

# 그리드 데이터베이스에서 계층 구조를 이용한 동적 재조직 기법

## A Dynamic Reorganization Method using the Hierarchical Structure in the Grid Database

천종현\* / Jong-Hyeon Cheon

장용일\*\* / Yong-il Jang

조숙경\*\*\* / Sook-Kyoung Cho

이순조\*\*\*\* / Soon-Jo Lee

배해영\*\*\*\*\* / Hae-Young Bae

### 요약

그리드 데이터베이스는 그리드 컴퓨팅 환경에서 분산된 데이터의 효율적 처리와 사용을 위한 데이터베이스 관리 시스템이다. 공간 데이터는 일반적인 데이터에 비해 지역적 특성에 따른 이용도가 높으며, 대용량의 저장 공간을 필요로 하는 특징을 포함한다. 그리드 데이터베이스는 이러한 공간 데이터의 관리를 위한 최적의 시스템으로 적용 가능하다. 그러나, 기존의 분산 데이터베이스 시스템과는 달리 지역적 자율성을 보장하기 때문에 단일 관리자에 대한 적용이 불가능하거나 비효율적인 시스템 구성이 이루어질 가능성이 있다. 또한, 동적으로 변화되는 환경에 유동적인 대응을 위해서는 효율적인 재조직 연산이 필요하다.

본 논문에서는 그리드 데이터베이스에서 계층 구조를 이용한 동적 재조직 기법을 제안한다. 제안 기법은 재조직 수행 시 데이터베이스들의 정보를 논리적으로 취합하는 오가나이저를 생성한다. 또한, 오가나이저를 계층적 구조로 구성을 함으로써 단계적인 정보 전파 및 재조직 연산 수행을 지원하게 된다. 이를 통해 지역적인 재조직 연산의 지원 및 효율적 균형 조정이 가능해진다. 부가적으로 불필요한 재조직 연산의 제거가 가능하다. 성능평가를 통해 제안 기법이 재조직 수행 후에 전체적인 처리량을 향상시킴을 보인다.

### Abstract

A Grid Database is a database management system to process effectively and use the distributed data in a grid computing environment. Spatial data is more important than other general data according to the local characteristics and requires a large storage. The grid database can be used as the optimal system for the management of the spatial data. However, contrary to the conventional distributed database systems, the Grid Database which guarantees the local autonomy has a possibility not to provide an effective system, or it is impossible to use a centralized management environment. In order to allow flexible responses to a dynamically changing environment, it is required to use effectively reorganized method.

■ 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성·지원사업의 연구 결과로 수행되었음.

■ 논문접수 : 2006. 3. 16 ■ 심사완료 : 2006. 4. 12

\* 인하대학교 컴퓨터 · 정보공학과 석사과정(jhcheon@dblab.inha.ac.kr)

\*\* 인하대학교 컴퓨터공학부 박사과정(yijang@dblab.inha.ac.kr)

\*\*\*인천대학교 정보기술대학 초빙교수(skcho@incheon.ac.kr)

\*\*\*\*서원대학교 컴퓨터공학부 부교수(sjlee@sewon.ac.kr)

\*\*\* 교신저자 인하대학교 컴퓨터공학부 교수(hybae@dblab.inha.ac.kr)

In this paper, hierarchical reorganization method is presented for dynamic reorganization in a grid database. When the reorganization is conducted, an organizer is created to collect the information of databases. In addition, the organizer which is constructed by the hierarchical structure supports information communication and reorganization, and then it allows the support of regional reorganization operation and effective balance control. The performance assessment of the proposed method shows that the processing capacity is increased after the reorganization.

**주요어:** 그리드 데이터베이스, 계층구조, 재조직, GIS, 공간 데이터

**Keywords:** grid database, hierarchy structure, reorganization, GIS, spatial data

## 1. 서론

최근 급속한 정보통신기술의 발전과 다양한 사용자의 요구에 의해서 공간정보 관리 시스템은 전통적인 GIS의 응용에서 지리정보와 비즈니스 프로세서가 통합된 새로운 환경으로 발전하고 있다. 공간정보에 대한 활용은 기업의 영업 또는 판매현장에서의 각종 위치정보와 관련된 고객, 경쟁사 데이터를 수집하여 이를 지도상에 시각적으로 표시함으로써 경영 및 마케팅 전략을 효과적으로 수립하는 LBS, 택배 및 물류, gCRM, gMining, 상권 분석 관리, 교통 정보, 여행 정보, 금융 분야 등으로 확산되고 있다. 이러한 환경에서 다양한 분산 데이터의 관리를 위한 그리드 데이터베이스가 최적의 시스템으로 적용이 가능하다[16].

그리드 데이터베이스는 그리드 컴퓨팅 환경에서 분산된 데이터의 효율적 처리와 사용을 위한 데이터베이스 관리 시스템이다[2, 3, 6, 8]. 이 시스템은 공간 데이터와 같은 대용량 자원을 공유하며 고속 연산 및 다양한 응용기기를 지원을 한다. 그러나 시스템의 효율적 운영을 위한 재조직이 발생하였을 때 기존의 분산 데이터베이스처럼 중앙 집중식 관리자가 존재하지 않기 때문에 시스템의 불균형적 운영이 발생할 수 있다. 분산 데이터베이스의 중앙집중식의 관리자는 전체적인 시스템을 관리 할 수 있다[10, 12]. 이 중앙집중식 관리자는 임의의 데이터베이스의 과부하로 인한 성능저

하를 막기 위하여 데이터베이스의 재조직에 직접 관여를 한다. 반면에 그리드 데이터베이스에서는 이러한 중앙 집중식 관리자가 없는 대신 지역적 자율성을 제공하여 상황에 따른 자체 처리가 가능해야 한다[6, 8]. 중앙집중식 관리자가 있는 분산 데이터베이스는 데이터의 상황 및 데이터베이스의 분산 된 상황 시스템의 효율성을 중앙에서 관리하기 때문에 시스템을 균형적으로 운영할 수 있지만, 그리드 데이터베이스에서는 지역적 상황에 따라서 개별적으로 재조직 활동이 이루어 진다. 이 때문에 분산 데이터베이스의 균형적인 시스템 운영에 비하여 불필요한 재조직으로 인한 불균형적 운영이 발생한다. 비교적 데이터의 크기가 큰 공간 데이터의 경우 불필요한 재조직이 발생 할 경우 시스템의 성능저하를 초래한다.

본 논문에서는 그리드 데이터베이스에서 계층 구조를 이용한 동적 재조직 기법을 제안한다. 제안기법은 임의의 데이터베이스에서 이벤트 발생시 이벤트에 관련된 데이터베이스들을 계층적으로 구성하여 균형적 재조직을 수행 한다. 오가나이저의 계층적 구조에서 요약 정보를 사용함으로써 최상위 오가나이저가 재조직 수행계획 수립 시 부담해야 하는 작업부하를 하위 오가나이저로 작업 부하를 나누면서 빠른 재조직 수행을 할 수 있다. 이 동적인 연결 구조는 이벤트 발생시 오가나이저를 생성하여 재조직에 관여함으로써 지역적인 재조직 연산의 지원 및 효율적인 균형 조정이 가능

해진다. 또한 동적인 오가나이저에 의한 재조직 수행으로서 불필요한 재조직 연산의 제거가 가능해진다. 따라서 동적으로 변화되는 환경에 유동적인 대응으로 효율적인 재조직 연산이 가능해 진다.

본 논문의 구성은 다음과 같다. 2장은 관련 연구로 분산 환경의 데이터베이스 시스템을 기술하고, 이 시스템들의 성능을 향상시키기 위한 기법들을 기술한다. 3장에서는 본 논문에서 제안하는 그리드 데이터베이스에서 계층 구조를 이용한 동적 재조직 기법에 대하여 동적인 연결 방법과 이 기법에 사용되는 자료구조, 그리고 재조직 방법에 대하여 기술한다. 4장에서는 제안한 시스템의 성능평가를 하고, 마지막으로 5장에서 결론을 기술한다.

## 2. 관련 연구

본 장에서는 분산 데이터베이스 시스템의 특징에 대해서 기술한다. 그리고, 분산 데이터베이스의 성능 향상을 위한 재조직 기법을 기술한다.

### 2.1 분산 데이터베이스와 그리드 데이터베이스

기하 급수적으로 늘어가는 방대한 데이터들의 효율적 성능향상을 위한 노력은 지속적으로 이루어져 왔다. 이러한 노력의 일환으로 분산 데이터베이스와 데이터베이스 클러스터 그리고 그리드 데이터베이스에 대한 연구가 계속 되었다.

분산 데이터베이스는 컴퓨터 통신망을 이용하여 여러 개의 지역 데이터베이스를 논리적으로 연관시킨 통합된 데이터베이스이다. 물리적으로는 분산되고 논리적으로는 집중되어 있는 형태의 구성으로 단순한 연결이 아닌 각 데이터베이스가 서로 관여를 하는 연결구조이다. 분산 데이터베이스의 장점은 데이터를 분

산 배치하므로 장애에 대한 대비에 강하고 다수의 이용자가 대규모의 데이터베이스를 낮은 비용으로 공유할 수 있는 점이다. 분산 데이터베이스는 중앙 집중형 데이터베이스 보다 저비용으로 구성이 가능하며, 확장성 및 가용성에 장점이 있다[10].

데이터베이스 클러스터는 한 지역의 같은 구역에서 물리적으로 연결이 되어있는 구조이다. 분산 데이터베이스가 데이터의 공유, 유통 및 투명성에 초점이 맞추어 있다면, 데이터베이스 클러스터는 고속의 네트워크 연결하여 데이터 처리의 성능성, 신뢰성, 분산성 및 가용성을 향상 시키는데 목적이 있다[11, 12, 14]. 그러나 이 시스템은 물리적인 근접성으로 인하여 확장하는데 한계가 있다.

그리드 데이터베이스는 기존에 제시되었던 데이터베이스 시스템보다 더 많은 것을 지원한다. 이 시스템의 초점은 대용량 자원의 공유와 고속연산 및 다양한 응용기기의 지원에 맞추어져 있다[2, 3, 6, 8]. 그리드 데이터베이스의 본질적인 기술로서 자원을 공유하고 높은 자원 활용도를 가지는 자원 가상화 기술, 리소스들이 그리드 상에서 제공되거나 제거되는 동적인 관리 기술, 서비스 지향구조를 가지는 웹서비스 기술이 있다. 특히 자원 가상화 기술은 스토리지, 서버 및 네트워크 도메인에 이르기까지 물리적인 자원을 가상적으로 묶어 자원이 동적으로 추가, 삭제, 변경될 수 있도록 하는 기술이다.

그리드 데이터베이스가 가지고 있는 이 기종의 통합 된 서비스, 확장성, 보안성, 적합성 및 결합 허용 등 기술적 특성들을 접목시켜 유비쿼터스 컴퓨팅에 관한 많은 문제들을 해결 할 수 있다[9]. 최근 유비쿼터스 시대에 맞추어 u-Korea, u-City 등 다양한 플랫폼이 제시되고 있다[13, 15]. 언제, 어디서, 어떤 장치를 사용하더라도 정보의 검색이나 접근이 가능해지는 유비쿼터스 환경에서 LBS, 텔레매티스 등 공간 정보를 이용한 연구들이 요구되

고 있다. 특히 텔레매틱스는 IT839의 9대 신성장동력의 하나로서 이동통신 기술, 자동차 제어기술, 차량의 위치 추적 및 실시간 교통 정보 제공을 위한 측위기술 안전한 접속을 위한 음성인식 기술 등 여러 기술이 종합적으로 관련되어있는 기술이다[15]. 이러한 유비쿼터스 환경에서 공간 정보를 활용하는 것이 필수적인 요소로 작용이 되며, 유비쿼터스 컴퓨팅 환경에 적합한 그리드 데이터베이스가 가장 적합한 환경이다.

## 2.2 재조직 기법

분산 환경에서 데이터베이스의 성능향상을 위한 여러 재조직 기법들이 연구되고 있다. 대표적인 재조직 기법으로는 온라인 재조직 기법, B+트리 색인의 온라인 재조직 기법 및 온라인 확장 기법 등이 있다.

분산 환경 데이터베이스에서 특정한 데이터만 검색하는 경우나 특정 노드의 데이터삽입이 집중되는 경우, 해당 노드의 과부하 또는 불균형 상태가 발생한다. 따라서, 사용자의 질의 요청에 대한 빠른 응답 속도와 가용성을 높이기 위해 데이터의 복제 및 분할 등 온라인 재조직 기법을 사용한다[1, 14].

그러나 동시에 두개 이상의 다중 노드에 과부하가 발생된 경우, 부하 분배를 위해 인접 노드로 여러 번의 반복된 데이터 이동이 발생되고, 재조직 수행 동안 시스템의 응답 속도가 늦어지는 문제점이 발생한다.

다중 노드에 발생한 과부하 문제를 빠르고 효율적으로 해결하는 B+트리 색인의 온라인 재조직 기법이 연구 되었다[1, 7]. 이 기법은 데이터에 대한 온라인 재조직이 결정될 경우 인접한 노드 중 부하가 적은 노드를 목적 노드(Target Node)로 선택하고 과부하가 발생된 원본 노드(Source Node)의  $B^+$  트리 색인 중에서 이동할 데이터의 부분

을 결정하고 데이터 이동을 수행하는 기법이다.

또한, 사용자 질의의 급격한 증가에도 안정된 서비스를 제공하기 위해 실시간 트랜잭션의 처리를 중지하지 않고 데이터베이스의 확장을 수행하는 온라인 확장 기법이 연구 되었다[12, 14]. 이 기법은 시스템의 가용성을 높이기 위한 방법이다. 이 시스템은 실시간 클라이언트 질의를 수행하고 있는 노드를 활동 노드(Active Node)라고 시스템의 확장을 위해서 사전에 등록되어 있는 노드를 예비 노드(Spare Node)라한다. 활동 노드 중에서 특정 노드가 고장을 일으킨 경우나 부하가 집중되는 경우, 예비 노드를 활성화 상태로 만들고 예비 노드에 기존의 분할 및 복제되어 있던 데이터를 온라인 재조직을 수행 함으로써 전체 시스템의 가용성을 높이고 병목현상을 제거하여 전체 트랜잭션 처리량을 증가시키고 평균 응답 시간을 줄이도록 하는 것이다.

## 3. 그리드 데이터베이스에서 계층 구조를 이용한 동적 재조직 기법

본 장에서는 그리드 데이터베이스에서 재조직 연산 시 불필요한 연산을 줄이고 균형적인 시스템 운영을 위하여 계층 구조를 이용한 동적 재조직 기법을 제안한다. 제안 기법은 이벤트가 발생하였을 때 재조직 연산을 위하여 오가나이저를 생성하고, 데이터베이스와 오가나이저 또는 오가나이저와 오가나이저 사이에 논리적으로 계층적 구조로 연결하여 재조직을 수행하는 기법이다.

### 3.1 데이터베이스 재조직을 위한 기본 연산

분산 데이터베이스, 데이터베이스 클러스터, 그리고 그리드 데이터베이스 등의 데이터를 분산하여 관리하는 시스템에서는 사용자의

질의 형태와 데이터의 접근 빈도, 접근 경로, 데이터 크기에 따른 데이터베이스 구성의 변화가 수시로 일어나게 된다. 이러한 환경에서 데이터베이스 관리자(DBA)는 수시로 데이터베이스의 상태를 파악하고 구성을 변경하여 항상 최적의 성능을 유지해야 한다.

일반적인 재조직 방법은 크게 데이터의 분할 및 복제를 통해 이루어진다. 복제는 원본 데이터로부터 생성되는 뷰, 스냅샷 또는 가상 테이블 정의 등을 통한 데이터의 복사본이다. 분할은 관계 데이터베이스에서 릴레이션의 스키마 변경을 통한 수직 분할과 데이터 범주에 따른 분리 과정인 수평 분할로 나뉜다. 재조직 방법은 시스템 환경에 따라 적용 방법에 차이가 있다. 또한, 재조직은 사용자 질의 패턴과 노드의 자원 배분 상태 등의 운영 환경을 고려하여야 한다.

일반적인 재조직 과정을 공통된 부분으로 정리하여 분류하면 복제, 이동, 분할 및 병합의 4가지로 분류된다. 본 논문에서는 이 4가지 과정을 데이터베이스 재조직을 위한 기본 과정으로 정의하고, 그 내용은 다음과 같다.

#### • 복제 과정

데이터베이스의 작업부하가 과도하게 발생하였을 때, 또는 빈번하게 다른 데이터베이스로부터 특정한 데이터를 요구할 경우 복제연산이 발생한다. 복제연산은 임의의 노드 내의 읽기 연산이 빈번한 데이터에 대한 복제본을 다른 노드에 생성하는 것이다. 데이터를 복제함으로써 사용자의 접근을 분산시킴으로써 데이터베이스의 작업부하를 나누게 한다. 또한, 공간 데이터같이 거의 업데이트가 이루어지지 않는 데이터들은 복제를 함으로써 네트워크 비용을 줄임으로써 성능향상을 얻을 수 있다.

#### • 이동 과정

임의의 노드에 의한 특정 데이터로의 접근이 빈번한 경우, 또는 임의의 노드에서 쓰기 연산이 빈번하게 발생 할 경우 이동 연산이 발생한다. 이동 연산은 임의의 노드로 해당 데이터를 이동함으로써 응답속도를 향상시키는데 목적이 있다.

<표 1> 복제 과정의 단계별 처리내용

연산 단계별 처리 내용	
발생 시점	<ul style="list-style-type: none"> <li>- 데이터베이스의 작업 부하(CPU, 평균 TPS, 평균 처리량 등)가 임계 값을 벗어난 경우</li> <li>- 임의 데이터에 대한 읽기 연산이 빈번한 경우</li> <li>- 임의 데이터에 대한 특정 노드로부터의 접근(ex: 빈번한 조인 연산 등)이 빈번한 경우</li> </ul>
연산 과정	<ul style="list-style-type: none"> <li>- 원본 데이터와 동일한 복제본을 목적 노드에 생성</li> </ul>
목적 노드의 선정	<ul style="list-style-type: none"> <li>- 작업 부하가 크거나 읽기 연산이 빈번한 경우 주변의 노드들 중 가장 사용률이 낮은 노드를 선택</li> <li>- 특정 노드로부터의 접근이 빈번한 경우 해당 노드를 목적 노드로 선정하지만, 목적 노드의 공간 및 부하 정도를 고려하여 가장 가까운 다른 노드 선택이 가능</li> </ul>

<표 2> 이동 과정의 단계별 처리내용

연산 단계별 처리 내용	
발생 시점	<ul style="list-style-type: none"> <li>- 임의의 노드에 의한 특정 데이터의 쓰기 연산이 빈번한 경우</li> </ul>
연산 과정	<ul style="list-style-type: none"> <li>- 원본 데이터를 목적 노드로 복제 후 원본 데이터 삭제</li> </ul>
목적 노드의 선정	<ul style="list-style-type: none"> <li>- 복제 과정에서와 같이 특정 노드로부터의 접근이 빈번한 경우 해당 노드와 그 주변 노드를 후보로 하여 목적 노드를 선정</li> </ul>

〈표 3〉 분할 과정의 단계별 처리내용

연산 단계별 처리 내용	
발생 시점	<ul style="list-style-type: none"> <li>- 데이터베이스의 작업 부하(CPU, 평균 TPS, 평균 처리량 등)가 임계 값을 벗어난 경우</li> <li>- 임의 데이터에 대한 쓰기 연산이 빈번한 경우</li> </ul>
연산 과정	<ul style="list-style-type: none"> <li>- 원본 데이터의 일부를 목적 노드 또는 그룹으로 복제</li> <li>- 원본 데이터의 전송 영역 삭제</li> <li>- 원본 데이터와 일부 복제본에 대한 스키마 조정</li> </ul>
목적 노드의 선정	<ul style="list-style-type: none"> <li>- 원본 데이터를 포함하는 노드를 제외한 다른 노드들 중 후보 그룹을 선정</li> <li>- 후보 그룹들 중에서 최적의 노드를 목적 노드들로 선정</li> </ul>

〈표 4〉 병합 과정의 단계별 처리내용

연산 단계별 처리 내용	
발생 시점	<ul style="list-style-type: none"> <li>- 분할 데이터 간의 조인 연산이 빈번히 발생하는 경우</li> </ul>
연산 과정	<ul style="list-style-type: none"> <li>- 임의의 분할 데이터를 목적 노드 또는 그룹으로 복제</li> <li>- 원본 데이터의 전송 영역 삭제</li> <li>- 각각의 데이터를 병합한 상태로 스키마 조정</li> </ul>
목적 노드의 선정	<ul style="list-style-type: none"> <li>- 각각의 분할 데이터가 위치한 노드들 중 노드의 부하가 적은 그룹을 목적 노드 또는 그룹으로 선정</li> </ul>

#### • 분할 과정

데이터에 대한 쓰기 연산이 빈번하거나 접근 빈도가 높아 시스템의 부하가 크지만, 이동 과정에서 와는 달리 접근 경로가 다양하여 목적 노드를 결정하기 어려운 경우 분할 연산을 수행한다. 분할 과정은 모든 복제본을 분할하여 새로운 데이터 그룹을 생성한다. 이를 통해 실질적인 작업 부하의 분산을 유도한다.

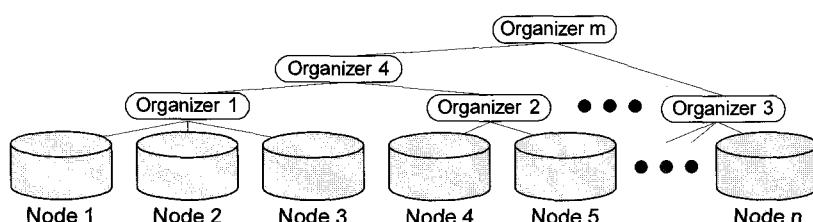
#### • 병합 과정

기존에 분할된 데이터 들 간에 조인연산에 의한 비용이 각각의 데이터에 대한 단일 접근 비용보다 클 경우 병합 연산을 수행한다. 분할

과정의 반대되는 과정으로서 사용자 질의의 접근 패턴이 바뀌어 기존의 분할 상태가 오히려 성능을 낮추는 경우 시행된다.

### 3.2 재조직을 위한 계층 구조

계층 구조를 구성하는 각각의 노드의 구성은 <그림1>과 같다. 최하위 단말 노드들은 그리드 데이터베이스를 구성하는 시스템들이며, 노드로 지칭한다. 중간 노드들과 최상위 노드는 모두 오가나이저(Organizer)로 지칭된다. 각각은 단말 노드에 존재하는 프로세스 형태로 동작되며, 계층 구조는 이들에 대한 논리적



〈그림 1〉 재조직을 위한 계층 구조

&lt;표 5&gt; 이벤트 정보 구조

필드	설명
Type	이벤트 타입 (REPL, MOVE, FRAG, MERG)
DataName	데이터의 고유 이름
SrcCnt	원본 노드 개수
SrcList	원본 데이터를 갖는 노드 또는 오가나이저 리스트
DestCnt	목적 노드 개수
DestList	재조직 목적 노드 또는 오가나이저 리스트

연결 구조를 나타낸다.

오가나이저는 이벤트가 발생한 데이터베이스에서 오가나이저 프로세스가 실행하면서 생성이 된다. 오가나이저는 정해진 수 이내의 노드 또는 오가나이저를 자식으로 포함한다. 하나의 오가나이저로 구성되는 그룹은 주로 재조직 연산 시 포함되는 노드들로 구성될 수 있다. 또한, 데이터베이스 또는 네트워크의 구조에 따라 구성 형태가 달라질 수 있다. 상위 계층으로 전달되는 재조직 관련 정보는 다음 절에서 설명한다.

### 3.3 이벤트 정보의 구조

이벤트 발생시 데이터베이스는 오가나이저를 형성하면서 다음의 <표5>와 같은 구조의 자료를 생성한다.

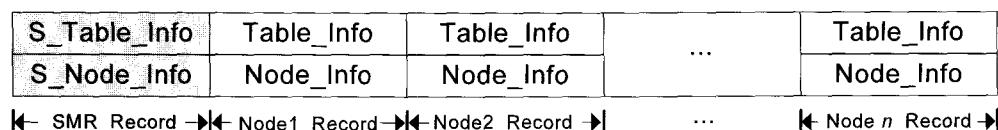
이벤트 정보는 데이터베이스에 이미 정의된 임계 값을 기준으로 그 한계를 초과하는 경우 생성된다. 데이터 구조는 재조직을 수행하기 위한 이벤트 타입, 이벤트가 발생한 노드 또는 오가나이저의 이름, 재조직할 데이터의 이름, 원본 노드 및 목적 노드 리스트, 그리고 데이터 크기를 포함한다.

Type은 이벤트 발생시 조건에 따라 앞서 정의한 4가지 재조직 과정 중 하나를 지정한다. DataName은 재조직의 대상이 되는 원본 데이터 이름이다. SrcList는 원본 데이터를 포함하는 노드 리스트 또는 오가나이저 리스트가 된다. 이는 계층구조에서 이벤트의 전달 위치에 따라 하위 그룹의 경우 단말 노드에 대한 리스트를 포함하고, 상위 그룹의 경우 중간 계층의 오가나이저 리스트를 포함하게 된다. DestList는 목적 노드에 대한 리스트로서 SrcList와 같이 노드 또는 오가나이저 리스트를 포함한다. SrcCnt와 DestCnt는 각각의 노드 개수를 나타내며, 이벤트 정보 전달 시 생성되는 패킷 구성 정보로 활용된다.

### 3.4 요약 정보의 구조

이벤트 정보와는 별개로 제안기법은 데이터베이스의 상태 정보를 상위 오가나이저로 전달한다. 오가나이저가 취합하는 자료구조는 <그림2>와 같다.

<그림1>과 같이 오가나이저는 논리적으로 연결된 모든 데이터베이스들로부터 <그림2>와 같은 오가나이저의 자료구조에 재조직 데



&lt;그림 2&gt; 오가나이저에서 취합하는 자료구조

〈표 6〉 단말 노드의 시스템 정보

필 드	설 명
Loc	노드 또는 오가나이저 위치
Name	데이터의 고유 이름
RES	평균 응답 속도
RES_D	평균 응답 속도 편차
TPS	단위 시간당 트랜잭션 처리량
TPS_D	그룹 내 TPS 편차
CPU_Rate	현재 데이터베이스 CPU 사용률
CPU_D	그룹 내 CPU 사용 편차
Connect_Client_Cnt	데이터베이스에 접속한 클라이언트 수
Connect_Client_Cnt_D	그룹 내 접근한 클라이언트 편차
DB_Total_Size	그룹 내 총 데이터베이스 크기
DB_Available_Size	그룹 내 이용 가능한 총 데이터베이스 크기
DB_Max_Size	사용 가능한 DB 중 최대 DB 크기

이터 정보 및 시스템 정보를 받는다. <그림2>의 Table\_Info는 재조직 데이터에 관한 정보로서 원본 노드에서 데이터를 전달하고, 목적 노드에서는 비어있는 정보로 전달한다. Table\_Info에서는 재조직 데이터의 읽기 연산 횟수, 쓰기 연산 횟수 등의 정보를 나타낸다. Node\_Info에서는 <표6>과 같이 시스템 정보를 나타낸다. 이 정보는 데이터 단위로 구성되며 노드의 종류에 상관없이 시스템 정보를 오가나이저에 전달한다.

<그림2>에서 SMRRecord는 요약 데이터로서 오가나이저가 받은 모든 데이터들을 취합하여 요약된 형태의 정보이다. 계층적 구조가 형성되어 상위 오가나이저에 정보를 전달할 때 이 요약 정보를 상위 오가나이저에 전달한다. 오가나이저는 Record의 응답속도나 TPS, CPU 사용률 및 클라이언트 접속자 수 등을 비교하여 재조직 연산 계획 수립 및 최적의 노드들을 찾게 된다.

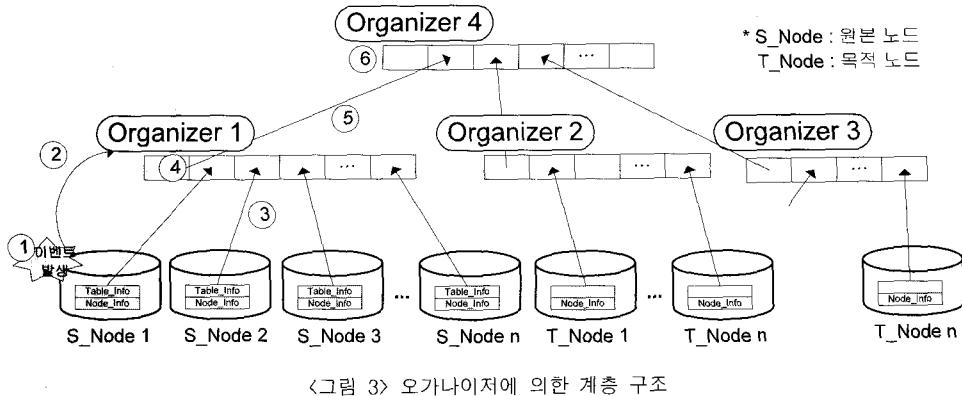
상위 계층으로 전달되는 정보는 각 데이터의 평균 정보를 포함하게 된다. 데이터베이스 정보는 상위 노드로 전달되면서 같은 데이터의 고유 이름을 가지는 정보들을 하나로 합친다. 합쳐진 정보는 그 종류에 따라 다음의 세 가지 경우로 분리된다.

- 평균: TPS, RES, CPU\_Rate 등
- 편차: TPS\_D, RES\_D, CPU\_D 등
- 합계: DB\_Total\_Size, DB\_Available\_Size 등

또한, 상위 오가나이저는 취합된 정보들을 바탕으로 자신이 속한 그룹에 대한 요약정보를 생성한다. 요약정보에는 그룹의 평균 TPS, RES, CPU\_Rate 등의 상태정보들이 포함된다. 상위 오가나이저는 이벤트 정보에 따라 목적 노드를 선정할 필요가 있을 경우 데이터베이스 정보를 사용한다. 이 때 각 정보의 특징에 따라 평균, 편차 또는 합계 정보를 이용할 수 있다. 예를 들어 상위 오가나이저가 임의의 데이터를 복제할 목적 노드를 찾는 경우 오가나이저는 먼저 TPS, RES 등의 평균 값 또는 합계 값을 통해 가장 여유가 많은 그룹을 찾는다. 해당 그룹을 찾기 어려울 때는 가장 편차가 심한 그룹을 대상으로 한다. 편차가 심한 그룹을 선택하고 재조직 연산을 수행하게 되면, 부가적으로 각 노드 간의 편차를 줄일 수 있다.

### 3.5 재조직 노드의 선정

제안 기법의 구조는 이벤트 발생시 오가나



이저를 구성하여 하위 레벨의 데이터를 취합하고, 취합된 정보로부터 요약 정보를 생성하여 상위 레벨로 전달하는 계층적 구조로 구성한다. 오가나이저는 요약 데이터를 이용한 수행으로 빠른 연산을 수행한다. <그림3>은 이벤트 발생시 오가나이저를 생성하면서 계층구조를 이루는 과정을 나타낸다.

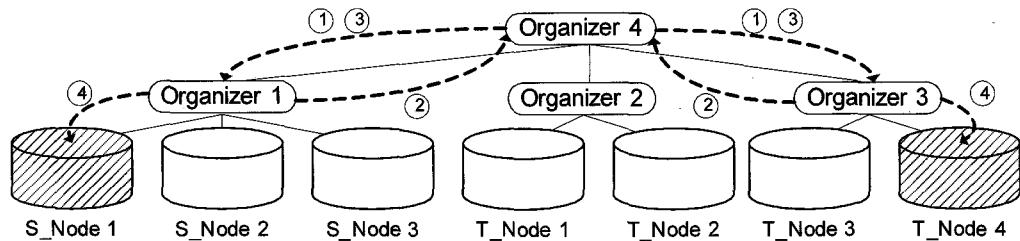
<그림3>의 S\_Node는 원본 노드를 나타내고 T\_Node는 목적 노드를 나타낸다. 재조직을 위한 계층적 구조 수립 과정은 다음과 같다.

- ① 임의의 데이터에 대한 처리 속도 저하, 사용자 폭증, 시스템 자원의 사용률이 높을 때 등의 상황에서 이벤트 발생
- ② 이벤트 노드를 중심으로 오가나이저를 생성하고 대상 데이터를 갖는 데이터베이스를 연결
- ③ 오가나이저에 연결된 노드들은 노드 정보와 데이터 정보를 함께 상위 오가나이저에게 전달
- ④ 오가나이저에서 정보를 취합하여 하나의 요약 정보를 생성
- ⑤ 상위 오가나이저가 있을 경우 ③번부터 반복
- ⑥ 최상위 오가나이저는 취합된 정보를 바탕으로 재조직 연산 계획 수립 예를 들어, 데이터에 대한 정보를 D라 하고,

노드에 대한 정보를 N이라 하고, 오가나이저1에 있는 데이터 D를 다른 오가나이저에 속하여 있는 노드에 복제한다고 가정한다. 이때, <그림3>의 ④번 수행으로 오가나이저1에서 취합한 데이터를 요약데이터 ( $D_1, N_1$ )이라 표현 할 수 있고, 오가나이저2와 오가나이저3에서 요약한 데이터는 ( $D_2, N_2$ ), ( $D_3, N_3$ )라고 할 수 있다. 그러나, 오가나이저2와 오가나이저3에서는 해당 데이터가 없으므로 노드 정보만을 최상위 오가나이저에게 전달한다. 최상위 오가나이저에서 정보들을 분석하여 최적의 하위 오가나이저를 선택하게 된다.

재조직 연산 수행 시 목적 노드가 정해져 있다면 오가나이저가 직접 대상 데이터베이스를 연결하여 재조직을 수행한다. 그러나, 목적 노드가 정해있지 않을 경우 <그림4>와 같은 순서로 재조직을 수행할 데이터베이스가 선정되고 메시지를 전달하게 된다.

아래의 예에서, 만약 오가나이저1과 오가나이저3을 선택하였을 때 최상의 오가나이저는 ( $D_1, N_2$ )의 정보를 얻을 수 있다. <그림4>의 ①, ②, ③의 과정은 오가나이저가 정확한 노드를 연결해 주기 위한 과정이다. <그림4-①>은 최상위 오가나이저가 선택한 하위 오가나이저에게 원본 노드 및 목적 노드를 선택 요청 메시지이다. 하위 오가나이저는 목적 노드를 선정해서 <그림4-②>의 과정으로 최상위



&lt;그림 4&gt; 재조직 메시지가 전달되는 순서

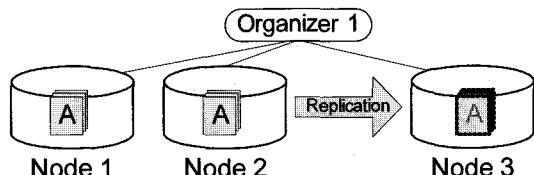
오가나이저에게 메시지를 전달한다. 최상위 노드는 하위 노드의 메시지를 받아 원본 노드에는 목적 노드 정보를, 목적 노드에는 원본 노드 정보를 <그림4-③>의 과정으로 하위 오가나이저에게 메시지를 전달한다. 최종적으로 <그림4-④>의 과정을 통해 단말 노드에 재조직 명령을 전달하면서 재조직을 수행한다.

### 3.6 계층 구조를 이용한 재조직

계층 구조는 평상시에는 구성되지 않는다. 그러나, 재조직 연산 수행 시 가장 기본적인 형태의 계층 구조를 구성한다. 이 때, 계층 구조는 원본 데이터들과 대상 테이블을 포함하는 노드들을 하나로 묶게 되는 오가나이저를 통해 구성된다. 예를 들어, <그림5>와 같이 복제 연산을 수행하기 위해서는 A테이블을 갖는 노드 1, 2와 복제의 대상이 되는 노드 3을 하

나의 오가나이저로 그룹을 구성하고, 복제 연산을 수행하게 된다.

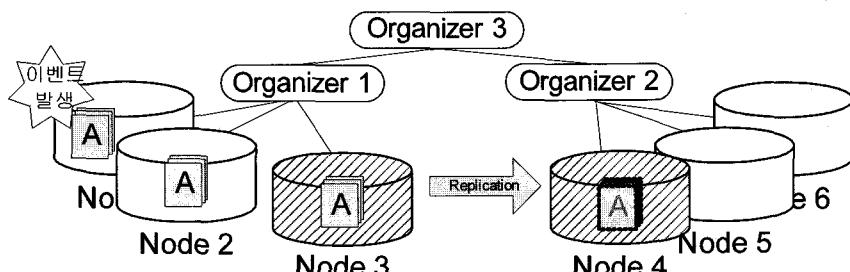
그러나, 데이터베이스에 특정한 데이터에 대한 임의의 접근으로 인한 과부하의 경우



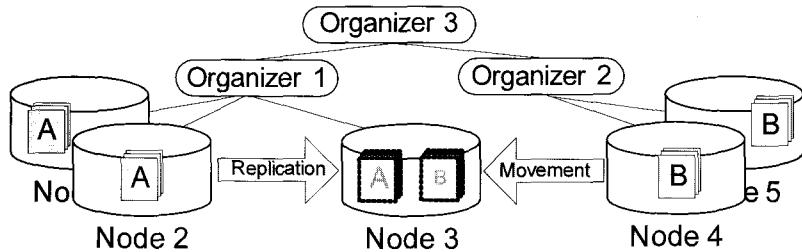
&lt;그림 5&gt; 복제 연산을 위한 계층 구조의 생성

대상이 되는 데이터베이스가 없기 때문에 대상이 되는 데이터베이스를 선정해 주어야 한다. 예를 들어 <그림6>과 같이 주위의 다른 오가나이저들과 비교를 하여 최적의 목적 노드를 찾아 복제 연산을 수행한다.

<그림6>과 같이 목적 노드의 공간 및 네트워크 부하 정도를 고려하여 가까운 오가



&lt;그림 6&gt; 다중 오가나이저에 의한 재조직



&lt;그림 7&gt; 중복된 재조직 연산에 의한 계층구조

나이저와 계층적 구조를 통하여 최적의 목적 노드를 선택 할 수 있다.

재조직 연산이 중복되어 수행되는 경우, 그룹이 중복되어 오가나이저가 두 개 이상 생기게 된다. 이 경우 <그림7>과 같이 먼저 생성된 오가나이저의 상위 계층에 새로운 오가나이저를 생성하게 된다.

상위 계층의 오가나이저는 자식 오가나이저로부터 재조직에 관련된 데이터베이스 정보를 전달받는다. 이 정보를 통해 오가나이저는 이미 수행되고 있는 기존 재조직 연산을 고려하여 새로운 재조직 연산을 수행할 수 있다.

이러한 재조직 수행 시 연결되는 동적인 계층적 구조는 재조직에 관련된 데이터베이스만 연결 함으로써 불필요한 데이터베이스 연결을 없애준다. 그리고 재조직 완료 시 오가나이저는 소멸되고 논리적 연결 구조를 해지 하면서 자원 낭비를 막는다.

#### 4. 성능 평가

본 장에서는 제안된 그리드 데이터베이스에서 계층 구조를 이용한 재조직 기법을 평가한다. 현재 그리드 데이터베이스를 평가하는 실제 모델이 없으므로, 구현 시스템에서의 제한된 프로세싱 노드 수에 제한을 받지 않는 MCC에서 개발한 CSIM언어를 이용하여 수행하였다[4, 5]. 성능 평가에 사용된 시스템 속성 및 데이터베이스 속성은 <표7>과 같다.

실험은 재조직 연산을 수행하기 이전의 트랜잭션 처리량과 재조직 수행 이후의 트랜잭션 처리량을 비교 하였다. 재조직 수행 기법으로는 중앙 집중식 기법, 인근 전파 기법, 그리고 제안 기법인 오가나이저 기법이 있으며, 이 기법들에 대한 설명은 다음과 같다. 중앙 집중식 기법은 분산 데이터베이스의 중앙 집중식 관리자 모델이다. 중앙 관리자는 모든 데이터베이스의 시스템 정보를

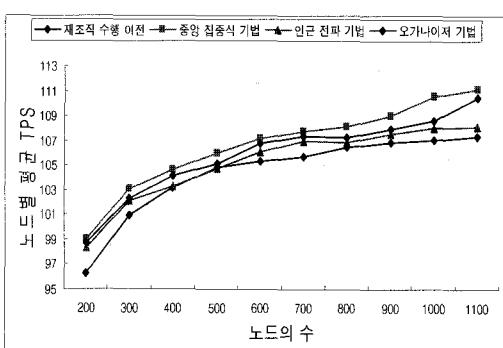
&lt;표 7&gt; 성능 평가에 사용된 시스템 환경

기 종	IBM PC 호환
CPU	Pentium IV 2.6GHz
메인 메모리	2 GB
디스크 크기	120 GB
네트워크 속도	1G bps
운영 체제	Windows XP Professional
개발 언어	Visual C++ 6.0

수시로 요청하여 시스템을 항상 체크를 한다. 따라서 재조직 이벤트가 발생시 미리 선정된 최적의 데이터베이스에 재분배를 수행하는 기법이다. 인근 전파 기법은 재조직 이벤트가 발생시 일정한 수의 인근 노드에 재조직에 관한 메시지를 전달하고, 최적의 데이터베이스를 찾아서 재조직을 수행하는 기법이다. 오가나이저 기법은 제안 기법으로써 이벤트 발생시 동적으로 오가나이저를 생성하여 재조직에 관련된 데이터베이스만 연결을 하고, 최적의 데이터베이스를 찾아 재조직을 수행하는 기법이다.

각 노드의 프로세스는 1G bps의 내부 네트워크 망을 통해 메시지와 데이터를 서로 교환 한다. 평가 환경으로 10,000명의 클라이언트가 5,000번의 질의를 수행하는 것으로 하였다.

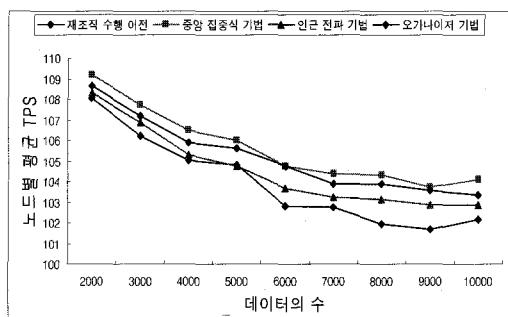
모의 실험에서 평가하는 기준은 비교 기법에 동일한 데이터 및 동일한 질의를 수행하였을 때 재조직 수행 시 질의 처리량(TPS)을 측정하는 것이다. 본 장에서 평가하는 방법은 노드의 수와 테이블의 수를 늘려가면서 제시된 기법들의 질의 처리량을 측정하는 방법으로 비교 평가한다. 노드의 수와 데이터의 수 이외의 조건에 의하여 질의 처리량이 달라지는 것을 방지하기 위해서, 사용하는 질의는 임의의 질의를 생성하여



〈그림 8〉 노드 수의 변화에 따른 질의 처리량

동일한 질의를 수행하고 테이블의 종류는 이름만 다를 뿐 모두 동일한 스키마와 크기 를 갖는 테이블을 사용한다.

〈그림8〉은 데이터의 수는 5,000개로 고정시키고 노드의 수를 200개에서 1,100개까지 점차 증가시키면서 질의 처리량을 측정한 결과이다. 이 실험은 노드의 수가 많아질 수록 재조직 수행하는 노드가 증가하고 대상이 되는 노드를 열만큼 적절하게 찾아 재조직을 수행하는지를 알아보는 실험이다.



〈그림 9〉 데이터 수의 변화에 따른 질의 처리량

〈그림9〉은 노드의 수를 500개로 고정시키고 데이터의 수를 2,000개에서 10,000개 까지 점차 증가시키면서 질의 처리량을 측정한 결과이다. 이 실험은 점차 많아지는 데이터 중에서 올바른 데이터를 찾아서 재조직 하는 것을 찾는데 있다.

위의 두 실험을 통하여 비교 모델 중에서 중앙 관리 기법이 가장 좋은 성능을 보였다. 이는 중앙 관리자가 수시로 각 데이터베이스에 시스템 정보를 요청하여 최적의 데이터베이스를 미리 파악하기 때문에 다른 비교모델보다 좋은 성능을 나타낸다. 그러나, 그리드 데이터베이스에서는 중앙 집중식의 기법 적용은 시스템의 효율적인 측면에서 적용하기 어렵다. 제안 기법은 재조직 수행 시 재조직을 수행하지 않는 경우 보다 약 10%~30% 성능 향상을 보이고 있고, 인근

전파 기법보다 약 10%~20%의 성능 향상을 보이고 있다.

## 5. 결 론

본 논문에서는 그리드 데이터베이스에서 동적인 계층적 오가나이저를 사용한 재조직 기법을 제안 하였다. 제안 기법은 이벤트 발생시 동적으로 오가나이저를 구성 함으로써 기존의 그리드 데이터베이스 환경에서 중앙 집중식 관리자의 부재로 인한 비효율적인 시스템 구성을 개선 하였다. 따라서, 공간 데이터와 같은 비교적 큰 용량의 데이터들의 불필요한 재조직 수행을 막을 수 있다. 또한 오가나이저를 계층적으로 구성하여 단계적인 정보전파 및 지역적인 재조직 연산 수행을 지원하게 되었고, 요약 정보를 사용함으로써 최상위 오가나이저에서 모든 정보를 취합 분석하는 부담을 떨어 빠른 수행이 가능해 졌다.

본 논문에서 제안한 기법은 재조직을 수행하지 않는 모델, 중앙집중식 관리자 모델, 인근 전파 기법 모델과 성능 평가를 하였다. 실험 결과와 같이 중앙집중식 관리자 모델이 가장 우수한 성능을 보였으나 단일 관리자에 대한 적용이 어려운 그리드 데이터베이스 환경에서는 재조직을 수행하지 않는 모델이나 인근 전파 기법을 사용한 모델보다 제안 기법이 우수함을 보였다.

## 참고문헌

1. C. Zou and B. Salzberg, "On-line reorganization of sparsely-populated b+ trees," Proceedings of ACM SIGMOD, 1996.
2. Foster, I. and Kesselman, C. (eds.), *The Grid: Blueprint for a New Computing Infrastructure.*, CA: Morgan Kaufmann Publishers, San Francisco, 1999.
3. F. Berman, G. Fox, and T. Hey (Eds.), *Grid Computing: Making the Global Infrastructure a Reality*, John Wiley & Sons, 2003.
4. H. Schwetman., CSIM User Guide for use with CSIM Revision 16, MCC, 1992.
5. K. Watkins., *Discrete event simulation in c.* McGraw-Hill, 1993.
6. Marla A. Nieto-Santisteban, Jim Gray, Alexander S. Szalay, James Annis, Aniruddha R. Thakar, and William J. O' Mullan, "When database System Meet Grid," *Proceedings of the CIDR Conference*, 2005.
7. Nagavamsi Ponnekanti and Hanuma Kodavalla, "Online Index Rebuild," *Proceedings of the ACM SIGMOD*, 2000.
8. N. Hong, A. Krause, S. Malaika, G. McCance, S. Laws, J. Magowan, Norman W. Paton, and Greg Riccardi, "Grid Database Service Specification," *Global Grid Forum 7*, 2003.
9. Oliver Storz, Adrian Friday, and Nigel Davies, "Towards 'Ubiquitous' Ubiquitous Computing: alliance with 'the Grid' ,," Department of Computing, Lancaster University.
10. S. Ceri and G. Pelagatti, "Distributed Databases: Principles & Systems," McGraw-Hill Company, 1984.
11. S.Ceri, G.Pelagatti and G.Martella, "Optimal File Allocation in a Computer Network: A Solution Based

- on the Knapsack Problem," Computer Networks, Vol.6, 1982, pp.345-357.
12. Svein Erik Bratsberg and Rune Humborstad, "Online Scaling in Highly Available Database," Proceedings of the 27th VLDB Conference, 2001.
13. 이근호, "U-City 비전과 서비스 시나리오 개발," 한국정보과학회 학회지, VOL.23, NO.11, 2005, pp.56-60.
14. 이충호, 장용일, 배해영, "확장 가능한 메모리상주 데이터베이스 클러스터에서 온라인 확장과 재조직 기법," 데이터베이스연구회, VOL.18, 2002.
15. 정지선, "u-Korea 구현을 위한 IT839 전략 분석," 한국전산원, 2005.
16. 황정래, 이기준, "gCRM과 공간데이터 마이닝," 공동춘계학술대회 논문집, 2002, p.38-44.

### 천종현

2005년 인하대학교 산업공학과(공학사)

2005년~현재 인하대학교 대학원

컴퓨터·정보공학과(석사과정)

관심분야: 메인 메모리 데이터베이스, 공간 데이터베이스 클러스터, 그리드 데이터베이스

### 장용일

1997년 인하대학교 전자계산공학과 (공학사)

2001년 인하대학교 컴퓨터공학부 (공학석사)

2003년~현재 인하대학교 컴퓨터공학부 박사과정

관심분야: 웹 & 모바일 GIS, 공간 데이터베이스 클러스터, 위치기반서비스, 그리드 데이터베이스

### 조숙경

1990년 인하대학교 전자계산학과(이학사)

1994년 인하대학교 전자계산공학과(공학석사)

2002년 인하대학교 전자계산공학과(공학박사)

2003년~현재 인천대학교 초빙교수

관심분야: 실시간 데이터베이스 시스템, 이동 데이터베이스 시스템, 데이터베이스 보안

### 이순조

1985년 인하대학교 전자계산학과(이학사)

1987년 인하대학교 전자계산학과(이학석사)

1995년 인하대학교 전자계산학과(공학박사)

1995년~1997년 대림대 전자 계산과 교수

1997년~현재 서원대학교 컴퓨터교육학과 조교수

관심분야: 데이터베이스, 실시간 데이터베이스 시스템, GIS, 데이터베이스 시스템의 보안

### 배해영

1974년 인하대학교 응용물리학과(공학사)

1978년 연세대학교 대학원 전자계산학과(공학석사)

1989년 숭실대학교 대학원 전자계산학과(공학박사)

1985년 Univ. of Houston 객원교수

1992년~1994년 인하대학교 전자계산소 소장

1982년~현재 인하대학교 컴퓨터공학부 교수

1999년~현재 지능형GIS연구센터 센터장

2000년~현재 중국 중경우전대학원 대학원 명예교수

2004년~현재 인하대학교 정보통신대학원 원장

관심분야: 분산 데이터베이스, 공간 데이터베이스, 지리정보 시스템, 멀티미디어 데이터베이스 등