E504 EJB 컨테이너 시스템의 데이터베이스 커넥션 관리 방법

서범수°, 김성훈, 장철수, 김중배
한국전자통신연구원 전자거래연구부
(bsseo°,saint, jangcs, jjkim)@etri.re.kr

The Database Connection Management Architecture for E504 EJB Container System

Beom-Su Seo°, Sung-Hoon Kim, Choul-Soo Jang, Joong-Bae Kim
Electronic Commerce Department
Electronics and Telecommunications Research Institute

요 약

EJB(Enterprise Java Bean)에서는 데이터베이스나 JMS(Java Message Service), 메일, ERP와 같은 외부 시스템 자원을 사용하기 위해 EJB 스펙은 JCA(Java Connector Architecture) 사용을 권장하고 있다. 본 논문에서는 E504(Enterprise 504) EJB 컨테이너가 사용하는 여러 가지 자원 중 데이터베이스 커넥션에 초점을 맞추어, EJB 스펙에서 논의하고 있는 리소스 관리 요구 사항과 JCA를 이용한 데이터베이스 커넥션 관리 방법 및 빈에서 발생 가능한 커넥션 요청 모델에 대해 논의한다.

1. 서 론

EJB 컨테이너는 빈에서 데이터베이스에 대한 접근을 위해 데이터베이스 커넥션을 제공하여야 한다. 빈에서는 자신이 사용할 데이터베이스에 대한 커넥션을 제공할 팩토리 객체(javax.sql.DataSource)를 JNDI로부터 lookup하여 해당 팩토리 객체를 통해 java.sql.Connection 객체를 얻어 데이터베이스를 사용한다. 데이터베이스에 대한 커넥션 요청은 트랜잭션의 연관 유무와 공유 모드 지정 유무 등에 따라 다양한 시나리오가 존재한다. 본 논문에서는 EJB 컨테이너의 리소스 관리 요구 사항을 논의하고 이를 구현한 데이터베이스 커넥션 관리자를 소개한다. 또한 커넥션 요청 시나리오에 따라 커넥션 관리자의 기능을 살펴본다.

2. 리소스 관리의 요구 사항

EJB 컨테이너는 다음과 같은 사항을 고려하여 리소스를 관리할 수 있어야 한다.

첫째, EJB 스펙에서는 데이터베이스뿐 아니라 JMS, ERP 및 기타 레거시 시스템도 리소스로 간주한다. 이러한 다양한 리소스를 EJB 컨테이너가 수용하고 효과적으로 관리하기 위해 EJB 스펙에서는 JCA를 제안하고 있다. JCA를 지원하는 리소스들은 컨테이너 코딩의 수정이나 어떤 로직의 변경없이 EJB 컨테이너와 바로 연동되어 사용될 수 있다.

둘째, EJB 컨테이너는 다양한 형태의 리소스를 하나의 트랜잭션에서 사용할 경우 이들 리소스들에 대한 일관성을 보장해 주어야 한다. 현재 E504 컨테이너에서는 다중 데이터베이스에 대한 트랜잭션만을 고려하여 구현하였지만 XAResource를 지원하는 리소스라면 어떤 유형의 리소스도 함께 사용될 수 있다.

셋째, 배포된 빈들이 동일 트랜잭션에서 하나의 데이터베이스만을 접근할 경우 XAResource를 등록하고 2-phase commit 프로토콜을 수행하는 오버헤드를 감소시키기 위해 로컬 트랜잭션을 적용하여야 한다. 로컬 트랜잭션은 트랜잭션매니저가 생성한 트랜잭션이 아닌 리소스와 연관되어 내부적으로 사용되는 트랜잭션을 의미한다.

넷째, EJB 스펙은 특정 빈이 동일 트랜잭션내에서 동일 데이터베이스를 사용할 경우 물리적 커넥션을 공유함으로써 처리 속도와 커넥션 자원을 절약할 것을 권장한다. 빈에서 데이터베이스를 사용하는 것은 트랜잭션과 밀접한 관계를 가진다. 트랜잭션이 없는 경우도 존재할 수 있으며 트랜잭션이 존재하는 경우 동일 데이터베이스를 공유 혹은 비공유 모드로 접근할 수도 있다. 또한 동일 트랜잭션에서 상이한 데이터베이스를 공유 혹은 비공유 모드로 접근하는 등의 다양한 경우가 발생한다. 그리고 트랜잭션이 시작되기 전에 데이터베이스 커넥션을 할당받고 이후 트랜잭션을 시작하여 커넥션을 사용할 수도 있다. 이 경우 트랜잭션 시작 전에 할당된 커넥션을 트랜잭션과 연관시켜 관리할 필요가 있다.

3. JCA를 적용한 E504의 커넥션 관리

JCA는 다양한 형태의 데이터 리소스들을 하나의 EJB 컨테이너에서 사용하기 위해 SUN에서 제안한 리소스 연동 방안이다. JCA는 EAI를 위한 표준화된 API 셋이라 할 수 있다. SOAP을 이용한 Web Service와는 달리 JCA는 EJB 컨테이너의 커넥션 매니저가 구현해야 하는 모듈과 연동될 데이터 소스들이 제공해야 할 어댑터를 위한 표준 인터페이스를 정의한다. 따라서 EJB 서버와 연동될 각각의 리소스 벤더들은 JCA 어댑터를 구현해야 한다. 그림 3과 같이 JCA를 적용하지 않는 경우 EJB 컨테이너는 각각의 리소스마다 해당 리소스를 위한 어댑터를 구현하여야만 한다. 이 경우 N개의 EJB 서버가 M개의 리소스를 사용하려면, 각각의 EJB 컨테이너마다 리

소스를 관리하기 위한 방식 또한 상이하므로 N*M개의 시스템 구성이 이루어진다. 그러나 JCA를 사용할 경우 EJB 서버들은 JCA에서 제안하는 표준화된 커넥션 매니저를 구현하고 각각의 리소스 벤더들은 JCA 어댑터를 제공하므로 N+M개의 시스템 구성으로 줄어든다.

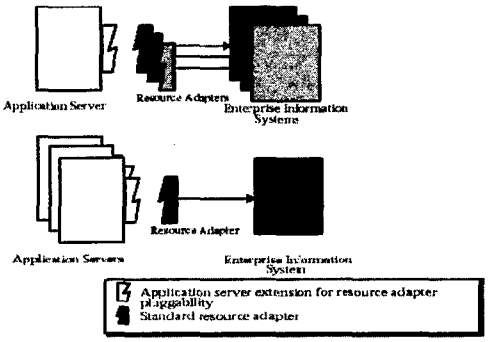

그림 1. JCA의 효과

구현 중인 E504 컨테이너 시스템은 이러한 JCA를 적용하여 커넥션 매니저를 구현하였으며 데이터베이스에 대해 JCA의 Resource Adaptor 및 Client Interface를 구현하였다. 그림 2는 JCA를 적용한 E504의 데이터베이스 커넥션 관리 구조를 나타낸다. 표준화된 인터페이스 역시 그림 2에 표시되어 있다. 빈이 ConnectionFacotry(javax.sql.DataSource)에 데이터베이스에 대한 커넥션을 요청하면 Connection Factory는 EJB 컨테이너에 존재하는 ConnectionManager에게 커넥션을 요청한다. ConnectionManager는 요청을 분석하여 어떤 데이터베이스 어댑터에 커넥션을 요청해야 하는지 판단하고 해당 데이터베이스를 물리적으로 접근하여 커넥션을 가져오는 객체인 ManagedConnecitionFacotry를 통해 ManagedConnection을 요청한다. ManagedConnectionFactory는 JDBC를 이용하여 특정 데이터베이스에 대한 물리적 커넥션, XAResource, XAConnection 등을 생성하고 이를 가지고 있는 ManagedConnection을 생성하여 ConnectionManager에게 전달한다. ConnectionManager는 전달받은 ManagedConnection을 트랜잭션 및 공유 및 비공유 여부에 따라 관리한다. 이때 동일 트랜잭션에서 서로 다른 데이터베이스를 접근한다면 생성된 ManagedConnection의 XAResource를 트랜잭션에 enlist 시킨다. 이 작업을 수행하는 모듈이 Transaction Enlist Process이다. 한번 생성되어 사용된 후 반환된 ManagedConnection은 Connection Pool에 보관되며, 현재 사용 중인 ManagedConnection은 Connection Cache에 저장된다. 여러 빈이 동일 트랜잭션에서 같은 데이터베이스를 접근할 경우 하나의 ManagedConnection만을 사용하도록 빈 배포자가 명시할 경우 이와 같은 공유 작업을 처리하기 위한 모듈이 Connection Sharing Process이다. 그리고, EJB 컨테이너나 빈이 트랜잭션을 커밋하거나 롤백하면 ManagedConnection은 자신을 관리하는 커넥션 매니저에게 커넥션의 커밋이나 롤백을 알리기 위해 ConnectionEventListener를 사용한다. ManagedConnection은 빈이 접근하는 논리적 커넥션(java.sql.Connection)을 생성하여 전달한다. 이러한 논리적 커넥션들은 물리적으로 동일 커넥션을 공유한다.

ManagedConnectionFactory와 ManagedConnection, 물리적 커넥션과의 관계는 4절의 Connetion 요청 시나리오와 다시 논의한다

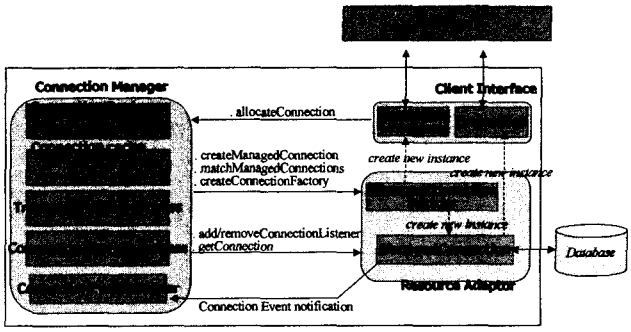

그림 2. JCA를 이용한 커넥션 관리 구조

4. Connection 요청 모델

빈에서 요청하는 커넥션은 트랜잭션 유무와 커넥션의 공유 모드 설정, 커넥션 요청 시점과 트랜잭션의 구동 시점 등에 따라 그림 3과 4와 같이 다양한 형태의 요청 모델이 존재한다. 그림 3에서 보는 바와 같이 하나의 ConnectionManager는 여러 개의 ManagedConnectionFactory를 가질 수 있다. 즉, 서로 다른 벤더가 제공하는 데이터베이스를 연동하기 위해 해당 벤더는 Resource Adaptor로써 이러한 ManagedConnectionFactory를 제공해야 한다. 개발 중인 E504 컨테이너에서는 JCA를 지원하지 않는 데이터베이스를 지원하기 위해 JDBC를 이용하여 Oracle, MS-SQL 등으로부터 커넥션을 얻어 온다. 한 트랜잭션에서 빈이 커넥션을 요청할 때 이전에 접근하지 않은 상이한 데이터베이스를 미래에 접근할 것인지를 알 수 없기 때문에, ManagedConnectionFactory는 각각의 데이터베이스가 제공하는 XADataSource로부터 XAConnection을 생성하고 이것으로부터 물리적 커넥션을 생성하여 XAResource를 생성한다. ManagedConnection은 이러한 객체들을 담고 있으며 빈이 SQL을 실행할 때 필요한 논리적 커넥션 객체인 Connection(java.sql.Connection의 구현 객체) 객체를 생성하여 빈에게 전달한다. 이 논리적 커넥션은 트랜잭션에 대한 정보, 물리적 커넥션, ManagedConnection에 대한 정보 등을 담고 있어 빈의 메소드 요청을 적절히 처리한다.

커넥션 요청 모델은 그림 3과 4와 같이 6가지 경우로 구분된다. 그림 2에서 논의한 ConnectionManager의 각 모듈 및 프로세스들은 다음의 각 상황을 처리한다.

(1)은 BMT(Bean Managed Transaction)로써 트랜잭션 시작 전에 커넥션을 요청한 후 한번도 사용하지 않는 경우이다. 이러한 경우는 빈 개발자의 코딩 실수로 인해 발생할 것이지만 시스템은 이렇게 할당된 커넥션을 해당 메소드를 호출한 쓰레드가 종료되면 커넥션 풀에 넣어 재사용할 수 있도록 지원해야 한다.

(2)는 CMT로써 빈 A가 빈 B의 메소드를 호출하여 새로운 트랜잭션이 시작되었으며 해당 트랜잭션이 빈 C에까지 전달되고 있다. (2)에서 요청한 DataSource(ConnecionFactory)는 공유 모드로 정의되어 있으므로 동일 트랜잭션내에서 해당 데이터소스에 아무리 많은 커넥션을 요청하더라도 실제 물리적 커넥션은 하나만 존재하며 물리적 커넥션을 공유하는 논리적 커넥션이 생성되어 빈에게 반환된다. 트랜잭션이 없이 커넥션이 요청되었다면 커넥션 매니저는 트랜잭션이 없는

ManagedConnection을 따로 관리하게 되며 공유모드로 선언되었다고 하더라도 물리적으로 다른 커넥션을 가진 ManagedConnection들이 생성되어 커넥션 매니저에 의해 관리된다.

(3)은 생성된 트랜잭션을 사용하므로 (2)에서 얻은 ManagedConnection과 (3)에서 얻은 ManagedConnection은 트랜잭션을 기준으로 동일 자료 구조에 저장된다. 접근하는 데이터소스는 다르지만 동일 데이터베이스를 접근하고 있으므로 전역 트랜잭션(Global Transaction : 트랜잭션 매니저가 생성한 트랜잭션)에 XAResource를 등록할 필요가 없으며 2PC 프로토콜을 수행할 필요가 없다. 이 경우 트랜잭션이 커밋되면 할당된 ManagedConnection과 연관된 로컬 트랜잭션들이 물리적 커넥션에 대한 커밋이나 롤백을 수행한다.

(4)는 비공유 모드로 커넥션을 요청하는 경우이며 트랜잭션이 없이 커넥션을 할당하는 경우와 동일하게 물리적 커넥션마다 하나의 ManagedConnection이 존재하며, 이 ManagedConnection에서 생성되는 논리 커넥션 역시 하나만 존재한다. 리소스 사용면에서 비효율적인 구조이나 데이터베이스를 독립적 커넥션 접근할 필요가 있는 응용을 위해 EJB 스펙에서 제안하고 있다

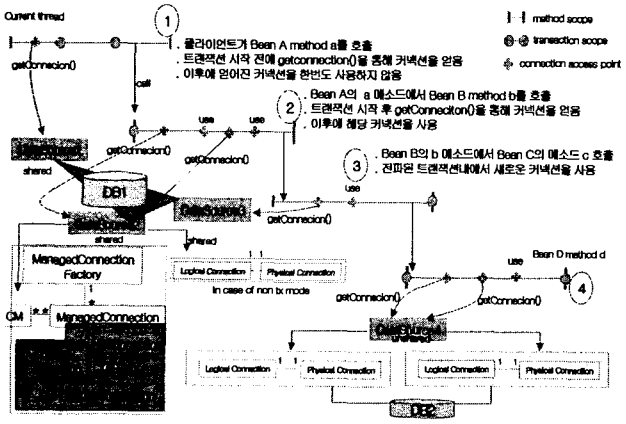

그림 3. 커넥션 요청 모델 I

(5)는 트랜잭션이 없이 커넥션을 요청한 후 이를 사용하고 명시적으로 빈에서 commit()을 수행한 경우이다. 전역 트랜잭션과 전혀 관계가 없기 때문에 이러한 사용자가 명시적으로 commit()을 수행하지 않고 해당 메소드를 종료한다면 커넥션 매니저는 수행된 모든 작업들을 롤백시키고 할당된 ManagedConnection을 풀에 넣어 재사용하도록 해야 한다. 사용자가 명시적으로 commit()/rollback()을 수행한 경우 ManagedConnection에 저장된 로컬 트랜잭션을 통해 커넥션 매니저에게 해당 ManagedConnection이 종료되었음을 알려줌으로써 커넥션 매니저는 ManagedConnection를 풀에 넣어 재사용한다.

(6)은 전역 트랜잭션에 XAResource를 등록하여 2PC 프로토콜을 이용하여 분산 트랜잭션을 처리하는 경우이다. 커넥션 매니저는 트랜잭션내에서 접근한 데이터베이스에 대한 URL을 검사하여 새로 요청한 커넥션이 이전에 요청되었던 데이터베이스와 전혀 다른 URL이라면 이전에 할당한 모든 ManagedConnection들에 대해 그들이 가지고 있는 XAResource들을 전역 트랜잭션에 등록한다. 이렇게 등록된 XAResource들은 전역 트랜잭션에 의해 관리되므로 2PC 프로토콜을 통해 물리적 커넥션 역시 전역 트랜잭션에 의해 커밋되거나 롤백된다. 따라서 이경우 로컬 트랜잭션은 물리적 커넥션에 대해 아무런 작업도 수행하지 않고 단지 커넥션 매니저에게 ManagedConnection의 사용이 완료되었음을 알리는 역할만을 수행하게 된다. 로컬 트랜잭션에 대해 자세히 논의하지 않았지만 하나의 리소스만을 사용할 때 2PC 오버헤드를 줄이기 위해 사용하는 것으로 MangedConnection 내부적으로 가지고 있으며 하나의 리소스만 사용될 경우 XAResource를 등록하지 않고, 단지 전역 트랜잭션이 커밋되면 로컬 트랜잭션이 커넥션에 대한 커밋을 담당한다.

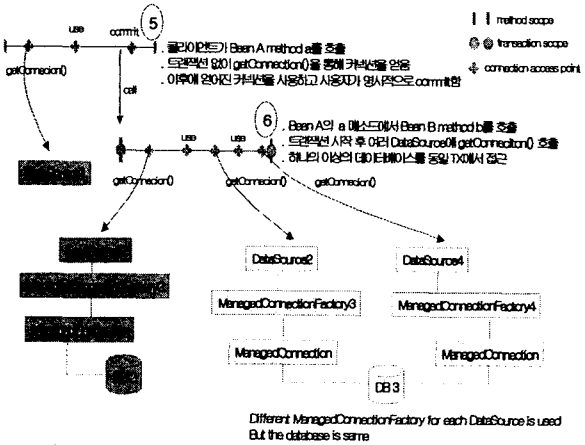

그림 4. 커넥션 요청 모델 II

5. 결론

본 논문에서는 EJB 컨테이너에서 다양한 리소스를 관리하기 위한 요구 사항에 대해 논의하였다. 또한 E504 EJB 컨테이너에서 이러한 요구 사항에 맞추어 데이터베이스 커넥션을 관리하기 위해 JCA를 구현한 커넥션 관리자에 대해서 간단히 논의하였다. 그리고 커넥션 관리자가 빈의 요청 시나리오에 따라 고려해야 할 사항에 대해 논의하였다. 부족한 지면으로 설계 상 이슈가 될 수 있는 로컬 트랜잭션 최적화 및 공유 알고리즘, XAResource가 연관된 전역 트랜잭션과 로컬 트랜잭션의 연관 모델과 같은 기술적 사항에 대해서는 다른 논문을 통해 논의하도록 하겠다. JCA는 본질적으로 EIS(Enterprise Information System)를 EJB 시스템에 연동하기 위해 제안된 방법이기 때문에 데이터베이스를 제외한 JMS, Mail, ERP 등의 다양한 시스템을 고려한 커넥션 관리자의 구조 확장이 필요하다.

6. 참고문헌

[1] Linda G. Demichiel 외 2인, "Enterprise Javabeans Specification, Version 2.0," Final Release, Sun Microsystems, 2001

[2] Rahul Sharma, "J2EE Connector Architecture, Final Release version 1.0," Sun Microsystems, 2001

[3] Susan Cheung 외 1인, "Java Transaction API (JTA) Version 1.0.1," Sun Microsystems, 1999

[4] "JBoss - World class J2EE technologies in open source," http://www.jboss.org

[5] http://www.openejb.org