분산트랜잭션 처리

박정선
명지대학교 산업공학과 교수

분산 시스템에서 하나의 서버에 속한 데이터 아이템들은 여러 서버들로 분산될 수 있으며, 클라이언트에서 발생되는 트랜잭션은 여러 서버들을 참여시킬 수 있다. 어떠한 경우에는 클라이언트 트랜잭션에 의해 접근되는 서버가 또 다른 서버를 접근하는 경우도 발생할 수 있다. 이와같이 여러 서버들을 직·간접적으로 접근하는 트랜잭션을 분산 트랜잭션이라 하고 그러한 트랜잭션 처리를 분산 트랜잭션 처리(Distributed Transaction Processing, DTP)라 한다.

이러한 특성을 지닌 분산 트랜잭션의 원자성을 보장하기 위해서는 그 트랜잭션에 참여하는 모든 서버들이 다 같이 완료 동작을 수행하거나, 취소 동작을 수행해야 한다. 이와 같은 동작을 위해서는 트랜잭션에 참여하는 서버들중 하나가 코디네이터가 되어 다른 서버들과의 통신을 통해 합의를 거친후, 공통된 행동을 취하는 "2-phase-commitment protocol"을 사용한다.

분산 트랜잭션 처리에서 동시성 제어(Concurrency Control) 기법은 기존의 방법과 같은 2-phase 록킹, 타임 스탬프 등을 사용한다. 하지만, 회복 기법은 비록 로깅 동작을 하지만 중앙 집중 시스템의 파손과 분산 환경하에서의 파손, 이 모두를 다루어야 하기 때문에 훨씬 다양하고 복잡한 회복기법을 사용해야만 한다.

여기에서는 하나 이상의 여러 데이터베이스에 대하여 수행되는 분산 트랜잭션 처리를 지원하기 위한 표준을 중심으로 살펴보기로 한다. ISO/IEC JTC1/SC21/WG6에서는 OSI-DTP 모델, 서비스, 프로토콜 등을 표준화하였다. ISO/IEC OSI-DTP는 OSI 참조모형의 제 7 계층에 속하는 표준 집합들 중의 하나로서 분산 트랜잭션을 지원하기 위한 통신 기능들을 제공하며, 분리되어 있는 개방형 시스템들 상에 존재하는 다수의 OSI 트랜잭션 자원들에 걸쳐 상호 호환을 가능하도록 하는 기본틀을 제공한다. 여기서 트랜잭션이라 함은 원자성, 일치성, 고립성, 영속성의 4가지 성질(ACID)을 만족하는 연산들의 논리적인 한 묶음을 나타낸다. 특히 하나 이상의 데이터베이스를 접근하는 분산 트랜잭션의 ACID 성질을 만족시키는 것은 복잡한 메카니즘이 필요하다. ISO/IEC 10026 DTP 표준에서는 이를 위한 기능들을 통신상의 관점에서 명시하고 있다. 현재 OSI-DTP는 모델, 서비스, 프로토콜이 국제표준(IS)으로 제정되어 있고, 프로토콜 구현을 위한 기능표준이 국제기능표준(ISP) 단계로 추진 중에 있다. 한가지 명심할 점은 DTP 표준은 통신 표준이라는 점이다. DTP 표준을 완전히 따른다고 해서, DTP에 참여하는 데이터 자원들의 기능에는 무관하게 분산 트랜잭션이 지원된다고 생각하면 안된다. 즉, 완벽한 분산 전역 트랜잭션(Global Transaction)을 수행하기 위해서는 여러 데이터베이스들 간의 통신 문제도 해결되어야 하고, 각 데이터 자원의 고유한 기능(예

를 들면, 각 DBMS의 동시성 제어, 회복, 완료능력 및 이들의 통합)들도 함께 연계되어 해결되어야만 한다. 이러한 기능들은 현재 학계에서 연구중이며 아직 실험실 단계의 구현 수준에 머물고 있다.

〈그림 1〉 제 7 계층 서비스들의 관계

RPC	RDA
DTP	
ACSE	CCR

DTP 규약은 완료, 동시성, 회복을 위한 CCR(Commitment, Concurrency and Recovery) 규약과 접속 관리를 위한 ACSE(Association Control Service Element) 규약을 이용한다. 따라서 DTP 규약은 CCR, ACSE 규약의 중재자처럼 작동하며 복수개의 응용 접속상에서의 여러 쌍의 통신 환경에서 작동한다. 반면에 CCR, ACSE 규약은 1:1 환경에서 작동한다. DTP 규약은 데이터 전송의 어떠한 형태를 명시하지는 않고, 다만 ASE(응용 서비스 요소)가 데이터를 전송할 수 있는 통신 환경을 제공한다.

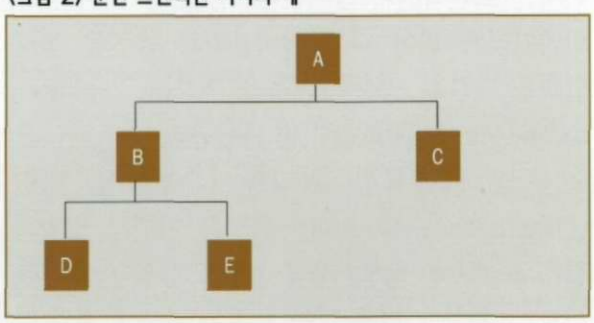
따라서 데이터 전송을 위한 사용자 ASE는 DTP 규약을 사용하게 될 응용이 결정하여야 한다. 이런 응용의 대표적인 것이 RPC와 RDA이다. 위에서 언급한 규약들은 모두 제 7 계층에 속하지만, 제 7 계층 내에서 그림 1과 같은 계층적 구조를 이루고 있다고 이해할 수 있다. 단,

이 그림에서 RDA를 위하여 DTP가 반드시 필요한 하부 계층이고, DTP를 위하여 CCR이 반드시 필요한 계층이라는 뜻은 아니다. 다만 각 서비스 간의 관계를 쉽게 설명하기 위한 그림이다. RDA가 DTP를 이용하게 되는 경우가 바로 RDA TP 응용 문맥이다.

DTP 표준은 다음과 같은 상황으로 쉽게 이해할 수 있다. 시스템 A로부터 시스템 B와 C에 대한 트랜잭션이 제출되고, 이를 받은 시스템 B는 다시 시스템 D, E에 추가 트랜잭션을 제출하여 전체적으로 하나의 전역 트랜잭션(Global Transaction)을 형성한다고 가정하자(그림 2). 이러한 전역 트랜잭션을 수행하기 위해서 앞서 설명한 RDA 기본 응용 문맥을 사용할 수 없다. 왜냐하면 RDA 기본 응용 문맥은 1단계 완료 규약만을 지원하고 각 사이트의 1:1의 관계만을 지원하므로, 트랜잭션의 ACID 성질을 만족시킬 수 없기 때문이다.

〈그림 2〉의 예에서 분산 트랜잭션 처리를 위하여 참가하는 시스템들은 각기 다른 종류의 DBMS를 사용할 수도 있다. 따라서 각 사이트 내에서 처리되는 부트랜잭션(Subtransaction)들은 각기 고유한 기능에 의하여 실행된다. 그러므로 상호간의 데이터베이스 오퍼레이션에 대한 완료/철회 등에 공통된 규약이 필요하다. 또한 시스템간에 통일된 통신 규약이 필요하며, 각 시스템이 현재 실행 중인 트랜잭션에 대하여 동일하게 이해하고 있어야 한다. 이러한 통신 서비스를 제공하는 것이 ISO/IEC의 DTP이다.

〈그림 2〉 분산 트랜잭션 처리의 예



DTP 서비스는 다음의 7가지 기능단위로 나뉘어져 있다.

(1) 대화 관리 기능단위

- TP-BEGIN-DIALOGUE : 대화를 설정함.
- TP-END-DIALOGUE : 대화를 종료함.
- TP-U-ERROR : TP 사용자에 의한 오류 호출
- TP-U-ABORT : TP 사용자에 의한 대화 비정상 종료 호출
- TP-P-ABORT : TP 제공자에 의한 대화 비정상 종료 호출

(2) 공유 제어 기능단위

이 기능단위에 속하는 서비스 요소는 없다.

(3) 편극화 제어(Polarized Control) 기능단위

- TP-REQUEST-CONTROL : 대화 제어권을 요청함.
- TP-GRANT-CONTROL : 대화 제어권의 요청을 허락함.

(4) 상호통신(Handshake) 기능단위

- TP-HANDSHAKE : 두 시스템간의 처리\를 동기화시킴. 지정 시점까지의 작업 처리를 일치시킨다.
- TP-HANDSHAKE-AND-GRANT-CONTROL : 처리를 동기화하고 제어권을 허락함.

(5) 완료 기능단위

- TP-DEFERRED-END-DIALOGUE : 현 트랜잭션이 완료됨과 동시에 대화의 종료를 알림.
- TP-DEFERRED-GRANT-CONTROL : 현 트랜잭션이 완료됨과 동시에 제어권을 넘겨줌.
- TP-COMMIT : 1단계 TP-PREPARE 대신 사용할 수 있다. 보통의 경우는 2단계가 끝난 후 트랜잭션의 완료를 요청함.
- TP-COMMIT-RESULT : 2단계 완료 규약 2단계에서 사용. TP-COMMIT 명령을 받는 쪽에서는 TP-COMMIT-RESULT 서비스로 받게 된다.
- TP-DONE : 2단계 완료 규약 2단계에서 사용. 완료/철회의 종료를 선언한다.
- TP-COMMIT-COMPLETE : 2단계 완료 규약 2단계에서 사용. 트랜잭션의 완료가 성공적으로 끝났음을 알림.
- TP-PREPARE : 2단계 완료 규약 1단계에서 사용. 상대방에게 트랜잭션의 종료 준비상태로 들어 가라고 요청함.
- TP-READY : 2단계 완료 규약 1단계에서 사용. TP-PREPARE에 대한 성공적인 응답.
- TP-ROLLBACK : 2단계 완료 규약 1, 2단계에서 사용. 1단계에서 TP-PREPARE에 대하여 완료하지 못할 경우 철회를 요청함.
- TP-ROLLBACK-COMPLETE : 2단계 완료 규약 2단계에서 사용. 트랜잭션의 철회가 성공적으로 끝났음을 알림.
- TP-HEURISTIC-REPORT : 2단계 완료 규약 2단계에서 사용.

(6) 연결 트랜잭션(Chained Transactions) 기능단위

이 기능단위에 속하는 서비스 요소는 없다.

(7) 비연결 트랜잭션(Unchained Transactions) 기능단위

- TP-BEGIN-TRANSACTION : 비연결 트랜잭션의 시작을 위하여 사용한다.

DTP 표준은 완료 규약에 대하여 두가지 형태를 사용할 수 있게 하고, 각 경우에 따라 서비스 호출의 순서가 다르도록 하였다.

하나는 응용 제공(Application Supported)

트랜잭션이고 다른 하나는 서비스 제공(Providor Supported) 트랜잭션이다. 응용 제공 트랜잭션일 경우는 DTP 하부의 CCR 서비스를 이용하지 않고, 2단계 완료에 필요한 통신상의 로깅, 회복 등의 기능을 응용 프로그램이 전담하게 된다. 즉, 통신상의 2단계 완료 규약의 완벽한 작동은 DTP 규약을 이용하는 응용 프로그램에 달려있다. 서비스 제공 트랜잭션의 경우는 DTP가 CCR을 이용하여 2단계 완료 규약을 제공하는 경우이다. 따라서 이때 응용 프로그램은 DTP 서비스 호출만으로 CCR이 제공하는 통신상의 완료, 회복 기능을 사용할 수 있다.

분산 트랜잭션 처리를 위하여 가장 중요한 기능 중의 하나는, 각 시스템간의 완료/철회를 서로 보장할 수 있는 2단계 완료 규약 메카니즘이다. 하지만 DTP, CCR 규약은 시스템간의 통신상에서 일어나는 2단계 완료만을 정의하고 있다. 그러므로 만약 어느 사이트의 DBMS가 준비상태가 지원되지 않는 데이터 자원이라면, 분산 트랜잭션 처리를 위하여 DBMS에 준비상태를 위한 부가적인 기능들을 추가하여야만 한다. 물론 DBMS가 준비상태를 지원할 수 있는 데이터 자원이라면 2단계 완료 규약 측면에서는 DTP에 바로 접속이 가능하다고 하겠다.

DTP 서비스는 사용자의 응용 프로그램에서 직접 호출하여 사용할 수도 있고, 앞서 언급한 바와 같이 RDA 등이 이를 호출하여 사용할 수도 있다. RDA가 DTP를 이용할 때, 이를 RDA TP 응용 문맥이라고 부른다. RDA TP 응용 문맥일 경우에는 사용자의 응용 클라이언트 프로세스는 둘 이상의 여러 서버에 동시에 접근이 가능하며, DTP에 의한 분산 트랜잭션의 처리가 가능하다. RDA TP 응용 문맥에서 응용 프로그램은 RDA 서비스 호출을 주로 사용하고 트랜잭션 관련 호출만 DTP 서비스 호출을 사용하지만, 사용자의 트랜잭션 관련 요청을 실제로 처리하는 메카니즘은 거의 다 DTP 서비스에서 수행된다. 따라서 여러 서버에 대한 동시 접근이 가능해 진다.

RDA TP 응용 문맥을 구현하기 위해서 별도의 표준이 존재하는 것은 아니다. DTP 응용 서비스 요소와 RDA 응용 서비스 요소가 갖추어진 상태에서, 이 둘을 연결해서 사용하는 것이라고 생각하면 된다. DTP만을 단독으로 이용하는 경우, DTP는 데이터베이스 언어를 송/수신하는 서비스가 없으므로 응용 프로그램이 이러한 기능을 수행해야 한다. 반면에 RDA TP 응용 문맥은 데이터베이스 언어의 송/수신은 RDA를, 트랜잭션 관리는 DTP를 이용하는 것이라고 이해할 수 있다. 즉, RDA TP 응용 문맥은 통신 기능보다는 데이터베이스 사용에 더욱 친숙한 사용자들을 위하여, DTP를 쉽게 이용할 수 있도록 RDA에 DTP를 포함시킨 것이라고 이해하면 될 것이다.

여기에서는 여러 시스템에 분산되어 있는 데이터베이스들이 개방형 통신을 이용하여 분산 트랜잭션을 처리할 수 있는 통신 표준으로 현재 제시되고 있는 DTP 표준에 대하여 알아보았다. 결국 이러한 분산 트랜잭션은 여러 데이터베이스를 동시에 한꺼번에 접근하기 때문에 발생하고, 이는 궁극적으로 여러 데이터베이스를 하나의 데이터베이스처럼 볼 수 있는 다중 데이터베이스 시스템의 기초적인 모형이라고 이해할 수도 있다.

다중 데이터베이스 시스템은 하나의 다중 데이터베이스 관리 시스템(Multidatabase Management System; MDBMS)과 지역 데이터베이스 시스템(Local Database System; LDBS)들의 집합으로 구성된다. MDBMS는 여러 LDBS를 접근하는 트랜잭션을 수행하는 응용 프로그램(사용자)을 위한 인터페이스(Application Program Interface; API)를 제공한다. 또한 각각의 LDBS는 지역 데이터베이스 관리 시스템(Local Database Management

System; LDBMS)과 지역 데이터베이스(Local Database; LDB)로 구성된다. LDBMS는 자신이 관리하는 LDB에 대한 트랜잭션들을 수행한다.

〈그림 3〉에서 보는 바와 같이 MDBS에는 두 종류의 트랜잭션이 있다. 하나는 MDBMS에서 수행되도록 요구하는 전역 트랜잭션(Global Transaction)이고, 다른 하나는 LDBMS에서 독자적으로 LDBS 응용이 요구하는 지역 트랜잭션(Local Transaction)이다. 전역 트랜잭션은 2개 이상의 여러 지역 데이터베이스를 동시에 접근한다. MDBMS는 전역 트랜잭션을 하나의 LDB를 접근하는 여러개의 전역 부트랜잭션(Global Subtransaction)으로 나눈다. 따라서 어떤 LDBS 내에서 수행되는 트랜잭션은 LDBS 응용이 수행을 요청한 지역 트랜잭션이거나, MDBMS로부터 제출된 전역 부트랜잭션이다.

MDBMS는 LDBS의 일반 응용처럼 LDBS들의 상위에 구축되며, 다중 데이터베이스 커널(Mulitidatabase Kernel; MDBK)과 각각의 LDBS를 위한 드라이버(Driver)로 구성된다. MDBK의 전역 데이터 관리자(Global Data Manager; GDM)는 각 지역 데이터베이스 스키마를 통합한 다중 데이터베이스 스키마를 관리하며, 전역 트랜잭션을 해당 LDBS들에서 수행할 수 있도록 전역 부트랜잭션으로 나눈다. 하나의 전역 부트랜잭션이 접근하는 모든 데이터는 한 LDBS에 저장된 것이다.

즉 MDBK는 다중 데이터베이스 스키마를 제외한 다른 어떠한 데이터도 직접 관리하지 않는다. 따라서 LDBS는 전역 부트랜잭션을 자신이 수행하는 지역 트랜잭션처럼 취급한다. 전역 트랜잭션 관리자(Global Transaction Manager; GTM)의 기능은 크게 다음 2가지로 나눌 수 있다. 첫째는 각 LDBS의 전역 부트랜잭션의 수행을 제어함으로써 전역 트랜잭션과 지역 트랜잭션 수행의 직렬화를 보장하는 동시성 제어이다. 둘째는 고장에 대비할 수 있도록 전역 트랜잭션의 원자성과 지속성을 유지하도록 하는 완료와 회복이다.

드라이버는 MDBMS의 일부분으로 각 LDBS상에서 운영된다. 이는 GTM으로부터 전역 부트랜잭션의 연산을 받아, 그것을 LDBMS에서 수행하여 그 결과를 다시 GTM에게 보낸다.

이 드라이버는 ODBC나 IDAPI에서와 마찬가지로 필요할 때 명령어의 번역 등을 수행한다. LDBMS를 이 같은 관점에서 볼 때 드라이버는 단지 LDBS의 응용에 불과하며, 전체 MDBMS도 LDBS의 응용으로 볼 수 있다. 그리고 LDBS의 상부에 있는 서버는 드라이버와 통신을 담당하며 LDBS에서 수행하여야 할 로깅, 2단계 완료 규약의 준비상태 등을 지원하는 모듈로 생각할 수 있다. DC

〈그림 3〉 다중 데이터베이스의 개념적 구조