증가한다. 특히, SGT기법에서 질의 거래는 방송주기마다 이미 읽기 연산한 대상과 완료된 갱신 거래의 갱신 대상이 일치하여 충돌가능성이 있을 경우 즉, 순환의 가능성으로 인해 철회되어 읽기 연산 수가 많을수록 지연이 길어진다. 이것은 연산 수가 많아질수록 순환 가능성은 더욱 증가하여 지연 정도의 증가치도 커진다. 그러므로, CCSG/QT의 응답 시간이 SGT보다 대체적으로 지연 정도가 더 작다. CCSG/QT의 질의 거래도 연산 수가 많다면 단순 질의 순환의 가능성이 커지므로 철회율 증가로 응답 시간이 길어진다.

6. 결론 및 향후 연구

본 논문에서는 방송환경에서 질의 거래를 위한 효율적인 동시성 제어 기법인 CCSG/QT를 제안했다. 일관성과 현재성을 동시에 요구하는[18] 주식 거래 또는 교통 정보 시스템과 같은 방송 기반 데이타베이스시스템의 여러 응용들이 있다. 본 논문은 이러한 요구사항에 부응하기 위한 효율적인 동시성 제어 기법을 제시하였다. 방송환경의 특수성을 고려한 CCSG/QT는 기존의 이동 컴퓨팅 환경을 위한 동시성 제어 기법과 비교될 때 다음과 같은 장점을 가지고 있다.

첫째, CCSG/QT는 서버에 의해 관리 유지되고 클라이언트에 의해 읽혀지는 데이타의 상호 일관성과 클라이언트에 의해 읽혀지는 데이타의 현재성을 만족시키기에 가장 적절한 정확성 검증 기준인 약한 일관성을 채택하였다.

둘째, CCSG/QT는 약한 일관성 검사를 위해 직렬화 그래프 즉, SG(Q)를 이용함으로써 하나의 질의 거래를 완료하는데 걸리는 시간을 단축했다.

셋째, CCSG/QT는 클라이언트가 서버쪽으로 정보를 요구하는 메시지 수를 최대한 줄임으로써 비대칭적 대역폭을 가진 방송환경의 특수성을 고려하였다.

그러나, 질의 거래를 위한 정확성 검증기준으로 약한 일관성을 채택할 때의 단점은 이동 주식 거래 시스템과 같은 응용에서 나타날 수 있다[19]. 그러므로, 실시간 시스템을 포함 모든 가능한 방송환경하의 이동 컴퓨팅 응용에서 적용될 수 있는 그러면서도 이동 클라이언트의 한계성을 극복할 수 있는 성능 향상된 기법에 대한 연구가 숙제라 할 수 있다. 또한, 이동 클라이언트 내의 질의 거래들뿐만 아니라 갱신 거래들을 좀 더 효율적으로 처리할 수 있는 최적화(optimization) 알고리즘을 캐싱 기법과 함께 연구해 볼 필요가 있다.

참 고 문 헌

[1] J. Shanmugasundaram, A. Nithrakashyap, R. Sivasankaran, and K. Ramamritham, "Efficient Concurrency Control for Broadcast Environments", *ACM SIGMOD*, 1999.

[2] P. A. Bernstein, V. Hadzilacos and N. Goodman, "Concurrency Control and Recovery in Database Systems", Addison Wesley, Reading, Massahuetts, 1987.

[3] P. M. Bober and M. J. Carey, "Multiversion Query Locking", *Proceedings of the VLDB Conference*, Vancouver, Canada, August 1992.

[4] D. Barbara and T. Imielinsky, "Sleepers and Workholics: Caching in Mobile Environment", *Proceedings of ACM SIGMOD Conference on Management of Data*, pp. 1-12, June 1994.

[5] J. Jing, A. Elmagarmid, A. Helal and R. Alonso, "Bit-Sequences: An Adaptive Cache Invalidation Method in Mobile Client/Server Environment", *ACM/Baltzer Mobile Networks and Applications*,

Vol.2, No.2, 1997.
[6] K. L. Wu, P. S. Yu and M. S. Chen, "Energy-efficient Caching for Wireless Mobile Computing", *Proceedings of the 12th International Conference on Data Engineering*, pp. 336-343, Feb. 1996.
[7] C. F. Fong, C. S. Lui and M. H. Wong, "Quantifying Complexity and Performance Gains of Distributed Caching in a Wireless Network Environment", *Proceedings of the 13th Inter ational Conference on Data Engineering*, pp. 104-113, April 1997
[8] D. Barbara, "Certification Reports: Supporting Transactions in Wireless Systems", *Proceedings of the 17th International Conference on Distributed Computing Systems*, pp. 466-473, May 1997
[9] E. Pitoura, "Supporting Read-Only Transactions in Wireless Broadcasting", *Proceedings of the 9th International Workshop on Database and Expert Systems Applications*, pp. 428-433, 1998.
[10] S. Lee, C. Hwang, W. Lee and H. Yu, "Caching and Concurrency Control in a Mobile Client/Server Computing Environment", 한국정보과학회 논문지, Vol 26, No. 8, August 1999.
[11] U. Lee, B. Hwang, "방송환경에서 이중 버전과 타임스탬프에 기반을 둔 낙관적 동시성 제어 기법", 한국정보처리학회 논문지, Vol 8-D, No. 2, May 2001.
[12] G. Herman, et. al, "The Datacycle Architecture for Very High Throughput Database Systems", *Proceedings of the ACM SIGMOD Conference*, 1987.
[13] E. Pitoura and P. Chrysanthis, "Scalable Processing of Read-Only Transactions in Broadcast Push", *International Conference on Distributed Computing Systems*, Austin, 1999.
[14] H. T. kung and J. T. Robinson, "On Optimistic Methods for Concurrency Control", *ACM Transactions on Database Systems*, Vol.6, No.2, pp.213-226, June 1981.
[15] T. Harder, "Observations on Optimistic Concurrency Control Schemes", *Information Systems*, Vol.9, No.2, pp.111-120, 1984.
[16] H. Garcia-Molina and G. Wiederhold, "Read-Only Transactions in a Distributed Database", *ACM Transactions on Database Systems*, 7(2), June 1982.
[17] S. Acharya, R. Alonso, M. J. Franklin, and S. Zdonik, "Broadcast Disks: Data Management for Asymmetric Communication Environments", *In Proceedings of the ACM SIGMOD Intl. Conference on Management of Data (SIGMOD 95)*, June 1995.
[18] P. Xuan, et. al, "Broadcast on Demand-Efficient and Timely Dissemination of Data in Mobile Environments", *IEEE Real-Time Technology and Applications Symposium*, pp.38-48, June 1997.
[19] Lee V., Son S. H., Lam K. "On the Performance of Transaction Processing in Broadcast Environments", *Int Conf on Mobile Data Access (MDA'99)*, Hong Kong, Dec 1999.

이 욱 현

1992년 이화여자대학교 전자계산학과(이학사). 1997년 한국과학기술원 정보및통신공학과(공학석사). 2000년 전남대학교 전산학과 박사 수료. 2002년 ~ 인하대학교 강의전임강사. 관심분야는 분산데이타베이스, 이동컴퓨팅, 객체지향시스템, 데이타 마이닝 등

황 부 현

1978년 숭실대학교 전산학과(학사). 1980년 한국과학기술원 전산학과(석사). 1994년 한국과학기술원 전산학과(박사). 1980년 ~ 현재 전남대학교 컴퓨터정보학부 교수. 2001년 ~ 현재 전남대학교 정보통신연구소장. 관심분야는 분산시스템, 분산데이타베이스, 객체지향시스템, 전자상거래