가상대학을 지원하기 위한 트랜젝션 모형

전우천*, 홍석기**
서울교육대학교 컴퓨터 교육과*, 건국대학교 경영학과**

요 약

최근 들어 인터넷의 발달 등에 힘입어 가상대학에 관한 관심이 증가하고 있으나, 기존의 가상대학 구축을 위해 적용되는 기술은 가상대학의 구현에 있어서 미비한 점이 많다. 특히, 현재의 트랜젝션 모형들은 가상대학을 지원하기 위해 많은 점이 보완되어야 한다. 본 연구에서는, 가상대학을 지원하기 위한 트랜젝션 모형을 제안한다. 본 연구에서는 첫째, 가상대학의 일반적인 환경에 대해서 논의하고 둘째 가상대학 지원을 위한 트랜젝션의 요구사항 들을 제시한다. 이 요구사항 들은 온라인 처리, 트랜젝션 길이의 유연성, 비예측성, 상호 작용성 또한 병렬성 등을 포함한다. 이 요구사항을 만족하는 트랜젝션 모형을 제안하고 또한 트랜젝션 처리 기법을 제안한다. 마지막으로, 앞으로의 연구과제 들을 제안한다.

A Transaction Model That Supports Virtual Universities

Woochun Jun*, Sukki Hong**

Seoul National University of Education, Dept. of Computer Education*
Konkuk University, Dept. of Business Administration**

ABSTRACT

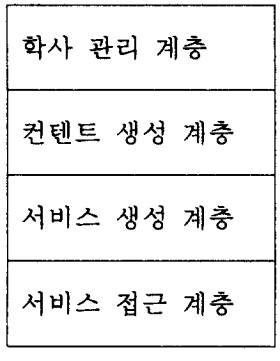
Recent advances in technologies such as internet have led to the emergence of virtual universities. However, existing technologies have many problems to be applied on virtual university implementation. Especially, existing transaction models are not appropriate to support virtual universities. In this paper, we present a transaction model to support virtual universities. First, we discuss general virtual university environments. After that, we introduce transaction model requirements for virtual universities. Those requirements include on-line processing, flexibility on transaction length, unpredictability, interactivity and parallelism. Then, we present a transaction model reflecting those requirements, and transaction processing techniques. Finally, we discuss future research issues based on our transaction model and processing techniques.

1. 서론

최근의 인터넷 기술과 통신 기술의 발달에 기초하여 가상대학에 대한 관심이 고조되고 있다. 가상대학은 교수와 학생이 일정한 강의실에서 일정한 시간에 만나 면대면 (Face-To-Face) 수업을 하는 전통적인 방식에서 벗어나 원격교육 (Distance Learning)에 기초한 학습자 위주의 교육에 기초를 둔다 [1].

가상대학은 일부대학에서 일부과목에 대해서 시험적으로 운영되는 형태에서 벗어나 점차 정규과목으로 확대 운영되고 있다. 또한, 현재 국내외에서 약 230개의 가상대학이 운영되고 있으며, 재교육 등 다양한 수요계층으로 말미암아 앞으로 가상대학에 대한 관심과 현재 제한되어 있는 교과목에 대한 개설도 크게 늘어나리라 전망된다 [1].

가상대학을 구현하기 위해서는 다양한 기술적 기반과 교육 방법 등이 제공되어져야 하며 또한 통합되어야 한다. 특히, 가상대학을 위한 기술적 기반으로서는 4 계층이 필요하다 [1]. <그림 1>은 그 계층 구조를 보여 준다.



<그림 1> 가상 대학의 기술 계층 구조 ([1])

제일 하부 계층인 서비스 접근 계층은 학습자들이 교육 컨텐트 (Content) 와 같은 상위 계층에 접근할 수 있는 물리적인 환경을 제공한다. 즉, 전송 프로토콜과 이에 관련된 네트워크 기술 등이 이 계층에 포함된다. 서비스 생성 계층에서는 개별학습/그룹학습, 동기/비동기 교육 등 다양한 교육 방법과 교수-학생간의 상호 작용을 지원한다. 콘텐트 생성 계층에서는 다양한 저작 도구를 관리하고 또한 컨텐트 데이터베이스 서버 관리 및 운영을 담당한다. 마지막으로, 학사 관리 계층에서는 입학, 등록, 수강에 관한 서비스를 담당한다.

가상대학이 기존의 면대면 형태의 교육을 보충하기 위해서는 멀티미디어를 이용한 교육이 보편화 될 전망이다 [2]. 가상대학에서 이루어지는 멀티미디어 교육은 기존의 단순한 지식전달 차원에서 벗어나 학습자의 자율성을 강조한다. 즉, 멀티미디어 매체는 일방적인 의사 소통이 아닌 양방향 의사소통을 가능케 하여, 교수와 학생,학생과 학생 사이의 상호 작용을 유도한다.

멀티미디어 자료들을 지원하기 위한 데이터 베이스로서 객체지향 데이터베이스 (Object-Oriented Database)가 가장 보편적으로 이용되고 있다. [1,3]. 앞으로도 객체지향 데이터 베이스는 가상대학을 지원하는 보편적인 데이터 베이스가 되리라 전망된다.

일반적으로 데이터 베이스에 대한 논리적인 작업은 여러 개의 연산으로 구성된다. 예를 들어, 은행의 고객 예금이체는 한 구좌에서 돈을 인출하여 다른 구좌로 적립하는 작업이다. 이 고객 예금이체 거래에서, 인출과 적립이 동시에 일어나거나 동시에 일어나지 않게 하여 데이터 베이스의 일치성 (Consistency)을 유지하는 것은 매우 중요하다. 즉, 예금이체는 원자성 (Atomicity) 을 만족해야 한다. 트랜젝션이란 데이터 베이스에 대해 하나의 논리적인 작업을 이루기 위한 여러 연산 (Operation) 으로 구성된다 [4]. 일반적으로, 트랜젝션은 4가지의 특성 (ACID 특성)을 만족해야 한다. 즉, 트랜젝션 내의 연산들은 모두 실행되거나 아니면 하나도 실행되지 않아야 하며 (Atomicity: 원자성), 한 트랜젝션의 실행은 데이터 베이스를 일관성을 유지해야 하며 (일치성: Consistency), 트랜젝션이 종료되기 전에는 중간결과를 다른 트랜젝션에게 공개할 수 없고 (고립성: Isolation), 마지막으로, 트랜젝션이 종료되면, 그 결과는 데이터 베이스에 기억이 되어야 한다 (영속성: Durability).

지금까지, 객체지향 데이터 베이스에 접근하기 위한 여러 트랜젝션 모형들이 제안되었다 [5,6]. 그러

나, 가상대학을 구현하기 위해서는 여러 가지 특성들, 즉 온라인 (On-Line), 실시간 (Real-Time), 상호작용성 (Interactivity), 원거리 접근 (Remote Access), 등이 고려되어야 하며 이러한 특성들이 트랜젝션 모형에 반영되어야 한다.

본 논문에서는 가상대학을 지원하는 멀티미디어 객체지향 데이터 베이스의 트랜젝션 모형을 제안한다. 제안된 모형은 가상대학을 지원하기 위한 특성 등을 모두 포함한다.

본 논문의 구성은 다음과 같다. 제2장에서는 가상대학을 구현하기 위해 일반적으로 요구되는 환경에 대해서 논의한다. 제3장에서는 이러한 가상대학의 환경을 구현할 수 있는 트랜젝션의 요구 사항들을 제시하고, 제4 장에서는 이 요구사항들을 기반으로 새로운 트랜젝션 모형을 제시하고 또한 트랜젝션 처리 (Transaction Processing) 기법을 제시한다. 마지막으로 제5장에서는 본 논문의 결론과 앞으로의 연구 과제를 제시한다.

2. 가상 대학 환경 연구

제 2 장에서는 먼저 가상대학 구현을 하기위한 환경에 대해서, 학습 데이터 베이스를 접근하여 트랜젝션을 처리하기 위한 관점에서 논의한다.

먼저, 궁극적으로 가상대학의 구현에 요구되는 사항은 실시간 (동기) 교육이라 할 수 있다 [1,2]. 즉, 인터넷 등 통신망을 이용해서 일방향 및 양방향으로 학습자가 교수나 교육 콘텐트와의 실시간 접속을 필요로 한다. 특히 멀티미디어 데이터는 시간 동기화 (Temporal Synchronization)가 되어야 정확한 정보가 전달된다 [7]. 또한, 온라인 토론도 주어진 강의에 대한 자신의 견해를 발표하고, 교수 및 다른 학생과 의견교환을 할 수 있는 유용한 수단이다.

실시간 교육과 더불어, 가상대학의 구현은 상호작용성 (interactivity) 을 필요로 한다. 즉, 멀티미디어와 학습자간의 상호 작용, 학습자와 학습자 사이, 학습자와 교수 사이의 상호작용을 필요로 한다. 특히, 멀티미디어와의 상호작용은 단순히 정보를 전달받는 차원을 넘어서 피드백 (Feedback)을 이용해 학습 목표를 향해 갈 수 있기 때문에 매우 중요하다 [2].

가상대학은 원격교육에 기초를 두었기 때문에 필연적으로 원격 데이터 베이스 접근 (RDA : Remote Database Access)을 필요로 한다. 원격 데이터 베이스 접근은 통신망을 이용하여 구현될 수 있고 이미 표준화된 서비스가 제공되고 있다 [8].

한편, 가상대학을 지원하는 멀티미디어 객체지향 데이터베이스에서는 사용자가 객체에 접근할 때 객체간의 관계를 이용하는 네비게이션 (Navigation)을 이용하게 되고 또한 멀티미디어 데이터의 용량의 크기로 인해서, 데이터베이스에 접근하는 트랜젝션의 처리 시간을 길게 한다 [9].

마지막으로, 가상대학에서는 학습자가 스스로가 자료를 찾아서 학습을 할 수 있는 셀프 서비스 (Self-Service) 환경을 제공해야 한다 [10]. 이러한 환경을 통해서 학습자가 멀티미디어 또는 텍스트를 이용하여 다양한 지식을 스스로 획득할 수 있도록 도와준다.

3. 트랜젝션의 요구사항

제 3 장에서는 앞 장에서 논의된 가상대학의 특성에 기초하여 가상대학 구현을 위한 트랜젝션 모형과 처리안을 제안한다.

트랜젝션이 만족해야 되는 가장 큰 요구 사항으로서 일치성을 항상 유지해야 한다. 일반적으로, 데이터 베이스 시스템의 특징은 데이터를 공유함에 있다. 즉, 데이터 베이스 시스템은 데이터를 여러 사용자가 동시에 접근할 수 있도록 한다. 이 같은 다중 사용자 시스템 (Multiple User System)을 가정할 경우 직렬성 (Serializability)이 현재 가장 널리 쓰이는 정확도 기준 (correctness criterion)이다. 여러 트랜젝션 내의 연산이 서로 섞여서 실행될 경우, 트랜젝션들이 임의의 순차적인 순서로 실행된 것과 같은 결과를 발생시키면, 직렬화를 만족한다고 정의한다 [11].

가상대학 구현을 위한 트랜젝션이 만족해 하는 가장 큰 조건은 트랜젝션의 길이가 유연해야 한다. 즉,

가상대학의 트랜젝션들은 크기가 큰 멀티미디어 정보를 이용하고 또한 네비게이션을 이용하기 때문에 길이가 길게 된다. 이와 같이 트랜젝션의 길이가 길게되면 경우 다음과 같은 문제가 발생한다. 먼저, 트랜젝션의 회복을 위해 재수행되야 되는 경우 길이가 길수록 재수행을 위한 시간과 비용이 증가하게 된다. 한편,트랜젝션의 수행동안에 필요한 자원들을 다른 트랜젝션들이 이용할 수 없을 가능성이 있기 때문에 전체적인 처리능력을 저하시킨다.

또한, 가상대학 구현을 위해서는 사용자 (학습자)의 비예측성 (Unpredictability)에 대비해야 한다. 사용자가 온라인으로 반응하기 때문에 트랜젝션 내에서 어떤 작업을 할 것인지를 예측이 힘든다. 따라서, 사용자에게 트랜젝션을 동적으로 관리할 수 있는 권한, 예를들어 임의의 시점에서 시작하고 종료할 수 있는 권한, 등을 주어야 한다.

가상대학의 구현을 위해서는 트랜젝션 모형에 상호 작용성을 반영해야 한다. 특히, 학습자와 학습자. 또는 학습자와 교수사이에 공동작업을 지원해야 한다. 즉, 공유한 데이터를 서로 주고 받으면서 작업을 완성할 수 있고 또한 공동의 작업을 하는 도중 작업을 서로 나누어 각자 해야 하는 등 하나의 큰 작업을 완성하기 위한 협력작업을 지원해야 한다. 예를들어, 학습자 A의 작업이 완전히 종료되지 않은 상황에서 학습자 B가 A의 현재까지의 작업을 볼 수 있게 하는 것이 공동작업에 도움을 줄 수 있다.

가상대학의 트랜젝션은 실시간 (Real-Time) 처리를 지원해야 한다. 즉, 각 트랜젝션은 일정한 시간에 처리가 완료가 되어야 하므로, 처리마감시간 (Deadline)을 갖게 된다. 따라서, 모든 트랜젝션은 처리마감시간이나 또는 웅급성 (Criticalness)에 따라 우선순위 (Priority)를 갖게 된다. 일반적으로, 실시간 처리에서는 처리마감시간이 빠르거나 웅급성이 큰 트랜젝션이 높은 우선순위를 가진다.

한편, 트랜젝션의 처리 속도를 높이기 위해서 병렬성 (Parallelism)을 지원해야 한다. 즉, 일치성을 위반하지 않는 범위에서, 트랜젝션을 동시에 병렬 수행할 경우 트랜젝션의 반응시간이 짧아진다. 병렬성은 객체지향 데이터 베이스에서 다음과 같이 이용될 수 있다. 멀티미디어 정보 등을 지원하는 객체지향 데이터 베이스에서는 객체에 접근하기 위한 수단으로 메소드 (Method)를 이용한다. 즉, 메소드는 객체에 관한 정보를 읽거나 또는 수정하기 위한 일종의 프로시쥐 (Procedure)이다. 두 메소드가 한 객체의 서로 다른 데이터에 접근하는 한, 두 메소드는 동시에 수행되어도 일치성을 위반하지 않는다. 따라서, 병렬성을 이용하여 전체적인 트랜젝션의 반응시간을 단축시킬 수 있다.

마지막으로, 학습자가 온라인으로 비예측적으로 작업을 하기 때문에, 트랜젝션의 수행에 필요한 코드를 미리 만드는 것은 현실적으로 매우 어렵다. 따라서, 학습자가 학습을 진행함에 따라서 수행에 필요한 연산이 정해지게 된다. 즉, 사용자 조정 트랜젝션 (User-controlled Transaction)이 미리 프로그램된 트랜젝션 (Programmed Transaction)보다 더 유용하게 사용될 수 있다.

4. 트랜젝션 모형

4.1. 연구 배경

제 3장에서 논의한 것과 같이, 가상대학 구현을 위한 트랜제션은 트랜젝션의 길이, 비예측성, 상호작용성, 또한 병렬성, 실시간 처리 등을 제공해야 된다. 따라서, [12] 에서 제의한 단순 트랜젝션 (Flat Transaction) 모형은 가상대학 환경에서 부적합하다.

본 연구에서 제시하는 트랜젝션 모형은 Split/Join 트랜젝션 모형 [13,14]과 중첩 트랜젝션 (Nested Transaction) 모형 [6]에 기초를 두고 있다. 그러나, 두 모형 모두 가상대학 구현을 위한 트랜젝션 요구사항 모두를 지원하지 못한다. 본 연구에서는 두 모형을 결합하여 새로운 모형을 제안한다.

Split/Join 트랜젝션은 기본적으로 진행중인 트랜젝션의 구조를 동적으로 변화시켜 유연성 (Flexibility)의 제공과 효율적인 자원 관리 등을 목적으로 한다. Split 트랜젝션은 진행중인 한 트랜젝션을 두개의 직렬화 가능한 트랜젝션으로 만든다. 이러한 경우에, 하나의 공동작업을 진행중이던 두 학습자가 각자 고유한 환경으로 돌아가 작업을 진행할

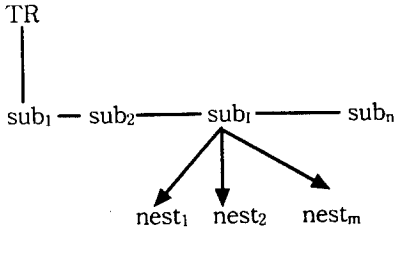
수 있게 해준다. 즉, 두개의 독립된 트랜젝션으로 진행이 될 수 있다. 이러한 경우, 하나의 긴 트랜젝션으로 진행되는 경우의 비효율성 (즉, 자원의 동결, 재수행 비용증가) 을 감소시킬 수 있다.

Join 트랜젝션은 Split 트랜젝션과는 반대로 진행중인 두개의 직렬화 가능 트랜젝션을 하나의 트랜젝션으로 결합시킨다. 가상학습에 있어서, 두 학습자가 각기 진행하던 작업을 합쳐서 하나의 공동 트랜젝션으로 만들 수 있다.

중첩 트랜젝션 모형은 서로 동시에 수행될 수 있는 상위 트랜젝션 (Top-level Transaction) 으로 구성된다. 또한, 하나의 상위 트랜젝션은 여러 개의 스텝들로 구성이 되며, 각 스텝은 더 이상 분리할 수 없는 원자 연산 (Atomic Operation)이나 또 종속 트랜젝션 (Subtransaction)으로 구성된다. 이 종속 트랜젝션은 상위 트랜젝션이나 다른 종속 트랜젝션과 함께 동시에 수행될 수 있다. 한편, 종속 트랜젝션은 다른 종속 트랜젝션을 가질 수 있다. 따라서, 중첩 트랜젝션은 단순 트랜젝션과 달리, 트랜젝션 내부의 병렬성을 이용할 수 있다.

4.2. 제안된 트랜젝션 모형

본 논문에서 제안하는 트랜젝션은 Split/Join 트랜젝션과 중첩 트랜젝션 모형을 결합하여, 가상대학 구현을 위한 트랜젝션의 요구사항을 만족시키고 또한 트랜젝션이 실시간 처리가 가능하도록 각각의 트랜젝션에 실시간 우선 순위를 부여한다. 기 기본구조는 <그림 2>에 있다.



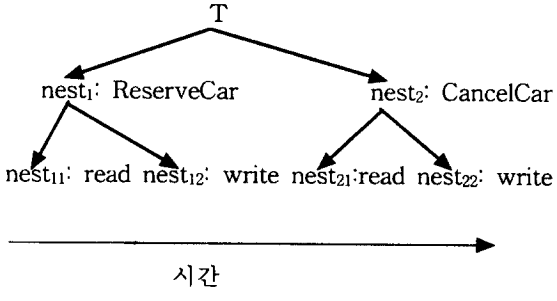
<그림 2> 트랜젝션 모형

TR은 학습자가 처음에 시작한 글로벌 트랜젝션 (Global Transaction) 이며 진행함에 따라 다양한 형태로 결합되고 또한 분리될 수 있다. 또한, 사용자의 의도에 따라, 아무 변형 없이 종료될 수 도 있다. sub_1, sub_2,,,,sub_n 은 TR이 시작한 후에 변형된 종속 트랜젝션이나 또는 분리되거나 결합된 트랜젝션을 나타낸다. (화살표가 없는 선은 구조가 다양하게 변할 수 있음을 나타낸다). 마지막으로, $nest_1$, $nest_2$,,,, $nest_m$은 중첩 트랜젝션에서 시작한 종속 트랜젝션을 나타낸다. 본 모형에서는 중첩 트랜젝션이 시작하면, 종료될 때까지 Split/Join 트랜젝션을 허용하지 않는다.

본 모형에서는 닫힌 중첩 트랜젝션 (Closed Nested Transaction) 모형을 채택한다 [6]. 닫힌 중첩 트랜젝션 모형은 열린 중첩 트랜젝션 (Open Nested Transaction) [15]과는 달리, 종속 트랜젝션이 종료되었더라도 그 결과를 다른 트랜젝션에 보이지 않는다. 즉, 닫힌 중첩 트랜젝션은 종속 트랜젝션의 수행 결과를 상위 트랜젝션이 종료되어야만 공개할 수 있다. 따라서, 높은 병렬성을 유지하기는 어렵지만, 트랜젝션의 실패에 따른 이미 수행된 연산에 대한 보상 트랜젝션 (Compensating Transaction)의 문제는 해결된다. 특히, 가상대학의 운영은 실시간과 온라인으로 운영이 되기 때문에 , 보상 트랜젝션이 정의되기 어려운 문제를 내포한다.

<그림 3>은 간단한 중첩 트랜젝션의 예를 보여준다. 자동차 대여를 위한 객체지향 데이터 베이스를 가정하고, 두 메소드 ReserveCar와 CancelCar를 가정한다. 두 메소드를 구성하는 연산을 <그림 3>과 같다고 가정한다. 이 경우에 상위 트랜젝션 T가 종속 트랜젝션 $nest_1$과 $nest_2$를 시작해 두 메소드 ReserveCar와 CancelCar를 수행한다. 마찬가지로, $nest_1$ ($nest_2$) 는 $nest_{11}$과 $nest_{12}$ ($nest_{21}$과 $nest_{22}$) 를 시작해서 Read와 Write를 수행한다. 트랜젝션이 종료되기 위해서는 각기 트랜젝션이 시작한 종속 트랜젝션 들이 모두 종료되어야 비로소 종료된다. 즉, $nest_1$이 종료되기 위해서는 $nest_{11}$과 $nest_{12}$가 성공적

으로 종료되어야 하며, 마찬가지로 T가 종료하기위해서는 T가 시작한 모든 종속 트랜젝션들, 즉, $nest_1$, $nest_2$, $nest_{11}$, $nest_{12}$, $nest_{21}$, $nest_{22}$ 가 모두 성공적으로 종료되어야 한다.



<그림 3> 중첩 트랜젝션의 예

4.3. 사용자 명령어

이 절에서는 4.2절에서 제시한 트랜젝션 모형에 기초하여 사용자의 명령어들을 소개한다. 기본적으로, Split/Join 트랜젝션을 위한 명령어는 [13]에 기초를 두고있으며, 또한 중첩 트랜젝션의 수행과 관련된 명령어들을 제시한다.

- Begin-Transaction
- Abort-Transaction
- Commit-Transaction
- Read-Data
- Write-Data
- Split-Transaction
- Join-Transaction
- Nest-Transaction
- Sub-Transaction
- Abort-Nest-Transaction
- Abort-Sub-Transaction
- Commit-Sub-Transaction
- Commit-Nest-Transaction
- Commit-Split-Transaction
- Accept-Join-Transaction
- Suspend-Transaction
- Resume-Transaction
- Transaction-Priority

Begin-Transaction, Abort-Transaction, Commit-Transaction 은 일반적인 데이터 베이스의 경우와 같은 의미를 가진다. 즉, 사용자가 트랜젝션을 시작하고 (Begin-Transaction), 종료하며 (Commit-Transaction), 또한 의도적으로 트랜젝션을 재수행시킬 수 있다 (Abort-Transaction).

Read-Data와 Write-Data는 각각 데이터 베이스를 읽고 또한 수정하기 위한 명령어로서, 낮은 레벨에서 수행하는 원자 연산부터 상위 레벨의 메소드까지 응용분야에 따라 다양한 개념을 포함한다.

Split-Transaction명령어는 하나의 트랜젝션 T를 두개의 직렬화 가능 트랜젝션, T_A 와 T_B로 아래와 같이 분리시킨다 [13].

Split-Transaction (

T_AReadSet, T_AWriteSet, T_AUser,

T_BReadSet, T_BWriteSet, T_BUser)

T_AReadSet과 T_AWriteSet (T_BReadSet과 T_BWriteSet) 은 각각 T_A (T_B)에 의해 읽히고 또한 수정되는 데이터들의 집합을 나타낸다. 또한, T_AUser와 T_BUser는 각각 T_A와 T_B를 주관하는 사용자를 나타낸다. T_A가 T_B에 앞서 직렬화 되기 위해서는 다음과 같은 조건을 만족해야 한다.

1) T_AWriteSet ∩ T_BWriteSet ⊆ T_BWriteLast

2) T_AReadSet ∩ T_BWriteSet = Ø

3) T_BReadSet ∩ T_AWriteSet ≠ Ø

1)은 T_A 는 T_B의 수정된 내용을 고칠 수 없지만 반대로 T_B는 T_A 의 수정된 내용을 고칠 수 있음을 의미한다. 2)는 T_A가 T_B의 수정된 내용을 읽지 않았기 때문에 T_A가 T_B에 앞서 직렬화가 가능함을 나타낸다. 3)은 T_B 는 T_A 의 수정된 내용을 읽을 수 있다는 것을 나타내기 때문에, T_A 가 T_B보다 앞서서 실행된 트랜젝션 임을 나타낸다.

다음 세가지 조건을 만족하면, T_A 와 T_B 가 서로 독립적인 경우를 나타낸다. 따라서, T_A 와 T_B는 임의의 순서로 직렬화가 될 수 있다 [13].

a) T_AWriteSet ∩ T_BWriteSet = Ø

b) T_AReadSet ∩ T_BWriteSet = Ø

c) T_BReadSet ∩ T_AWriteSet = Ø

Join-transaction은 두개의 트랜젝션을 하나의 트랜젝션으로 만든다. 예를들어, Join-Transaction ($T:T_A$) 의 경우는 T_A를 T로 결합시킨다. 이 경우에, T_A는 T의 마지막으로 합쳐진다 [13].

Nest-Transaction은 사용자가 객체지향 데이터 베이스를 접근하여 작업을 시작할 때 쓰인다. 마찬가지로, Sub-Transaction은 종속 트랜젝션을 시작할 때, 즉 메소드를 시작할 때 쓰인다.

Abort-Nest-Transaction과 Abort-Sub-Transaction은 각각 Nest-Transaction과 Sub- Transaction을 재수행하기 위해서 사용된다. Abort-Sub-Transaction과는 달리 Abort-Nest-Transaction은 Nest-Transaction 전체를 다시 실행시켜야 한다. 단순 트랜젝션과 마찬가지로, Commit-Sub-Transaction, Commit-Nest-Transaction, Commit-Split-Transaction은 각각 Sub-Transaction, Nest-Transaction, Split-Transaction의 종료를 나타낸다. Sub-Transaction과 Nest-Transaction은 4.2절에서 논의한 바와 같이 각기 트랜젝션에서 시작한 종료 트랜젝션이 모두 종료되어야 비로소 종료될 수 있다. Commit-Split- Transaction은 지금 진행중인 트랜젝션 T를 T_A와 T_B로 나눌 수 있고, 특히 T_A를 종료시킨 후 T_B를 수행시키기 위해서 사용된다. 물론 이 경우, 앞서 논의한 1), 2), 3) 세 조건을 만족해야 된다.

Accept-Join-Transaction은 Join-Transaction을 시작하기 전에, 합쳐지는 트랜젝션의 승인을 얻기 위해서 사용된다. 즉, 두 사용자의 승인이 없으면, Join-Transaction을 수행할 수 없다 [13].

Resume-Transaction은 Split-Transaction 또는 Commit-Split-Transaction으로 분리된 트랜젝션의 시작을 위한 사용자 명령어이다. 또한, Suspend-Transaction은 트랜젝션의 수행을 일시정지 시키는데 사용된다 [13].

마지막으로, Transaction-Priority는 트랜젝션의 실시간 우선 순위를 제공하기 위해 사용된다. 이 우선 순위는 응용 분야 또는 사용자의 판단에 따라 다양하게 선택될 수 있다.

4.4. 트랜젝션 처리 기법

본 절에서는 본 논문에서 제안한 트랜젝션 모형을 기초로, 효과적인 트랜젝션 처리를 위한 기법들을 제안한다.

앞에서 논의한 바와 같이, 긴 트랜젝션은 재수행과 자원의 관리 측면에 부정적인 영향을 미친다. 따라서, 사용자가 긴 트랜젝션을 분리시켜 작은 단위로 만들 필요가 있다. 하지만, 너무 작은 단위의 트랜젝션으로 만들 경우, 이에 따른 처리 비용의 증가로 역시 전체적인 성능에 나쁜 영향을 줄 수 있다. 가상대학에서는 교육용 소프트웨어에 학습자가 실시간으로 온라인으로 반응하는 것이 일반적인 추세로 진행될 것으로 기대된다. 따라서, 사용자가 트랜젝션을 진행시킨 후, 필연적으로 생각시간 (User Thinking Time)을 필요로 한다. 이와 같은 사용자의 생각 시간 동안에는 어떤 의미 있는 작업이 진행 중이 아니

기 때문에, 생각 시간이 시작되는 경우 트랜잭션을 분리시키는 것이 전체적인 성능의 향상을 기대할 수 있다.

실시간 처리를 위해서 각각의 트랜잭션, 즉 글로벌 트랜잭션, Split/Join 트랜잭션, 중첩 트랜잭션, 또한 종속 트랜잭션은 각기 처리 마감시간 (Deadline), 또는 트랜잭션의 남은 처리시간 (Remaining Service Time), 대기 여유시간 (Slack Time) 또한 응급성 (Criticalness) 에 따라 각기 사용자가 우선 순위를 부여한다. 실시간 우선 순위도 트랜잭션이 변형되고 또한 중첩 트랜잭션의 경우 종속 트랜잭션이 종료되면 달라질 수 있다고 가정한다. 이 실시간 우선순위 할당 기법은 본 논문에서는 다루지 않는다.

5. 결론 및 연구과제

본 논문에서는 가상대학의 구현을 위한 학습 데이터 베이스를 접근하여 트랜잭션을 처리를 위한 관점에서 필요한 일반적인 조건들과 특성들을 논의하고 이를 기반으로 가상대학을 지원을 위한 트랜잭션의 요구사항을 제시하였다. 또한, 이 요구사항들을 만족시킬 수 있는 트랜잭션 모형을 제시하였고, 이 모형을 기반으로 트랜잭션 처리 기법을 제시하였다. 본 트랜잭션 모형은 직렬성을 만족하고, 응용분야에 따른 유연성을 제공하기 위해 원자성과 고립성을 만족시키지 않을 수 있도록 했다.

앞으로 본격적으로 운영될 가상대학을 위해본 연구는 다음과 같은 연구가 필요하다. 본 트랜잭션 모형은 트랜잭션의 오랜 지속성, 비예측성, 상호 작용성, 내부 병렬성을 제공한다. 본 모형의 상호 작용성과 병렬성은 제한된 범위에서 이루어지고 있으나, 앞으로는 가상대학의 구체적인 학습요소의 필요에 따라 더 확대될 필요성이 있으며, 또한 현재 널리 쓰이는 정확도 기준인 직렬화 대신에 더 높은 동시성 (Concurrency) 을 제공할 수 있는 기준이 제공되어야 한다.

또한, 본 모형을 기초로, 실시간 처리를 위해 각 트랜잭션의 우선순위 할당 기법과 처리 기법이 개발되어야 한다. 특히, 실시간 처리에서는 전체적인 트랜잭션의 반응시간보다 처리마감시간 실패율 (Missing Ratio)가 트랜잭션 성능의 평가 척도가 되고 또한 높은 우선순위의 트랜잭션이 낮은 우선순위를 가진 트랜잭션에 의해 대기되는 상황 (Priority Inversion) [16,17,18] 을 해결하는 기법이 개발되어야 한다.

마지막으로, 본 모형의 기반을 둔 트랜잭션 처리를 위한 동시성 제어 (Concurrency Control) 기법, 회복 (Recovery) 기법 등이 보완되어야 한다. 동시성 제어 기법 중 낙관적 방법 (Optimistic Scheme)은 동적으로 변화는 트랜잭션에 부정적인 것으로 알려져 [13]. 비관적 방법 (Pessimistic Scheme) 중 특히 로킹기법 (Locking Scheme)에 대한 연구가 필요하리라 기대된다.

참고 문헌

[1] 황대준, (1998), "가상대학의 현황과 발전방향", 한국정보과학회지, 제 16권 10호, Oct., 6- 15.

[2] 김성일, (1998), "가상대학의 당면과제와 운영방안", 한국정보과학회지, 제 16권 10호, Oct., 16-25.

[3] Jasmine Manual CD Developer Edition, Computer Associates, (1998).

[4] Korth, H. F. and Silberschartz, A., (1991), Database System Concepts, 2nd Edition, McGraw Hill, Singapore.

[5] Muth, P., et al, (1992), "A Transaction Model for an Open Publication Environment", in Chapter 6, Database Transaction Models for Advanced Applications, edited by A. K. Elmagarmid, Morgan-Kaufmann Publishers, San Mateo, California.

[6] Moss, J. E. B., (1985), Nested Transactions An Approach to Reliable Distributed Computing, MIT Press, Cambridge, Mass..

[7] 권택근, (1996), "주문형 비디오 서비스를 위한 멀티미디어 서버", 데이터 베이스 월드, 1996년 6

월호, 19 22.

[8] 박정선, (1996), "원격 데이터 베이스 접근", 데이터 베이스 월드, 1996년 7월, 30 34.

[9] Kim, W., (1990), Introduction to Object-Oriented Databases, The MIT Press.

[10] Finkelstein, R., (1998), "The Jasmine Object-Oriented Database Solution for Online Interactive Transaction Processing (OLIP) and Electronic Commerce, Jasmine White paper.

[11] Bernstein, P. A., Hadzilacos, V. and Goodman, N., (1987), Concurrency Control and Recovery in Database Systems, Addison-Wesley.

[12] Eswaran, K. P., et al, (1976), "The Notions of Consistency and Predicate Locks in a Database System", Communications of the ACM, Vol. 19, No. 11, Nov., 255-265.

[13] Kaiser, G. E. and Pu, C., (1992), "Dynamic Restructuring of Transactions", in Chapter 8, Database Transaction Models for Advanced Applications, edited by A. K. Elmagarmid, Morgan-Kaufmann Publishers, San Mateo, California.

[14] Pu, C., Kaiser, G. E., and Hutchinson, N., (1988), "Split-transaction for Open-ended Activities", Proceedings of the 14th Very Large Data Bases Conference, Los Angeles, California, USA, pp. 26- 37.

[15] Muth, P. E., Rakow. T. C., Klas, W., Neuhold, E. J., (1992), "A Transaction Model for an Open Publication Environment", in Chapter 6, Database Transaction Models for Advanced Applications, edited by A. K. Elmagarmid, Morgan-Kaufmann Publishers, San Mateo, California.

[16] Abbott, R. and Garcia-Molina, H., (1988), "Scheduling Real-time Transactions: A Performance Evaluation", Proceedings of the 14th Very Large Data Bases Conference, Los Angeles, California, 1 12.

[17] Huang, J., et al, (1991), "On Using Priority Inheritance in Real-time Databases", 12th Real-time Systems Symposium, 210 221.

[18] Ulusoy, O. and Belford, G. G., (1993), "Real-Time Transaction Scheduling in Database Systems", Journal of Information Systems, Vol. 18, No. 8, 559 580.