XMDR-DAI를 이용한 데이터 이주 기법에 관한 연구

문석재* · 정계동* · 최영근*

A Study on Data Migration using XMDR-DAI

Seok-Jae Moon* · Gye-Dong Jung* · Young-Keun Choi*

요 약

기업들은 개별적이고 다양한 형태의 데이터베이스를 활용하여 업무적으로 발생하는 데이터를 저장, 관리하고 있다. 이렇게 개별적으로 존재하는 데이터베이스에 저장된 데이터들은 기업에서 특정 서비스 목표 달성을 위해서 비즈니스 프로세스라는 것을 통해 접근되고 있고, 데이터에 대한 이주가 빈번하게 이루어지고 있다. 이에 본 논문에서는 SQL Query기반의 비즈니스 프로세스가 수행되는 과정에서 데이터의 접근 및 통합에서 발생하는 데이터 이주 과정을 XMDR-DAI(eXtended MetaData Registry-Data and Integration)를 이용한 방법을 제시한다.

ABSTRACT

Enterprises saved and controled data using individual and various database which were occurred on business. The saved data that were existed in the database were approached by business processing for specified achievement of service's aims and were occurred frequently data migration. This research propose the data migration method using XMDR-DAI(eXtended Meta-Data Registry-Data and Integration) that were occurred by approaching and integration in the process of business processing based on sql query

키워드

XMDR, 데이터 접근, 통합, 이주, 상호운용

Key word

XMDR(eXtended MetaData Registry), Data access, Integration, Migration, Interoperability

* 광운대학교 컴퓨터과학과

접수일자 : 2009. 06. 30
심사완료일자 : 2009. 07. 23

Ⅰ. 서 론

현재 대부분의 기업들은 다양한 종류의 데이터베이스를 활용하여 데이터를 저장 및 관리하고 있다. 이처럼 분산되어 각각 운영되고 있는 데이터베이스들의 데이터 소스들은 기업의 특정 서비스 목표 달성을 위해 비즈니스 프로세스라는 것을 통해서 접근하고, 필요에 따라 통합되고 있다[1].

기업들은 다양한 데이터 소스들의 접근 및 통합에 대한 방법으로 EAI[2], ERP[3], DW[4]와 같은 기술을 사용하고 있다. 일반적으로 이런 기술들은 비즈니스 프로세스 수행 과정에서 빈번히 일어나는 데이터들의 이주에 관해서 미들웨어 측면에서 지원하고 있지만, 데이터 이주에서 발생하는 메타데이터 스키마 이질성, 데이터 이질성에 대한 솔루션이 미흡한 실정이다[5]. 따라서 본 논문에서는 레거시 또는 데이터베이스간의 데이터 상호운용에서 발생하는 이질성 문제를 XMDR-DB를 이용하여 해결하고, SQL Query기반의 프로세스가 수행되어 내부적으로 데이터 접근 및 통합에서 발생하는 데이터들의 이주 과정을 XMDR-DAI (eXtended MetaData Registry-Data and Integration)을 이용한 방법을 제시한다.

본 논문에서 제시한 XMDR-DAI를 이용한 데이터 이주 과정은 복제, 이동, 분할, 병합으로 분류하여 설명하고, 다음 3가지 사항을 가정한다.

- 분산되어 각각 운영되고 있는 데이터베이스들의 데이터 소스들을 접근하기 위한 SQL Query기반의 프로세스들은 XMDR을 참조하여 정의한다. 이 XMDR은 각 데이터베이스들의 메타데이터 스키마 기반으로 정의된 것으로써, 스키마 변환, 매칭, 매핑 및 인스턴스 값에 대한 연관 정보가 등록되어 있다.
- SQL Query기반의 프로세스는 사전에 XMDR의 메타데이터 스키마 정보를 참조하여 작성된 것으로 Query Repository에 등록되어 있는 것이다. Query Repository에 있는 프로세스들은 사용자들이 주기적으로 같은 작업을 반복하는 SQL Query가 미리 정의되어 등록되어져 있다.
- 사용자가 Query Repository를 이용하여 생성한 SQL Query기반의 프로세스는 XML 기반의 메시지로 분산되어 있는 데이터베이스들의 전달되고 SOAP 프로토콜을 이용하여 통신한다.

본 논문에서 제시한 분산 환경의 협업 데이터베이스 간의 데이터 이주 기법은 XMDR-DAI를 이용함으로써 데이터 상호운용에서 발생하는 이질성을 고려할 수 있고, 사용자에게 과도한 SQL Query에 생성 작업을 줄일 수 있다. 또한 Proxy-DB는 가상 데이터베이스 역할을 함으로써 데이터 상호운용에 필요한 데이터의 접근과 데이터 통합 서비스를 하는데 용이해진다. 본 논문의 구성은 2장에서는 관련연구를 서술하고, 3장에서는 XMDR-DAI를 서술하고, 4장에서는 XMDR-DAI를 이용한 데이터 이주 정책에 대해서 기술한다. 5장에서는 구현 및 비교분석에 대해 기술하고, 마지막 6장에서는 결론 및 향후연구에 대해 기술한다.

Ⅱ. 관련연구

XMDR(eXtended Meta-Data Registry)은 XML 기반의 RDB(Relational DataBase) 메타데이터를 객체지향 데이터베이스에 저장하는 기술과 분산된 데이터의 이질성을 해결하는 MDR(Meta-Data Registry), 그리고 데이터의 효율적인 이용을 위한 연관성을 정의한 온톨로지 시소러스(Ontology thesaurus)를 결합한 것이다[6].

XMDR은 ISO/IEC 11179에서 제안한 정보 공유 및 교환을 위한 표준으로 현재 많은 프로젝트가 진행 중에 있다. 특히, ISO/IEC 1119-3에서는 공유 데이터의 관리를 위한 메타 모델, 기본 속성이 제시되어있는데, 메타모델은 의미적인 내용과 분산된 환경하의 사용자들이나 정보처리 시스템간의 공유되는 데이터 요소의 구문을 위한 표준과 안내를 제공하고 있다. 본 논문에서의 DAI (Data Access and Integrator)은 XMDR을 이용하여 위와 같은 명확한 요구사항을 만족 하도록 구성된 것으로써, 효율적인 SQL Query 데이터의 상호운용을 수행할 수 있다. 또한 분산 데이터의 자율성과 독립성을 보장할 수 있고, 기본 데이터베이스들의 메타데이터 스키마의 재구성이나 변경 없이 사용이 가능하다[17].

Ⅲ. XMDR-DAI

본 논문에서 제시한 XMDR-DAI(eXtended MetaData Registry-Data and Integration)는 User Layer와 분산되어

있는 Data Layer의 DRs(Data Resources)사이에서 미들웨어로 위치한다. XMDR-DAI를 이용한 데이터 통합 서비스 구조는 그림1과 같다.

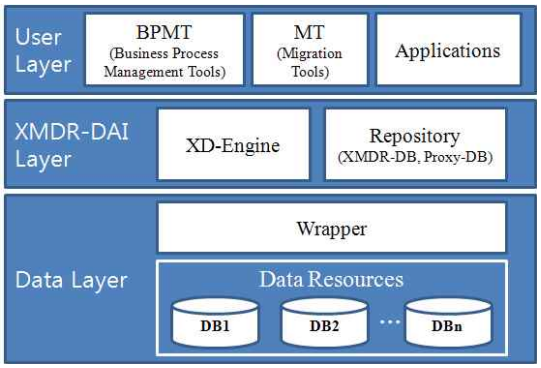

그림 1. XMDR-DAI를 이용한 데이터 통합 서비스 구조
Figure 1. Data integration service construction using XMDR-DAI

3.1 Users Layer

이 계층은 데이터 접근 및 이동에 관한 작업을 수행하기 위해서 사용자가 BPMT(Business Process Management Tools), MT(Migration Tools)등 애플리케이션을 이용하여 비즈니스 프로세스를 이션하거나 SQL Query를 이용하여 데이터를 복제, 이동, 분할, 병합 방법으로 이주를 지원해주는 애플리케이션들이 있는 영역이다. 이러한 애플리케이션들은 XD-Engine에서 지원하는 기능과 Repository(XMDR-DB, Proxy-DB)에서 제공하는 정보를 참조하여 데이터 액세스, 이주 등에 대한 작업을 정의 할 수 있다.

3.2 XMDR-DAI Layer

이 계층은 그림2에서와 같이 6개의 구성요소로 이루어진 XD-Engine과 Repository(XMDR-DB, Proxy-DB)를 이용하여, 데이터 액세스 및 통합, 이주에 대한 서비스를 할 수 있도록 지원해주는 영역이다. 예를 들어, 애플리케이션이 DRs(Data Resources)측에 데이터 삽입, 삭제, 변경에 대해서 요청한다. 이런 경우 Query 제어 기능과 데이터 이주 서비스를 처리할 때 발생하는 메타데이터 스키마 이질성, 데이터 이질성을 해결할 수 있다. 또한 데이터 수집할 수 있는 가상 데이터베이스인 저장소 기능을 제공한다.

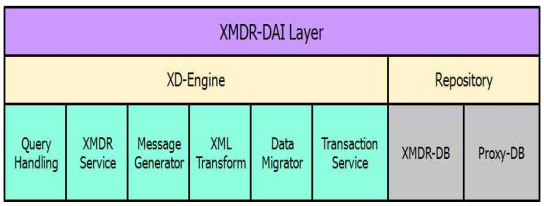

그림 2. XD-Engine, Repository 구성요소
Figure 2. XD-Engine, Repository component

- Query Handling: 이 요소는 XMDR-DAI에 중추적인 요소로써 SQL Query기반의 프로세스들의 흐름을 제어하고, 애플리케이션이 요청하는 사항에 의해 XMDR Service에게 메타데이터 스키마 정보 요청을 하는 역할을 한다. 또한, Users측에서 수행하고자 하는 프로세스에 대한 결과 데이터가 Proxy-DB에 있는지 확인하는 역할과 Users측에서 새롭게 정의하여 생성한 프로세스나 SQL Query가 DRs(Data Resources)에서 수행된 결과 데이터를 저장할 수 있도록 임시 테이블을 생성하는 역할을 한다.
- XMDR Service: 이 요소는 User층에서 SQL Query를 생성할 때 필요한 메타데이터 스키마 정보를 제공한다. 사전에 XMDR-DB에 등록된 메타데이터 스키마의 변환, 매칭, 매핑정보, 그리고 인스턴스 값들에 대한 연관 정보를 XMDR Service를 통해서 제공받을 수 있게 된다. 이에 사용자는 SQL Query를 작성할 때 스키마 재구성을 고려하지 않고 쉽게 구성할 수 있다.
- Message Generator: 이 요소는 각 데이터베이스에서 수행되어지는 SQL Query를 형식에 맞는 메시지로 구성하고, 생성하는 역할을 한다. 생성된 메시지에는 일반정보, 데이터베이스의 물리적 위치정보, 입력/출력 파라미터 정보, 버전정보, 메타데이터 스키마 정보, 그리고 데이터 간의 연관정보 등으로 구성된다. 이런 구성으로 생성된 메시지는 협업되어있는 데이터베이스 간의 상호운용에 필요한 데이터 공유 및 교환 수단으로 이용된다.
- Data Migrator: 이 요소는 각 데이터베이스에서 SQL Query가 수행되어 나온 결과 데이터를 Proxy-DB에 저장하거나, 수행된 결과 데이터가 다른 데이터베이스로 이주되는 과정에서 데이터의 흐름을 제어할 수 있는 기능을 제공한다. DM(Data Migrator)는 XMDR기반으로 미리 정의되고 작성된 SQL Query가 수정 없이 수행되어 데이터 이주가 발생하는 경우와 사용자가

스키마 정보를 참조한 후, SQL Query가 작성되어 수행되는 경우에 발생하는 데이터 이주에 대한 흐름을 제어한다.

- XML Transform: MG(Message Generator)에서 생성된 메시지를 이 기종 환경에 맞게 트리 기반의 XML 문서로 변환시켜주는 역할을 한다. XML 문서로 변환된 메시지는 각 데이터베이스들에 SOAP을 통해 전달, 전송된다.
- Transaction Service: 이 서비스는 트랜잭션을 상태를 파악하고, 파악된 정보를 로그에 저장하는 역할을 수행하여 트랜잭션의 상태를 정확하게 조정자에게 알려주는 역할을 하는 감시자(Monitor)와 실제 트랜잭션을 운영하고 상태에 따른 운영지시를 수행하는 조정자(Coordinator)로 구성된다. 본 XMDR-DAI에 트랜잭션은 데이터 이주 시 Proxy-DB에 대한 처리 방법이고, 가장 간단하고 널리 사용되는 완료 규약인 2PC방식[16]을 적용한다.
- XMDR-DB: XMDR-DB는 분산 데이터의 자율성과 독립성을 보장할 수 있도록 구성되어 있다. 또한 기본 데이터베이스들의 스키마의 재구성이나 변경 없이 사용이 가능하도록 되어 있다. XMDR-DB의 구성은 데이터베이스들의 메타데이터 스키마 정보들을 시소러스화하여 표준에 맞추어 기한 것이다. 메타데이터의 관계성, 이질성을 해결한 메타-시멘틱 온톨로지, 레거시 시스템들의 물리적인 정보인 위치정보, 접근 권한 등을 관리하는 메타-로케이션, 그리고 실제 상호운용되는 데이터 값과 분산 데이터 값들 간의 연관관계성을 시소러스화하여 정의한 인스턴스-시멘틱 온톨로지로 이루어져 있다[9].
- Proxy-DB: 사용자가 정의한 프로세스가 DRs(Data Resources)에서 수행되어 리턴 된 결과 데이터 또는 자주 사용되는 프로세스들이 주기적으로 수행되어 리턴 된 결과 데이터가 저장되어 있다. Proxy-DB는 협업 환경에서 각각 분산되어 있는 데이터 소스들을 통합하여 사용자에게 제공하는 서비스를 하고, 분산 데이터베이스간의 데이터를 교환할 수 있는 서비스를 제공한다.

3.3 Data Resources Layer

이 계층은 User 층에게 데이터 소스에 대한 접근과 통합된 데이터 정보를 제공해주는 소스로써 실제 협업되고 있는 데이터베이스들의 집합이다. 각 데이터베이스들은 프로세스를 통해 수행된 결과 데이터를 제공하고, 다른 데이터베이스들의 요청한 데이터 상호운용을 Wrapper를 통해 서비스한다.

Ⅳ. XMDR-DAI를 이용한 데이터 이주

본 장에서는 분산 환경에서의 협업된 데이터베이스 간의 발생하는 데이터 이주 기법에 대해서 제시한다. 제시한 이주 기법은 XMDR-DAI를 통해서 데이터를 통합하는 과정에서 발생하는 데이터 이주 이벤트를 복제, 이동, 분할, 병합에 대한 경우로 구분하여 고려한 것이다.

4.1 데이터 이주

협업된 데이터베이스, 데이터베이스 클러스터, 그리고 그리드 데이터베이스 등과 같은 환경에서 데이터 이주는 사용자가 어떤 목적으로 Query를 정의하였는지에 따라서 접근 빈도, 접근 경로, 데이터 이주 크기가 결정된다. 일반적으로 데이터 이주는 크게 복제 및 분할로 이루어진다. 우선 복제는 원본 데이터로부터 생성되는 뷰, 스냅샷, 가상 테이블 정의 등을 이용한 복사본이다. 그리고 분할은 데이터가 RDB에서의 릴레이션에 스키마 변경에 의해서 수직 분할, 데이터 범주에 따른 분리 과정인 수평 분할로 나누워지며, 시스템 환경이나 데이터 서비스 목적에 알맞게 적용하는 방법에서 차이가 있다. 또한 비즈니스 프로세스에서 사용자가 작성한 query 패턴에 의해 적용되는 범위도 다르다. 따라서 협업된 데이터베이스 간의 발생하는 데이터 이주는 복제, 이동, 분할, 병합 4가지로 분류할 수 있다.

4.1.1 XMDR-DAI의 운용 시나리오

다음은 통합 자산 운용관리 시스템에서 XMDR-DAI를 이용하여 자산에 대한 운영을 2가지 시나리오 측면에서 SQL Query기반의 프로세스로 수행되는 과정을 전반적으로 나타낸 것이다.

첫 번째, 그림3에서 보는 봐와 같이 applications에서 정의되어 수행하려고하는 SQL Query기반의 저장-프로시저인 sp_insa_tamat100, sp_asset_info가 QH(Query Handling)에게 전달된다. 이 두 개의 저장-프로시저는 XMDR-DB을 참조 및 적용되어 생성된 것이다.

applications에서 sp_insa_tamat100, sp_asset_info가 수행되기 전에 사용자는 사전에 수행되어있는 결과 데이터가 있는지 확인한다. 만약 없다면 스키마 정보를 이용해서 테이블을 Proxy-DB에 생성한다. QH(Query Handling)에 전달된 SQL Query는 해당 데이터베이스에 wrapper에게 전송되어 Query에 대한 실행을 하게 된다.

그림3에서처럼 sp_insa_tamat100, sp_asset_info는 해당 데이터베이스인 PersonsDB, AssetDB에서 정의된 순서대로 수행되고, 그 결과 데이터가 DM(Data Migrator)에게 전달되고, DM은 결과 데이터를 Proxy-DB에 이주하게 된다. 사용자는 applications를 통해 수행되어 이주된 결과(데이터)를 확인하게 된다.

두 번째, applications에서 XMDR-DB에 등록된 메타데이터 스키마 정보를 보고, SQL Query를 작성하여 프로세스가 수행되는 경우 데이터 이주되는 과정을 보여준다. applications에서 수행하고자 하는 프로세스를 Proxy-DB에 있는 결과 데이터와 XMDR-DB에서 제공하는 메타데이터 스키마 정보를 참조하여 SQL Query를 생성한다. QH(Query Handling)에게 전달된 Query는 해당 데이터베이스에 wrapper에게 전송되어 실행하게 된다. 그림5에서 QH(Query Handling)의 의해 전송된 SQL Query는 PurchaseDB에 실행되어 DM(Data Migrator)에게 결과 데이터가 전달되고, DM(Data Migrator)은 이 데이터를 Proxy-DB에 이주하게 된다.

사용자는 applications를 통해 수행되어 이주된 결과 데이터를 확인하게 된다.

4.1.2 복제 시나리오

특정 데이터베이스에서 과도한 작업부하가 생기는 경우, 또는 같은 형태의 SQL Query를 빈번하게 다른 데이터베이스로부터 특정 데이터를 요구할 경우 복제가 발생한다. 데이터 복제는 임의의 데이터베이스 내의 읽기 연산이 빈번한 데이터에 대하여 복사본을 해당 데이터베이스에 생성하는 것이다. 이런 경우는 데이터를 복제함으로써 사용자의 접근을 분산시킴으로써 작업에 대한 부하를 나누게 한다. 또한, 공간 데이터와 같이 업데이트가 거의 발생하지 않는 데이터를 복제함으로써 네트워크에 대한 비용을 줄일 수 있다.

그림4는 그림3의 통합자산운용관리시스템에서 XMDR-DAI의 Proxy-DB를 이용한 데이터가 복제가 되는 경우를 나타낸 것이다. 새로운 자산을 신청하기 위해서 자산신청 애플리케이션을 이용하여 SQL Query 기반인 sp_insa_tamat100(인사정보)와 sp_asset_info(자산정보)를 주기적으로 PersonsDB와 AssetDB에 접근하여 데이터를 요구하여 읽기 연산이 빈번한 경우이다.(단, 인사정보, 자산정보는 업데이트가 빈번하게 발생하지 않는 경우이다.)

이러한 경우에는 그림3에서처럼 자산신청 애플리케

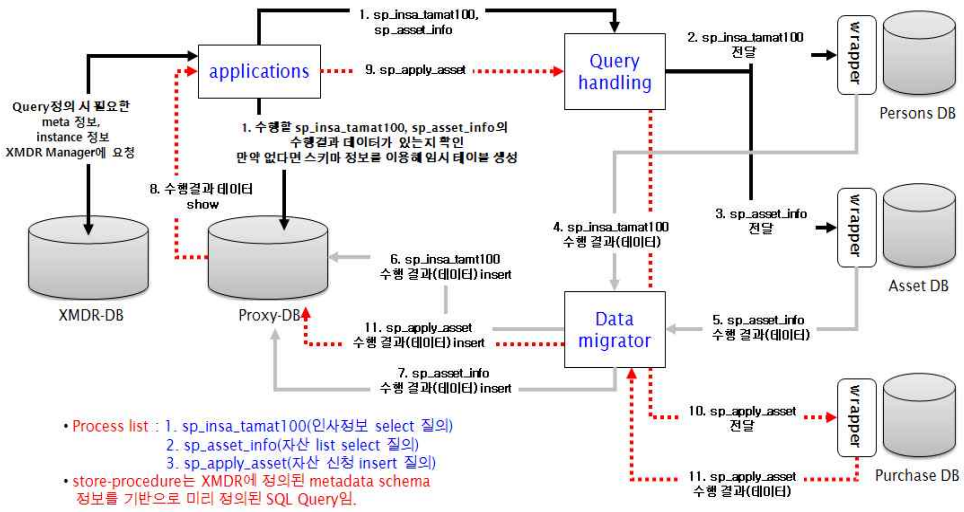

그림 3. XMDR-DAI의 Proxy-DB를 이용한 데이터 이주 흐름 과정
Figure 3. Data Migration using Proxy-DB of XMDR-DAI

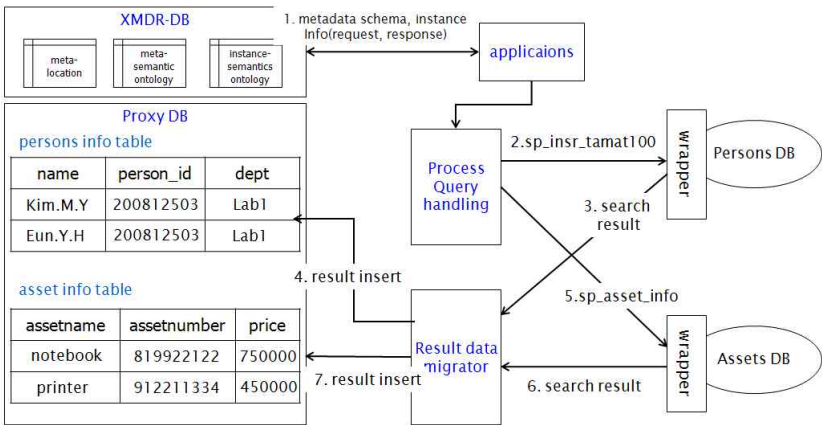

그림 4. 데이터 이주 1: 복제
Figure 4. Data Migration 1: replication

이션에서 주기적으로 PesrsonsDB와 AssetDB 측에 주기적으로 읽기 연산을 수행해야 함으로 데이터 복제가 필요하다. 이는 업데이트가 거의 발생하지 않는 것이다.

그림4에서는 표1에서의 QP1:sp_insa_tamat100, QP2: sp_asset_info가 수행되어 Proxy-DB에 DM(Data Migrator)에 의해서 데이터가 복제된 것이다.(QP1, QP2는 XMDR-DB를 참조하여 작성됨)

이와 같이 데이터를 복제함으로써 사용자의 접근을 분산시킴으로써 작업에 대한 부하를 적게 할 수 있다. Proxy-DB에 복제된 데이터인 인사정보와 자산정보는 주기적으로 발생하는 자산신청 프로세스에서 SQL Query 작성 할 때 이용된다.

표 1. SQL-Query 1(저장-프로시저)
Table 1. SQL-Query 1(Store-Procedure)

QP1 : *sp_insa_tamat100*
select name, person_id, dept from tamat400 where dept_code='8322' and dept_name='lab1'

QP2 : *sp_asset_info*
select assetname, assetnumber, price from assetinfo where item_code='it01'

4.1.3 이동 시나리오

특정 데이터베이스에 의한 특정 데이터로의 접근이 빈번한 경우, 또는 특정 데이터베이스에서 쓰기 연산이 빈번하게 발생하는 경우에 데이터에 대한 이동이 발생한다. 데이터 이동은 접근이 빈번한 데이터를 쓰기 연산을 자주 요구하는 데이터베이스로 데이터를 이동함으로써 응답 속도를 향상시키는데 목적이 있다. 그림5는 그림4에 복제 과정이 수행된 후에, XMDR-DAI의 Proxy-DB를 이용하여 데이터가 이동이 되는 경우를 나타낸 것이다.

표 2. SQL-Query 2(저장-프로시저)
Table 2. SQL-Query 2(Store-Procedure)

QP3 : *sp_apply_asset*

```
@p_persons_info.name VARCHAR(10),
@p_persons_info.person_id VARCHAR(9),
@p_persons_info.dept VARCHAR(20),
@p_asset_info.set_info. VARCHAR(100),
@p_asset_info.assetnumber VARCHAR(25),
@p_asset_info.price INT(10);

Insert Inot apply_item(apply.name,
        apply.person_id, apply.dept,
        apply.assetname, apply.assetnumber)
Value (p_persons_info.name,
        p_persons_info.person_id,
        p_persons_info.dept,
        p_asset_info.assetname,
        p_asset_info.assetnumber,
        p_asset_info.price);
```

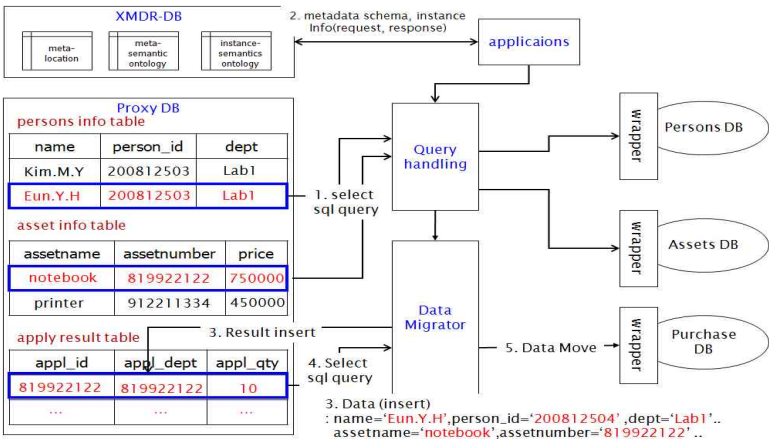

그림 5. 데이터 이주 2: 이동
Figure 5. Data Migration 2: Move

표2의 QP3:sp_apply_asset에서 value의 파라미터 값으로 그림5에서 수행된 Proxy-DB에 있는 데이터가 이용된다. QP3:sp_apply_asset은 자산신청을 하는데 있어서 반드시 인사정보와 자산정보가 필요한 SQL Query이다. 표2와 같이 작성된 SQL Query는 자산신청을 할 때마다 주기적으로 PurchaseDB에 쓰기 연산이 발생하는 경우이다. 이런 경우에는 쓰기 연산이 자주 요구되는 PurchaseDB에 부하를 줄이고, Proxy-DB에서 자신신청의 쓰기 연산을 대신함으로써 응답속도를 높일 수 있게 된다. 애플리케이션에서는 QP3:sp_apply_asset에 대한 처리를 Proxy-DB을 통해 수행할 수 있게 되고, 이후에 일괄처리 형태로 쓰기 연산이 된 데이터는 실제 PurchaseDB에 이동이 된다.

4.1.4 분할 시나리오

특정 데이터에 대한 쓰기 연산이 빈번하고, 접근 빈도가 높아 데이터베이스에 부하가 커지는 경우이다. 하지만 데이터 분할은 데이터 이동과정에서와는 달리 데이터에 접근하는 빈도나 쓰기 연산이 여러 데이터베이스에서 요청할 경우에 분할하여 이동 시킬 때 발생한다. 데이터 분할은 데이터를 요청한 각 데이터베이스 측에 복제본을 각각 이동함으로써 실질적인 작업 부하의 분산을 유도한다. 그림6은 Proxy-DB에 복제되어 있는 자산정보 데이터를 애플리케이션 및 레거시 시스템들이 빈번하게 쓰기 연산을 요청하는 경우에 대한 데이터 분할에 대해 나타낸 것이다. 예를 들어, 각각의 레거시 시스템이나 데이터베이스에서 그림6에서 보는 봐와 같이 각각 Proxy-DB에 있는 자산데이터에 대해서 insert, update 등과 같은 SQL Query를 빈번하게 요청하게 될 때 자산데이터에 대한 접근 빈도나 쓰기 연산이 높아지므로 Proxy-DB에 대한 부하가 커진다. 이때 XMDR-DAI에서 자산데이터에 한해서 각각 요청되는 레거시 시스템이나 데이터베이스에 자산데이터 복제본을 분할하여 이주를 시킨다. 이렇게 함으로써 실질적인 특정 데이터 작업 부하를 줄일 수 있다.

4.1.5 병합 시나리오

임의의 데이터베이스가 특정 데이터베이스에 있는 데이터에 대해 쓰기 연산이나 접근 빈도가 높은 경우인 데이터 분할과 반대되는 과정으로써 여러 데이터베이스에 각각의 해당되는 데이터에 대한 쓰기 연산이나 접근 빈도가 높은 경우에 발생한다. 데이터 병합은 분산되어 있는 데이터 접근 시 조인 연산에 의한 비용이 각각의 데이터에 대한 단일 접근 비용보다 커져서 분할 상태가 오히려 성능을 낮추는 경우에 수행한다. 그림7은 Proxy-DB에 이주된 데이터들을 특정 애플리케이션 또는 레거시 시스템에서 인사정보와 자산정보 데이터에 대해서 조인 연산에 의한 접근이 빈번하여 각각 데이터에 대한 접근 비용이 더 높아서 인사정보와 자산정보 데이터를 병합한 경우이다. 그림7에서처럼 Proxy-DB에 이

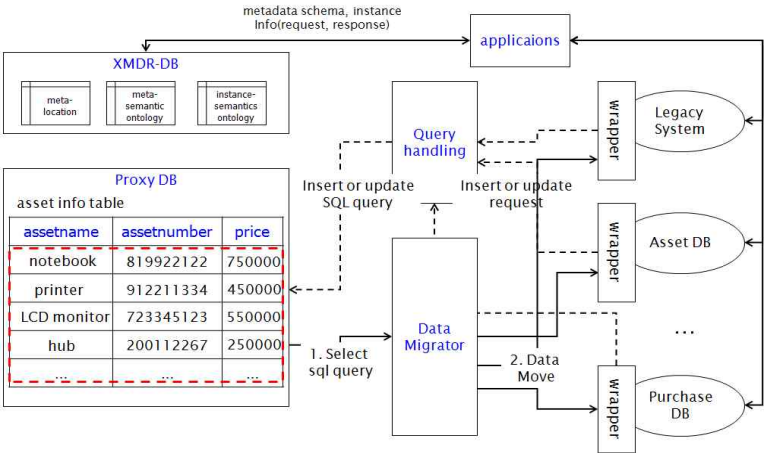

그림 6. 데이터 이주 3: 분할
Figure 6. Data Migration 3: Move

주되어온 인사정보와 자산정보에 대한 각각을 하나의 테이블에 병합하여 여러 애플리케이션 또는 레거시 시스템에서 데이터에 대한 쓰기, 접근을 할 때 Join형식의 SQL Query를 할 필요가 없어지게 되므로 사용자는 Join에 대한 부담이 줄어든다.

4.2 XMDR-DAI의 Proxy-DB에서 트랜잭션 운용

본 논문에서 제시한 XMDR-DAI는 이용한 분산 환경에서의 협업된 데이터베이스 간의 데이터 이주를 Proxy-DB를 이용함으로써 발생하는 트랜잭션에 대한 설명한다. 트랜잭션을 운영하기 위한 완료조건은 각 트랜잭션에 참여하게 되는 모든 데이터베이스에서 동의가 되며, 이로써 트랜잭션의 원자성이 보장된다. 이러한 방법으로 2PC방식[16]을 사용한다.

XMDR-DAI의 트랜잭션 조정자 : Ci(i=1, 2, ,... n)
XMDR-DAI의 트랜잭션 감시자 : Mi(i=1, 2, ..., n)
레거시의 트랜잭션 감시자 : LMj(j=1, 2, ..., m)
i : 트랜잭션 번호, j : 데이터베이스 시스템 번호

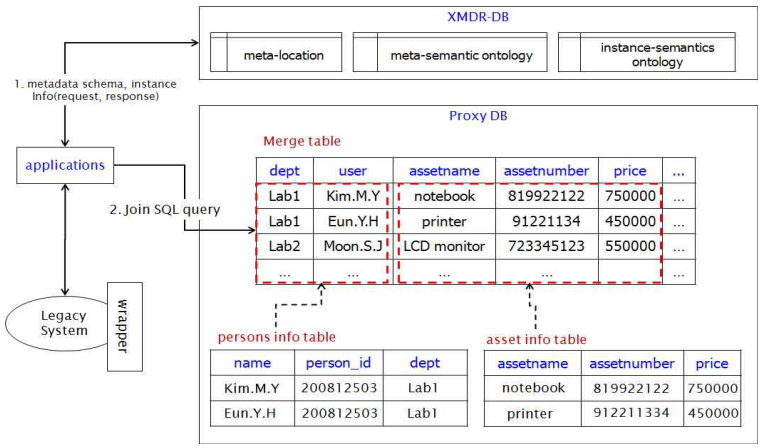

그림 7. 데이터 이주 4: 병합
Figure 7. Data Migration 4: Merge

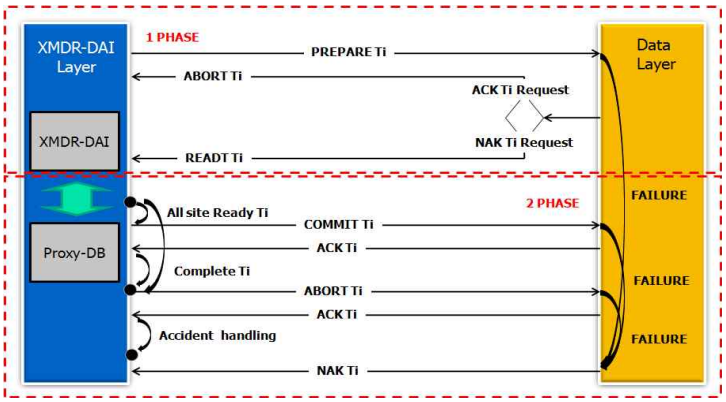

그림 8. 2PC 트랜잭션 처리
Figure 8. 2PC T Tnsi TnsiP ocess

4.2.1 2PC(Two Phase Commit)의 적용

Ci가 트랜잭션을 생성하여 시작하며 이는 트랜잭션 감시자(Mi)에 의해 로그에 기록되고, 트랜잭션이 수행된 모든 LMj들의 완료를 Ci가 받았을 때, 해당 조정자는 2PC를 시작한다.

• 1 Phase

트랜잭션의 시작은 로그에 <start Ti>레코드가 추가됨으로서 시작된다. 다음으로 각 데이터베이스 준비여부를 확인하기 위해 <prepare Ti>레코드를 추가하고 모든 DRs에 prepare Ti 메시지를 전송한다. 데이터베이스는 준비여부를 확인하여 준비상태이면 <ready Ti>를 LMj에 추가하고 ack Ti를 전송하여 XMDR-DAI에 <ready Ti>를 로그에 저장하도록 Ci에게 응답하고, 준비상태가 아닐 경우 <abort Ti>를 LMj에 추가하고 nak Ti를 전송하여 XMDR-DAI에 <abort Ti>를 로그에 저장하도록 Ci에 응답한다.

• 2 Phase

Ci가 모든 데이터베이스로부터 prepare Ti에 대한 응답을 받았을 때, 또는 일정시간 응답이 없을 경우, Ci는 트랜잭션 계속유무를 판단한다. 모든 응답이 <ready Ti>이면 레거시에 <commit Ti>를 전송하여 트랜잭션을 완료시키고, <abort Ti>를 하나라도 포함하고 있으면 Ti는 중단된다. <commit Ti>나 <abort Ti>를 로그에 저장하고 모든 DRs에 <commit Ti>나 <abort Ti>메시지를 전송하고 그 결과에 대해서 <ack Ti>신호를 받으면 로그에 <complete T>레코드를 로그에 저장하고 Ti를 종료한다.

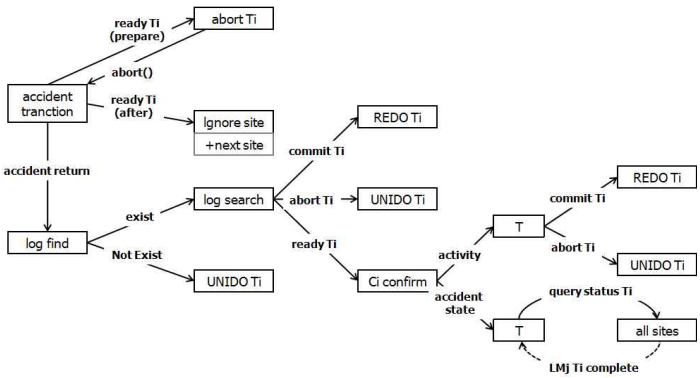

그림 9. 데이터베이스 고장에 대한 처리
Figure 9. Process of Database fault

여기서 고장이 발생하거나 데이터베이스에서 동작실패가 발생할 경우 <nak Ti>를 XMDR서버에 전송하여 고장처리를 할 수 있도록 한다.

4.2.2 2PC(Two Phase Commit)의 고장처리

고장으로 인한 장애가 발생했을 경우 DB의 고장과 조정자의 고장에 대해서 서로 다른 방식으로 대처한다. 레거시에 의한 장애의 고장 처리는 ready Ti 메시지에 대한 응답에 따라 처리한다. DB가 ready Ti 신호이전에 장애가 발생했다면 abort Ti를 전송한 것으로 가정한다. ready Ti 이후에 장애가 발생했다면 장애가 발생한 DB는 무시하고 다음 DB로 넘어가서 계속 수행한다. 장애로부터 복귀하면 로그조사를 통해 로그가 존재하지 않으면 undo Ti를 수행하고, 존재하면 다음을 따른다.

- 로그에 commit Ti를 포함한 경우 redo Ti를 수행
- 로그에 abort Ti를 포함한 경우 undo Ti를 수행
- 로그에 ready Ti를 포함한 경우 Ci를 확인하여 동작중이면 Ti의 상태에 따라 commit Ti이면 redo Ti를 abort Ti이면 undo Ti를 수행한다.

Ci가 고장상태이면 모든 레거시에 Query를 전송하여 Ti의 정보를 가지고 있는 레거시가 회복될 때까지 지속한다. Ci의 고장은 DB들이 Ti의 운명을 결정해야 한다. DB들이 Ti의 운명을 결정하지 못하고 Ci의 회복을 기다려야 할 때도 있다. 다음과 같은 경우들을 보자.

- Ti에 포함된 레거시의 로그에 <commit Ti>가 있는 경우 Ti는 commit되어야 한다.
- Ti에 포함된 레거시의 로그에 <abort Ti>가 있는 경우 Ti는 abort되어야 한다.
- Ti에 포함된 레거시의 로그에 <ready Ti>가 있는 경우 Ci의 회복을 기다리는 것보다는 abort Ti를 수행한다.
- 그이외의 경우는 Ci가 회복되도록 기다린다.

트랜잭션 모니터는 각 트랜잭션의 진행상황을 파악하여 조정자(Ci)에 의해 발생되는 작업들을 로그에 저장하여 관리하는 역할을 수행한다. 트랜잭션의 운명을 결정하기 위해서 로그의 내용을 확인하므로 로그는 안전한 관리가 필요하다. 이러한 로그 정보의 기록과 관리는 수행하고 이를 조정자(Ci)에게 제공하는 역할을 수행한다.

V. 구현 및 비교분석

5.1 구현

본 논문에서의 XMDR-DAI 구현은 Windows2003, .NET으로 구축하였으며, Query Repository는 Oracle 8i를 사용하여 등록하였고, Proxy-DB는 Oracle8i를 사용하였다. 적용은 데이터 마이그레이션 툴을 이용하여 통합유형 자산 관리에서 자산 신청을 처리하는 과정에서의 일부분을 예로 하였다. 그림10은 사용자가 Query Repository를 접근하여 수행하고자 하는 SQL Query를 선택한다.(필요에 따라서 SQL Query 수정가능) 그리고, SQL Query 수행에 따른 결과 데이터를 저장하기 위해 Proxy-DB에 테이블 생성할 SQL Query를 작성한다. 선택된 SQL Query는 XMDR-DAI를 통해서 해당 시스템에 전달되어 수행되고, 결과 데이터 는 Proxy-DB에 복제되어 저장된다.

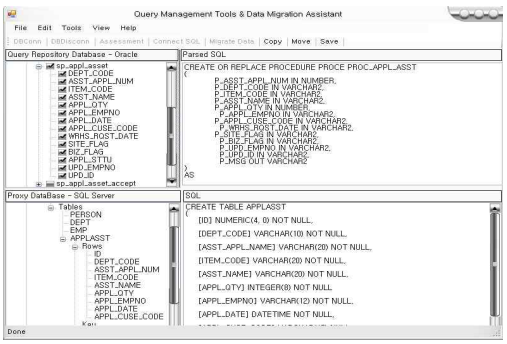

그림 10. 데이터 관리 툴 적용 1
Figure 10. Data management tools apply 1

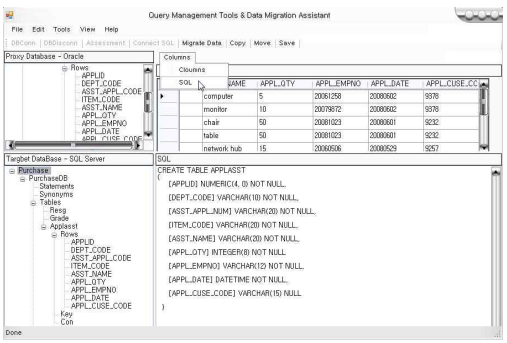

그림 11. 데이터 관리 툴 적용 2
Figure 11. Data management tools apply 2

그림11는 그림10에서 수행된 Query가 처리되어 리턴된 결과(데이터)가 Proxy-DB에 저장되어진 화면이다. 사용자는 Proxy-DB에 저장된 결과를 확인, 참조한다. 그리고 다음 수행해야할 프로세스에 따란 타켓 데이터베이스 측으로 데이터 이동을 해야 할 경우 SQL Query에 인스턴스 값을 추가하여 작성하는 화면이다.

5.2 비교 및 시험 평가

본 논문에서는 SQL Query 기반의 비즈니스 프로세스가 수행되는 과정에서 데이터 이주를 XMDR-DAI을 이용하였다. 이에 따라 OGSA[7], Grid-DBMS[8]와 비교 분석한 내용은 표3과 같다.

표 3. 타 시스템과 비교 분석
Table 3. The comparison of frameworks

		OGSA	Grid-DBMS	XMDR-DAI
데이터가용성	연속성	○	×	○
	접근성	○	○	○
	유연성	△	×	○
데이터투명성	이질성 극복	○	△	○
	데이터 교환	○	△	○
데이터활용	검색 효율성	△	△	○
	분석 효율성	△	×	△

XMDR-DAI를 이용한 4가지 이주 정책에 따라서 각 레거시 시스템이나 데이터베이스에서 데이터 소스 접근 및 통합 서비스를 할 때 연속성, 접근성, 유연성을 보장할 수 있다. 또한 SQL Query 기반의 프로세스가 수행되는 과정에서 발생하는 메타데이터 스키마 이질성, 데이터 이질성대한 문제를 XMDR을 이용하여 극복할 수 있다. 그리고 데이터 활용 측면에서도 단일 Query로 검색이 가능하고, Query 조건에 연관성을 가지는 데이터를 검색할 수 있어 효율적이다. 본 시스템에서 XMDR-DAI를 이용한 통합 자산운운용관리시스템에서 SQL Query 기반의 프로세스에 따른 데이터 접근, 수집, 통합 서비스를 수행할 때 Query에 대한 처리 시간에 대해서 성능을 평가한다.

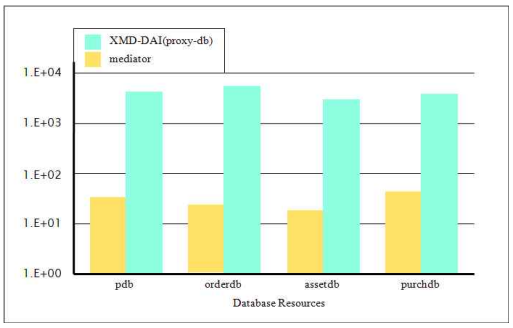

그림 12. Query 처리 시간 비교
Figure 12. The comparison of Query process time

성능 평가를 위한 테스트는 협업된 4개의 데이터베이스 서버를 이용하였고, 약20,000건 이상의 결과를 가지는 SQL Query문을 사용하였다. 그림12는 SQL Query문을 Proxy-DB를 이용하여 각 데이터베이스 서버에서 처리한 Query 처리 시간과 mediator를 이용한 Query 처리 시간을 비교하였다. pdb는 mediator를 이용한 Query 처리 시간이 약55ms 정도의 시간이 걸린데 반에, XMDR-DAI를 이용한 Query 처리 시간이 약7000ms 정도 걸린다. orderdb도 약40ms와 약8000ms 시간이 소요된다.

XMDR-DAI를 이용한 처리 시간이 mediator를 이용한 처리 시간보다 약6000~7000ms 정도 더 소요되지만, 이는 각 데이터베이스 서버마다 네트워크에서 소요되는 시간, 각 데이터베이스 서버의 래퍼에서 소요하는 시간, XMDR-DB와 Proxy-DB에 접근하는데 소요되는 시간이다. 그림11에서처럼 테스트한 결과에 따르면 각 소요되는 시간이 거의 일정하고, 각 요소마다 처리되는 시간이 전체 시간에 비해 미약하다. 그러므로 XMDR-DAI를 이용한 데이터 접근 및 통합하는 방식을 이용하는 것이 기존 mediator를 이용하는 것보다 비교적 효율적이라 할 수 있다.

Ⅵ. 결론 및 향후연구

본 논문에서는 분산되어 각각 운영되고 있는 데이터베이스들의 데이터 소스들이 SQL Query 기반의 프로세스를 이용하여 수행하는 과정에서 내부적으로 발생하는 데이터 이주에 대하여 XMDR-DAI를 이용하여 복제, 이동, 분할, 병합 정책에 대해 제시하였다. 또한 XMDR

을 이용하여 프로세스 수행에 따른 데이터 소스들에 대한 접근과 이동에서 데이터 이질성, 데이터 트랜잭션을 고려하였다. 따라서 본 논문에서 제시한 XMDR-DAI를 이용한 데이터 이주 기법은 분산되어 각각 운영되고 있는 데이터베이스들의 데이터를 비즈니스 프로세스를 처리하는 기업환경에 적합한 방법이다. 이후에는 워크플로우 시스템간의 데이터 상호운용이 될 수 있도록 확장되어야 한다. 그리고 Query Repository가 웹 서비스까지 확장되어, 기업환경에서 다양한 서비스를 제공하기 위한 서비스 계층까지 확산되는 연구가 필요하다.

참고문헌

[1] dwards, P. & Newing, R. *Application Integration for E-business.*

[2] ILUMA Technologies, Inc. Copyright © 2001. "Enterprise Application Integration (EAI)". http://www.iluma.com/solutions_eai.asp

[3] Lee, J., Siau, K., and Hong, S. "Enterprise Integration with ERP and EAI", *Communications of the ACM,* Vol. 46, No. 2, 2003, 54-60.

[4] Jung, R., Winter, R.: Justification of Data Warehousing Projects, Research Report, Competence Center Data Warehousing Strategy, University of St. Gallen, St. Gallen 2000.

[5] M. Madhavaram, D. L. Ali, M. Zhou, Integrating Heterogeneous Distributed Database Systems, Computers and Industrial Engineering, 31(1/2), (1996) 315-318.

[6] Kevin D. Keck and John L. McCarthy, "XMDR: Proposed Prototype Architecture Version 1.01", http://www.xmdr.org, February 3, 2005.

[7] Antonioletti, M., Atkinson, M, Baxter, R., Borley, A., Chue Hong, N., Collins, B., Hardman, N., Hume, A., Knox, A., Jackson, M., Krause, A., Laws, S., Magowan, J., Paton, N., Pearson, D., Sugden, T., Watson, P. and Westhead, M., *The design and implementation of Grid database services in OGSA-DAI.* Concurrency and Computation: Practice and Experience, 2005. 17(2): p. 357-376.

[8] G ALoisio, M Cafaro, S Fiore, M Mirto, "The Grid-DBMS:Towards Dynamic Data Management in Grid Environments", Information Technology:Coding and Computing(ITCC'05), IEEE, Vol.2,pp.199-204, April 2005.

[9] 문석재, 정계동, 강석중, 최영근, "저장-프로시저 기반의 비즈니스 프로세스 상호운용을 위한 XMDR Hub 프레임워크", 한국해양정보통신학회 제12권 12호 p.2207~2218, 2008.12

저자소개

문석재(Seok-Jae Moon)

2002년 독학사 전자계산학 이학사
2004년 광운대학교 컴퓨터소프트웨어학과 석사
2004년~2005년 필컴정보시스템 주임연구원
2006년~ 현재 광운대학교 컴퓨터과학과 박사과정
※관심분야 : XMDR, 데이터 그리드, 상호운용성

정계동(Gye-Dong Jung)

1985년 광운대학교 전자계산학 졸업
1992년 광운대학교 산업정보학 석사
2000년 광운대학교 컴퓨터과학박사
1993년~2004년 광운대학교 정보과학원 교수
2005년~ 현재 광운대학교 교양학부 교수
※관심분야 : XML 분산시스템, 분산 컴퓨팅기술, 이동에이전트

최영근(Young-Keun Choi)

1980년 서울대학교 수학교육과 이학사
1982년 서울대학교 계산통계학과 이학석사
1989년 서울대학교 계산통계학과 이학박사
1983년~ 현재 광운대학교 컴퓨터과학과 교수
1992년~2000년 광운대학교 전산정보원 원장
2002년~2005년 광운대학교 교무연구처장
※관심분야:객체지향설계, 분산시스템, 이동에이전트, 상호운용성