모바일 데이터베이스 환경의 신뢰성 보장 질의처리 시스템 설계

주 해 종[†] · 박 영 배[††]

요 약

무선 네트워크(Wireless Network)의 약한 연결성 및 접속단절, 모바일 클라이언트의 이동성, 모바일 클라이언트의 휴대성으로 인해 발생하는 모바일 데이터베이스 시스템(Mobile Database System) 관련 이슈들과 이 문제들을 해결하기 위한 연구들도 한창이다. 이동 컴퓨팅은 언제 어디서나 원하는 모든 정보를 이용할 수 있는 사용자의 편의성이나 성능 면에서의 요구를 만족시키고 있지만, 데이터 관리 측면에서는 해결되어야만 하는 많은 문제점들을 안고 있다. 본 논문은 모바일 클라이언트-서버(Mobile Client-Server) 환경에서 모바일 데이터베이스 시스템 특성상 가질 수 있는 무선 네트워크의 약한 연결성 및 접속성 단절로 인한 데이터베이스 비축(Database Hoarding)과 관련된 문제, 공유 데이터(Shared Data)의 일관성(Consistency)유지 문제, 그리고 로그(Log) 최적화 문제를 해결하기 위한 모바일 질의 처리 시스템(MQPS : Mobile Query Processing System)을 설계하는데 목적이 있다. 또한, MQPS의 효율성 증명을 위해 C-I-S(Client-Intercept-Server) 모델과의 성능비교를 통해 제안한 시스템이 우수하다는 것을 입증하였다.

키워드 : 모바일 데이터베이스, 모바일 클라이언트-서버, 모바일 질의처리 시스템

Design of Reliable Query Processing System in Mobile Database Environments

Joo Hae Jong[†] · Park Young Bae[††]

ABSTRACT

Many researches are going on with regard to issues and problems related to mobile database systems, which are caused by the weak connectivity of wireless networks, the mobility and the portability of mobile clients. Mobile computing satisfies user's demands for convenience and performance to use information at any time and in any place, but it has many problems to be solved in the aspect of data management. The purpose of our study is to design Mobile Query Processing System(MQPS) to solve problems related to database hoarding, the maintenance of shared data consistency and the optimization of logging, which are caused by the weak connectivity and disconnection of wireless networks inherent in mobile database systems under mobile client-server environments. In addition, we proved the superiority of the proposed MQPS by comparing its performance to the C-I-S(Client-Intercept-Srever) model.

Key Words : Mobile Database, Mobile Client-server Model, MQPS(Mobile Query Processing System)

1. 서 론

21세기가 시작되면서 모바일(Mobile)과 무선 인터넷(Wireless Internet)은 중요한 이슈로 떠오르고 있다. 무선 기술과 정보 통신 기술의 급속한 발전으로 인해 무선 통신 기능을 장착한 휴대용 무선 단말기를 통해 언제 어디서나 원하는 정보를 억세스 할 수 있는 이동 컴퓨팅(Mobile Computing) 시대가 도래하였다. 이로 인해 이동 컴퓨터를 포함하는 분산시스템에서 데이터 관리(Data Management)가 더욱 복잡해지고 있으며, 특히 이동 컴퓨팅 환경이 가지는 모바일 클라이언트의 이동성 및 휴대성, 무선 링크 사용이라는 제약점들은 분산 환경에서의 데이터베이스 관련 문제들에 대한 해법들이 이동 컴퓨팅의 관점에서 다시 연구되어져야만 하는 원인이 되고 있다.

무선 네트워크(Wireless Network)의 약한 연결성 및 접속단절, 모바일 클라이언트의 이동성, 모바일 클라이언트의 휴대성으로 인해 발생하는 모바일 데이터베이스 시스템(Mobile Database System) 관련 이슈들과 이 문제들을 해결하기 위한 연구들도 한창이다[2, 3, 6, 13, 14]. 이동 컴퓨팅은 언제 어디서나 원하는 모든 정보를 이용할 수 있는 사용자의 편의성이나 성능 면에서의 요구를 만족시키고 있지

† 정 회 원 : 대원과학대학 멀미티디어과 조교수
†† 정 회 원 : 명지대학교 컴퓨터공학과 교수
논문접수 : 2005년 3월 7일, 심사완료 : 2005년 5월 10일

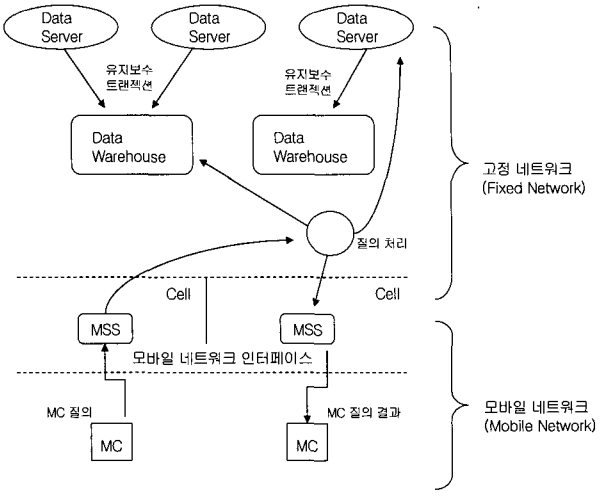
만, 데이터 관리 측면에서는 해결되어야만 하는 많은 문제점들을 안고 있다[2, 9].

본 논문은 모바일 클라이언트-서버(Mobile Client-Server) 환경에서 모바일 데이터베이스 시스템 특성상 가질 수 있는 무선 네트워크의 약한 연결성 및 접속성 단절로 인한 데이터베이스 비축(Database Hoarding)과 관련된 문제, 공유 데이터(Shared Data)의 일관성 (Consistency)유지 문제, 그리고 로그(Log) 최적화 문제를 해결하기 위한 모바일 질의 처리 시스템(MQPS : Mobile Query Processing System)을 설계하는데 목적이 있다. 이 논문의 구성은 다음과 같다. 2장에서는 본 논문의 이론적 고찰을 위해 모바일 클라이언트-서버 구조와 모바일 데이터베이스 시스템 특성에 따른 문제정의를 살펴본다. 3장에서는 모바일 데이터베이스 환경에 문제를 해결하기 위한 모바일 질의 처리 시스템 설계와 요소 정의 , 그리고 동작 원리를 설명한다. 4장에서는 모바일 질의 처리 시스템 시뮬레이션 모델의 시뮬레이션 결과를 통한 분석을 하고, 마지막으로 5장에서는 결론을 맺는다.

2. 이론적 고찰

모바일 컴퓨팅 환경은 모바일 클라이언트(MC : Mobile Client)라고 부르는 모바일 컴퓨터들과 컴퓨터들의 유선 네트워크들로 구성된다. 모바일 클라이언트는 모바일 지원 스테이션(MSS : Mibile Support Station)이라고 하는 컴퓨터를 통해 컴퓨터의 유선 네트워크와 통신을 수행한다. 각 MSS는 자신이 지원 가능한 지리적 영역, 즉 셀 내부의 모바일 클라이언트들을 관리한다.

이처럼 모바일 컴퓨팅 환경에서 질의 처리를 위한 일반적인 네트워크 구성은 (그림 1)과 같이 MSS라 불리는 고정 네트워크(Fixed Network)상에 위치한 송수신기와 이들과 무선에 의해 연결된 다수의 MC들로 구성된다. 노트북과 같이 컴퓨팅 파워나 자원면에서 데스크탑 PC에 뒤지지 않는 고성능 휴대용 컴퓨터 뿐만 아니라 팜, 포켓 PC, 인터넷 지원 셀루러 폰 등과 같은 저가의 휴대용 무선 단말기들이 모두 MC의 범주에 포함 될 수 있다.

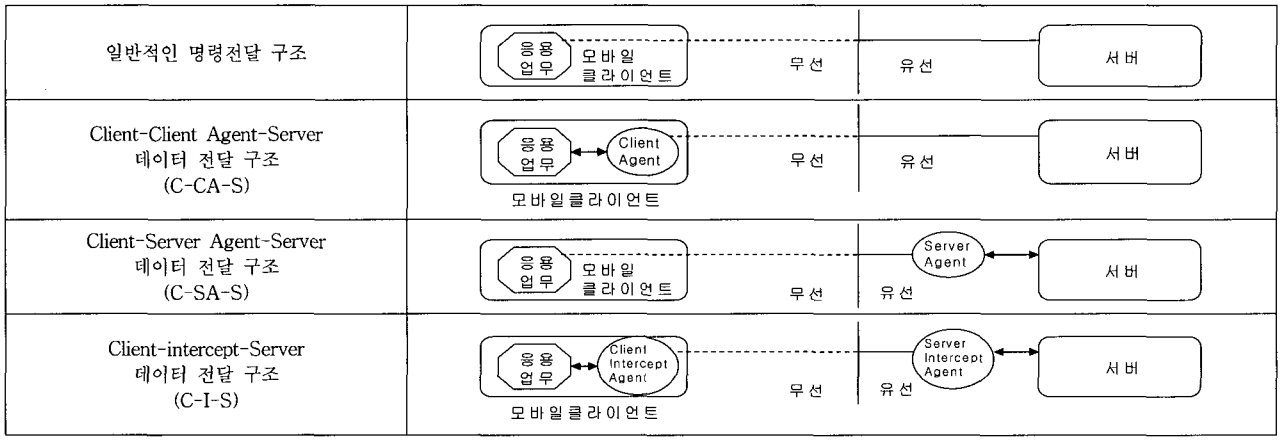


(그림 1) 모바일 컴퓨팅 환경에서의 질의 처리를 위한 네트워크 구조

2.1 모바일 클라이언트-서버 구조

일반적으로 표준화된 클라이언트-서버 모델에서, 서버는 보안과 시스템 관리를 위한 서비스 제공 측면에 중점을 두는 반면에, 클라이언트는 독립적인 서비스를 제공받는 방대한 집단의 성격을 가지고 있었다. 그러나 이러한 모델에 이동성(Mobility)이 부가되면서 클라이언트-서버간의 구별이 불분명해지게 되어서, (그림 2)와 같은 새로운 모바일 클라이언트-서버 모델(Mobile Client-Server Model)을 이끌어내게 만들었다[9, 14].

(그림 2)의 모바일 환경에서의 일반적인 명령전달 구조는 응용 클라이언트(Application Client)가 무선망을 통해 유선망상의 서버로 요청(Request)하는 모델을 나타내며, Coda[14]에 의해 소개된 C-CA-S 모델은 소규모 UNIX 파일 시스템을 가진 서버와 대규모 클라이언트에 사용되며, 이 클라이언트 에이전트(CA)는 서버에 의해서 정상적으로 수행된 파일 시스템 연산들을 모바일 클라이언트를 대신에서 수행한다. 또한 C-SA-S 모델에서 서버 에이전트(SA)는 서버와의 인터페이스 작업을 수행하는 유선망에 존재하는 프록시



(그림 2) 새로운 모바일 클라이언트-서버 모델 구조

(Proxy)와 같으며, 그러한 프록시를 서버 에이전트(SA)라고 부른다. 이와 같이 기존의 클라이언트-서버 모델의 기능을 향상시키기 위해서 클라이언트 또는 서버 측에 에이전트를 사용하였지만, 이동하는 동안 클라이언트 에이전트(CA)와 동작을 할 수 있도록 서버 또는 클라이언트 응용(Application)을 수정하기에 항상 수월한 것만 아니다. 이러한 난제를 해결하기 위해 클라이언트와 서버 양쪽에 에이전트를 두어, 쌍방의 에이전트간에 무선망과 유선망의 모든 통신을 담당하도록 한 것이 바로 C-I-S 모델이다.

2.2 모바일 데이터베이스 시스템 특성

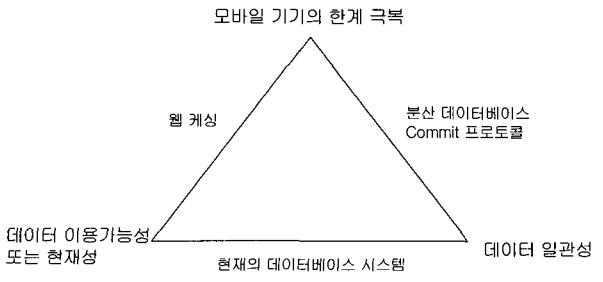
모바일 데이터베이스(Mobile Database)는 이동 컴퓨팅 기기에 사용되는 데이터베이스를 말한다. 이러한 모바일 데이터베이스는 어디에서나 데이터베이스를 사용할 수 있고, 기존의 데이터베이스 또는 모바일 데이터베이스 간의 복제 및 동기화를 강조하며, 응용과 결합되어 모바일 기기에 탑재할 수 있는 내장형(Embedded)이라는 특성을 가진다[1, 2, 6].

(그림 3)과 같이 모바일 데이터베이스 환경에 적합한 데이터베이스 서비스를 설계할 때, 세가지 요소를 고려해야 한다[14, 15]. 즉 첫째, 본질적으로 모바일 기기가 지니고 있는 무선망에서의 한계를 극복해야 하며 둘째, 특정 데이터베이스 응용에서 해당 데이터의 이용가능성 또는 현재성(Currency)이 보장되어야 하고 셋째, 모바일 기기에 표시되는 데이터는 데이터 서버(Data Server) 또는 데이터 웨어하우즈(Data Warehouse)로부터 사용되는 데이터와 일관성(Consistency)을 유지해야 한다는 사실이다.

2.3 모바일 데이터베이스 환경의 문제 정의

모바일 환경은 유선에 비해 저속이며 고가인 무선망을 사용하므로 모바일 서비스를 위한 연결성은 비용, 대역폭, 신뢰도 면에서 계속 변화하는 특징을 가진다. 또한, 모바일 환경에서 무선망은 짧은 순간동안 자주 끊어지며 연결 강도도 변화하는 약한 연결성을 가지고 있다. 모바일 서비스 환경은 약한 연결성으로 인해 유선 환경과 같이 연결 상태가 정상적이고 안정적인 경우에는 나타나지 않는 문제들이 발생하므로, 모바일 환경에서의 데이터베이스 연산은 (그림 3)의 특성들을 고려하여 네트워크의 연결 상태에 적응할 수 있어야 한다[2, 6].

접속 단절(Disconnection)은 연결성 자체가 결여된 약한



(그림 3) 모바일 데이터베이스 환경을 위한 세가지 고려 요소

연결 상태의 극단적인 경우라고 할 수 있다. 모바일 환경에서는 모바일 클라이언트(CA)가 무선 통신 서비스 범위를 벗어나는 경우와 같은 강제적인 접속 단절 이외에도 사용자가 통신 비용이나 모바일 클라이언트(CA)의 전력 소비, 무선 대역폭 등을 절약하기 위하여 의도적으로 접속 단절 상태로 들어가는 자발적인 접속 단절이 있을 수 있다[1, 2, 9].

모바일 환경의 약한 연결성과 접속 단절에 때문에 데이터를 미리 받아두는 데이터 비축(Data Hoarding)에 관련된 문제, 공유 데이터의 일관성 유지 문제, 그리고 단절 후 재접속시에 효율적인 재통합(Reintegration)이 가능하도록 어떤 정보를 보관할 것인가 하는 로그(Log) 최적화 문제가 발생할 수 있다[5, 7, 13].

이러한 문제 해결을 위해 (그림 3)의 세가지 요소를 고려하여 대용량의 데이터 이용 가능성 또는 현실성, 데이터 일관성 유지를 보장하는 모바일 클라이언트-서버 데이터베이스 응용에 적합한 데이터베이스를 생성하고 관리할 수 있도록 하는 것은 매우 중대한 일이 아닐 수 없다. 따라서 모바일 환경의 한계를 극복하고 유연한 데이터 현재성을 보장하며 데이터의 일관성을 유지시켜, 모바일 클라이언트-서버 데이터베이스 응용에 적합한 데이터 서비스를 제공하기 위한 신뢰성 보장 모바일 질의 처리 시스템(MQPS)을 설계하는 데 본 논문에 목적이 있다.

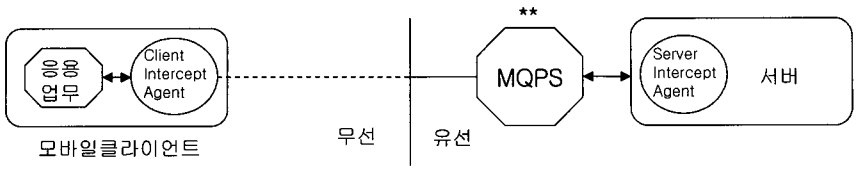
3. 모바일 질의처리 시스템(MQPS) 설계

모바일 데이터베이스 환경을 위해 (그림 2)에서 제시된 모바일 클라이언트-서버 모델들은 무선망의 약한 연결성과 접속 단절의 문제점을 해결하여 대용량의 데이터 처리를 위한 모바일 기기의 한계 극복과 현실 데이터의 이용가능성, 그리고 사용 데이터의 일관성 유지를 하려고 하였으나 완벽하게 이러한 문제점 해결을 달성할 수 있는 방안은 아니다. 무선망에서는 약한 연결성으로 인해 유선망 환경과 같이 연결 상태가 정상적이고 안정적인 경우에는 나타나지 않는 문제들이 발생하므로 무선망 환경에서의 모바일 데이터베이스 연산은 이러한 특성들을 고려하여 네트워크의 연결 상태에 적응 할 수 있어야만 한다.

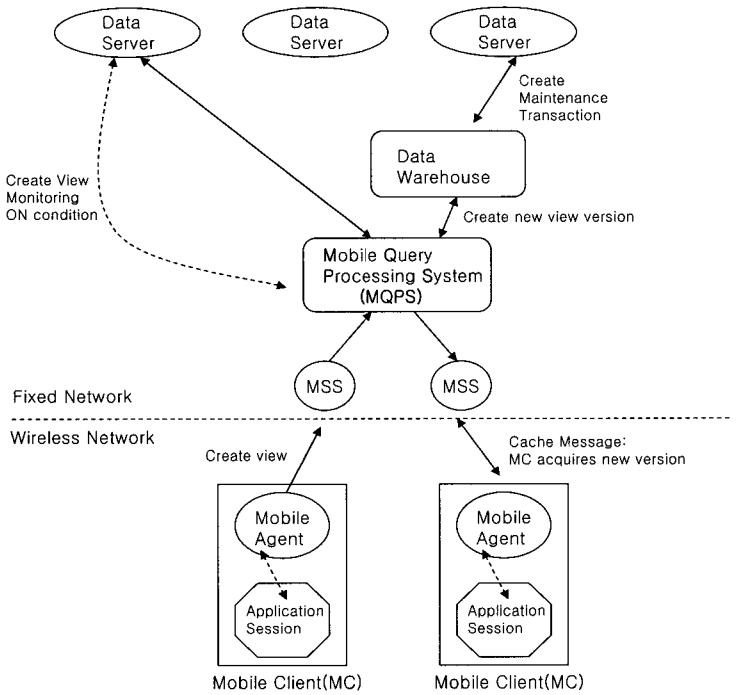
이 논문에서는 이러한 단점을 보완하기 위하여 클라이언트와 서버 양쪽에 에이전트를 가지는 시스템(C-I-S)이외에 모바일 클라이언트 지향적인 데이터 웨어하우즈(Mobile Client-Oriented Data Warehouse) 기능을 가진 추가적인 개체로 모바일 질의처리 시스템(MQPS : Mobile Query Processing System))을 (그림 4)와 같이 제안하고자 한다.

3.1 요소 정의

MQPS는 C-I-S 모바일 컴퓨팅 모델에서 접속 단절된 모바일 데이터베이스 연산을 지원하기 위하여 사용되었던 뷰(Materialized Views)의 버전(Version)을 관리하도록 클라이언트 에이전트(Client Intercept Agent)와 서버 에이전트(Server Intercept Agent) 사이에 탑재하였다.



(그림 4) MQPS를 포함하는 모바일 클라이언트-서버 구조



(그림 5) MQPS 구성도

MQPS는 접속 단절이 발생하면 사용하였던 뷰(Materialized Views)에서 발생한 데이터 변화를 보관하고, 뷰의 복잡한 버전을 유지 관리하도록 다음과 같은 구성요소를 가지고 있어야 한다.

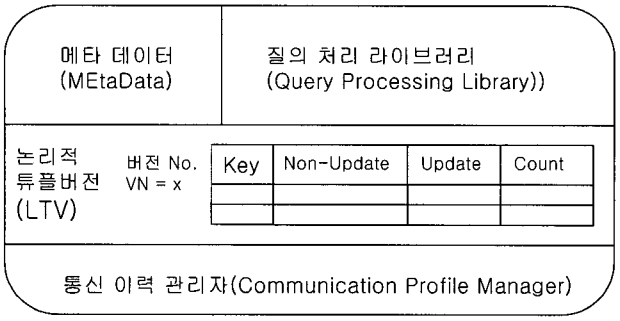
- 메타 데이터(MetaData) : DBMS 에이전트가 접근하여 사용하는 시스템 데이터베이스로서 데이베이스에 포함되는 모든 데이터 객체들에 대한 저의나 명세에 대한 정보를 유지하는 데이터베이스 이다. 추가적으로 질의 응답에 사용될 사본 데이터 사이트, 타임 스탬프, 테이블의 사이즈 정보 등을 포함하고 있다.
- 질의 처리 라이브러리 : 모바일 클라이언트의 요청을 수행하기 위한 DBMS 에이전트가 각 데이터 소스를 처리하는데 필요한 SQL 질의와 ON 조건식을 여러 하위 질의(Subquery)와 서브 조건(Subcondition)으로 분해하여 분산 질의 처리를 수행하는 루틴들의 집합으로 구성된다.
- 논리적 튜플 버전(LTV) : MQPS 에이전트에 의해서 관리되는 뷰의 버전은 LTV라고 하는 논리적인 데이터 구조로 표현된다. LTV는 키(Key), 비갱신(Non-Update), 갱신(Update), 빈도수(Count) 필드로 구성되는 관계형 스키마 구조로 구성된다.
- 통신 이력 관리자 : 모바일 에이전트에게 질의 응답의 유효한 변화에 대해 결정할 수 있도록 응용 이동성 인식(Application-Mobility-Aware)을 허용하고 피드백을 제동하는 일을 수행한다.

이러한 요소들을 포함하는 MQPS의 전체적인 시스템 구성도는 (그림 5)와 같다.

3.2 MQPS의 구조 및 처리과정

제안된 (그림 5)의 모바일 질의 처리 시스템(MQPS)의 전체 구조를 보면, 유선망(Fixed Network)에서 데이터 서버와 데이터 웨어하우즈가 사용했던 뷰(Materialized View)를 관리하도록 MQPS는 데이터 서버와 더욱 밀접하게 관계되어 있는 것을 볼 수 있다. 데이터 서버 계층은 정적인 뷰의 특성을 가지는 데이터 웨어하우즈를 갱신하는 유지 관리 트랜잭션을 주기적으로 구성하는 책임을 맡고 있다.

그러나 접속 단절이 발생한 경우, 사용 데이터의 일관성 유지와 현실 데이터 이용 가능성을 보장하기 위해서 모바일



(그림 6) 설계된 MQPS의 내부 구조

클라이언트(MC)의 응용 세션(Application Session)이 요청하는 정보를 유지할 수 있도록 MQPS는 뷰의 버전을 지속적으로 유지해야 한다. 따라서 MQPS는 MC의 특정 응용 세션에서 요청한 뷰의 버전을 보관하고 있는 유선망에 존재하는 버퍼(Buffer)와 같은 역할을 한다고 볼 수 있다. 그러므로 뷰의 변경된 속성들을 저장하기 위한 MQPS의 저장 공간은 얼마나 많은 버전을 보관해야 할지 판단할 수 없으므로 동적이어야만 한다.

설계된 MQPS의 내부 구조는 (그림 6)과 같이 메타 데이터, 질의 처리 라이브러리, 논리적 튜플 버전(LTV), 그리고 통신 이력 관리자로 구성된다.

모바일 클라이언트(MC)는 확장된 SQL "create view"를 포함하고 있는 "create view" 메시지를 통해 질의를 보낸다. 이 문장은 SQL 질의를 포함할 뿐만 아니라 "ON condition"이라는 응용 이동성 인식(Application-Mobility-Aware) 문장도 포함한다. "ON condition"은 사용했던 뷰(Materialized View)의 새로운 버전을 생성하고 뷰 유지 관리를 알리는데 사용되는 문장이다. 이렇게 MC에서 작성된 질의는 모바일 에이전트로부터 유선망의 MQPS로 보내지게 되면, MQPS는 질의 처리기를 통해 보내진 SQL 문장에 따라 뷰를 생성하게 된다. 그 다음 "ON condition"에 의해 데이터 소스를 모니터링하기 시작한다. 모니터링 중에 관련된 변화가 발생하면 뷰 유지관리가 수행되어야 하며, MC에 의해 요청된 질의는 질의 처리기로 다시 보내져서 새로운 버전이 생성된다. MQPS는 이러한 새로운 버전을 저장하고 새로운 뷰를 처리한다. 그런 다음에 MC의 에이전트는 메시지에 의해 비동기적으로 갱신된 새로운 뷰를 MQPS로부터 받게 된다.

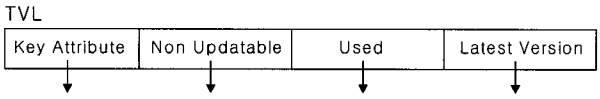
3.3 MQPS의 설계 및 알고리즘

3.3.1 MC 응용 세션의 데이터 처리을 위한 설계

MQPS의 핵심 목적은 MC의 응용 세션에 의해서 사용하도록 최종 버전을 항상 유지 관리하는데 있다. 따라서 항상 MC는 새로운 응용 세션으로 개시하고 최종 버전을 요청하게 되므로, 비 선형 리스트(Linked List)로 버전 관리를 위한 자료 구조를 준비해야 한다. (그림 7)은 이러한 버전 관리를 위한 튜플 버전 리스트(Tuple Version List)를 나타내고 있다.

(그림 7)에서와 같이 TVL은 "key attribute"와 "non-updatable attribute"를 포함하고 있으며, MC에 의해서 사용된 갱신 가능한 버전 관리 리스트와 튜플의 가장 최근의 버전을 관리하는 리스트로 구성되어 있다.

모바일 컴퓨팅 환경에서 접속 단절시의 대량의 데이터 처리를 위한 모바일 기기의 한계를 극복하고 현실의 데이터를 사용 가능하며 사용 데이터의 일관성을 유지하기 위한 MQPS는 사용되었던 모든 뷰를 관리하기 위하여 (그림 8)과 같은 세 개의 테이블을 가진 MQPS 관계 스키마가 요구된다.



(그림 7) MQPS의 버전 관리를 위한 튜플 버전 리스트 구조

```
CREATE TABLE Fixed_Attributes
( tuple_ID     INTEGER     NOT NULL,
  attributes   data type   NOT NULL ) ;

CREATE TABLE Updatable_Attributes
( tuple_ID     INTEGER     NOT NULL,
  VN           INTEGER     NOT NULL,
  attributes   data type   NOT NULL ) ;

CREATE TABLE Latest
( operation    CHAR(6)     NOT NULL,
  attributes   data type   NOT NULL ) ;
```

(그림 8) MQPS에서 뷰 관리를 위한 스키마 구조

```
public void Update_TVL() {

 VN = Latest Version +1 ;

 // Query MC_Request_Table for version VN
    SELECT * ;
    FROM MC_Request_Table ;
    WHERE tuple VN = VN ;
    GROUP BY attributes ;

 For each tuple T {
     Find the TVL that matches the key ;
     if found {
        if TVL.LatestInUse = TRUE {
           append TVL.Latest to Used ;
           set TVL.LatestInUse = FALSE ;
        } //end if

        if operation = "update" {
           Latest = (VN | attribute) ;
        else operation = "delete"
           Latest = (VN | null) ;
        } //end if
     else {
        create new TVL ;
        insert key and non-updatable attributes ;
        Latest = (VN | attribute) ;
     } //end else
     } //end if
 } //end For
 VN = Latest version number ;
 }
```

(그림 9) 튜플 버전 리스트(TVL) 갱신 알고리즘

```
public void Send Latest(Latest_Version) {

For each TVL {
    if Latest.value ≠ the last element of Used.value {
       TVL.LatestInUse = TRUE ;
       send latest.value to MC's agent ;
       }
    else {
       send the no-change token ;
       }
 } //end For

 이용가능 버전 목록에 Latest. Version number 추가 ;
 MC's agent 로 Latest Version number 전달 ;
}
```

(그림 10) 변경된 버전을 MQPS에서 MC로 전송하는 알고리즘

```
public void Update_TVL(Latest Version) {

For each tuple T in table Latest {

// Query Fixed_Attributes for T
   SELECT * ;
   FROM Fixed_Attributes FA, Latest L ;
   WHERE FA.attributes = L.attributes ;

   if found {
      if operation = "update" {
        INSERT INTO Updatable_Attributes(tuple_ID, VN, etc);
        VALUES(FA.tuple_ID, VN, etc);
        }
      else operation = "delete" {
        INSERT INTO Updatable_Attributes(tuple_ID, VN, etc);
        VALUES(FA.tuple_ID, VN, NULL);
        } //end if
   else {
      // Create new tuple_ID integer
      INSERT INTO Updatable_Attributes(tuple_ID, VN, etc);
      VALUES(FA.tuple_ID, VN, etc);

      INSERT INTO Fixed_Attributes(tuple_ID, etc);
      VALUES(new tuple_ID, etc);
      }
   } //end if
} //end For
}
```

(그림 11) 뷰 관리 관계 테이블 갱신 알고리즘

(그림 9)는 MC에 의해서 사용되었던 뷰의 버전 관리를 위해 튜플 버전 리스트(TVL)가 어떻게 갱신되는지를 알기 위한 TVL 갱신 알고리즘이다.

(그림 10)은 MQPS에 의해서 관리되고 있는 튜플의 버전이 갱신되었을 때, 모바일 클라이언트(MC)로 TVL의 갱신된 값을 전송하는 알고리즘이다.

(그림 11)은 튜플 버전이 갱신된 사항(TVL)을 (그림 8)에서 생성된 MQPS의 뷰 관리를 위한 관계 테이블로 변경 사항을 입력하기 위한 알고리즘이다.

3.3.2 MC 사용 데이터의 현실성(Currency) 보장을 위한 설계

MQPS의 다른 기능 설계 중에 하나는 MC의 현실 데이터의 이용가능성을 보장하기 위하여 서버측의 DBMS 에이

```
public class Coordinating DBMS-agent() {

// Query DS2.r2
   SELECT DS2.r2.b, DS2.r2.c ;
   FROM DS2.r2 ;

   B = b and C = c ;
   local variable version number v = 0 ;

   Start view recomputation: {
      Supply query trip plan (query tree)
            with Query DS2.r2 results.
      Launch view evaluation DBMS-agent
            with query trip plan and version number v = 0
   } //end recomputation

   Begin Monitoring: {
   Every (Monitor_time) {
      SELECT DS2.r2.b, DS2.r2.c
      FROM DS2.r2

      if (B  ≠ b or C ≠ c) {
        version number v = v + 1 ;
        Start view recomputation: {
            Supply query trip plan (query tree)
                  with Query DS2.r2 results ;
            Launch view evaluation DBMS-agent
                  with query trip plan
                  and current version number v ;
        B = b and C = c ;
        } //end recomputation
      } //end if
   } //end Every
   } //end Monitoring
   }
```

(그림 12) 데이터 현실성 평가 알고리즘

전트와 연동되어 모니터링 뷰(Monitor View)를 생성하여 현실성을 재계산하는 방법이다. 이를 위해 데이터 서버 DS1, DS2, DS3 각각의 r1(a,b), r2(b,c), r3(c,d) 테이블이 있고, 이들 테이블을 조인하여 데이터의 현실성을 측정하기 위한모니터링 뷰가 생성되어 있다고 가정하자. (그림 12)는 DBMS 에이전트와 모니터링 뷰가 연동되어 데이터의 현실성 평가를 하는 알고리즘이다.

3.3.3 MC 사용 데이터의 일관성(Consistency) 유지를 위한 설계

MQPS의 또 다른 기능 설계 중에 하나는 MC의 사용 데이터의 일관성을 보장하기 위하여 트랜잭션이 발생한 후에 뷰와 데이터 소스를 사상(Mapping)하는 작업이 필요하다. 이를 위해서 본 논문에서는 다음과 같은 (정의 1)하에 MC의 사용 데이터 일관성 보장을 유지하는 방법으로 사용한다.

(정의 1)

데이터 소스 상태 벡터 $ssv_k=[ds1_k, ds2_k, ..., dsm_k]$는 주어진 트랜잭션 T_k가 실행된 후에 모든 데이터 소스의 상태를 포함한다고 하자. 그러면 T_k 트랜잭션을 수행한 후에 뷰에서 데이터 소스 상태 m까지의 사상(Mapping)이 다음과 같이 존재한다 :

1. 각 뷰의 상태 vs_j는 임의의 k에 대하여 타당한 상태 $m(vs_j) = ssv_k$를 반영한다.
2. 만약 $vs_j = vs_i$ 이면, $m(vs_j) < m(vs_i)$이다.

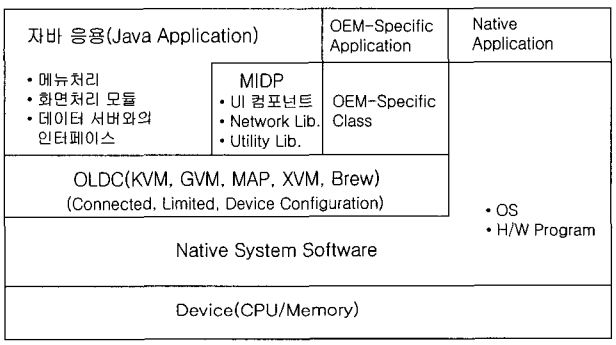
4. 시뮬레이션 모델과 분석

이 장에서는 3장에서 제안한 MQPS를 MC의 응용에 적용시켜 시뮬레이션하기 위한 환경을 제시하고 이를 통하여 기존 방식과 MQPS를 비교 평가한다.

4.1 시뮬레이션 모델

제안한 MQPS의 시뮬레이션 환경은 MC 환경을 위해 VM 제공 에뮬레이터(Emulator), MQPS 환경을 위해 Windows 2000 운영체제와 SQL Server 2000 DBMS, 그리고 JAVA (IDK1.3)를 사용하였으며, VM SDK로는 KVM SDK 1.0을 사용하였다.

MC의 응용 프로그램은 자바(Java) 프로그램으로 구동되며, (그림 13)과 같은 응용 구조를 가진다. 자바 응용(Java Application)은 MC의 응용 프로그램이고, OEM-Specific 응용과 Native 응용은 벨소리, 주소록과 같은 휴대폰 고유의 기능을 수행한다. MIDP(Mobile Information Device Profile)는 자바 구동을 위한 플랫폼으로 무선 전화 및 쌍방향 무선 호출기 등이 소프트웨어 애플리케이션 과 상호 교류하는 방법을 정의한 J2ME API로 UI 컴포넌트, 네트워크 라이브러리, 그리고 유틸리티 라이브러리를 포함하고 있다. Native System Software는 MC의 운영체제 및 하드웨어 등의 프로그램을 말한다.



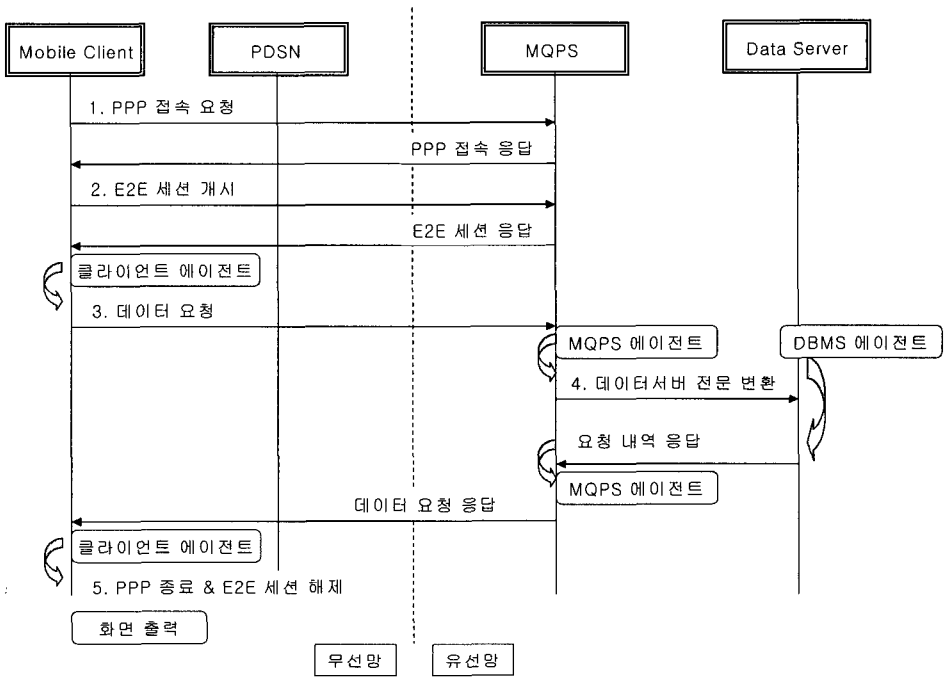
(그림 13) MC의 응용(Application) 구조

본 시뮬레이션의 응용은 MC를 이용한 금융계좌 조회 업무에 관한 것이다. MC를 이용하여 은행 데이터 서버의 금융계좌를 조회하는 금융계좌 조회 업무에 있어서, 무선으로 데이터를 송수신하는 MC와 MC와의 TCP/IP 프로토콜을 이용한 패킷 데이터 전송 서비스를 제공하는 PDSN 및 MC의 계좌 조회 응용으로 부터의 금융계좌 조회 요청에 따라 해당 은행 데이터 서버 금융계좌의 거래내역 정보를 제공하는 데이터 서버와 연동되어 데이터의 비축과 현실 데이터의 이용가능성 보장, 그리고 사용 데이터의 일관성 유지를 하기 위한 MQPS로 구성 되는 적용 사례를 (그림 14)를 통해 볼 수 있다.

4.2 성능 분석

본 논문에서 제안한 MQPS 구조의 성능 평가를 위해, 2장에서 설명한 C-I-S 구조와 MC에서 요청한 응용 세션에 대한 데이터 신뢰성 보장 서비스를 위한 MQPS와의 메시지 전달 빈도수와 데이터 전송 또는 메시지의 크기를 비교·분석하였다.

임의의 데이터 소스를 i라 하면, 데이터 소스는 $1 \leq i \leq u$ 의 범위에 있다고 하자. 그리고 메시지의 크기를 N 이라



(그림 14) 시뮬레이션 응용의 데이터 처리 흐름도

하면, 데이터 소스 i의 메시지 크기는 Ni가 된다. 또한, n_i는 데이터의 갱신 횟수를 나타낸다. 이때 C-I-S의 메시지 전달 빈도수는 (식 1)과 같고, 전송한 메시지의 크기는 (식 2)와 같다[14].

$$O(1 + \sum_{j=1}^{u} nj(2u-1)) \quad \text{(식 1)}$$

$$O(1) + \sum_{j=1}^{u} nj\ (O(1) + (\sum_{i=1,}^{u} (O(1) + O(1/Ni)))) \quad \text{(식 2)}$$

제안한 MQPS 구조에서, 임의의 메시지에 대해 (그림 12)에서 제시한 알고리즘에 의해 데이터의 정확성을 평가하므로 데이터 소스 u에 대해 뷰 정확성 검사를 실시한다. 따라서 MQPS 구조의 메시지 전달 빈도수는 (식 3)과 같다.

$$O(u) \quad \text{(식 3)}$$

그리고 MQPS 구조에서 전송한 메시지 또는 데이터의 크기는, 초기 전송 메시지의 크기가 $N1$인 메시지를 다음 사이트 DS2로 전송하면 메시지의 크기는 $N1+N2$가 되며, 이 메시지를 DS3로 전송하게 된다, 그러므로 MQPS 구조에서 전송한 메시지의 크기는 (식 4)와 같다.

$$N1 + (N1+N2) + (N1+N2+N3) + ... = Nu + \sum_{j=1}^{u-1} Nj(u-j) \quad \text{(식 4)}$$

모바일 컴퓨팅 환경에서 무선 업무처리를 요하는 사용자가 증가함에 따라 무선망의 약한 연결성과 접속 단절시의 데이터 정확성과 신뢰성 보장 문제는 모바일 데이터베이스 환경에서 중요한 문제가 아닐 수 없다. 따라서 무선 환경의 사용자가 증가하더라도 안정적이고 신뢰성있는 서비스를 제공하기 위하여, 모바일 클라이언트(MC)의 요청 메시지 전달 빈도수를 줄이는 요소와 모바일 기기의 한계성을 극복하기 위해 전송 메시지의 크기를 최소화하는 요소는 모바일 데이터베이스 환경에서 안정된 서비스를 제공하기 위한 중요한 요소이다.

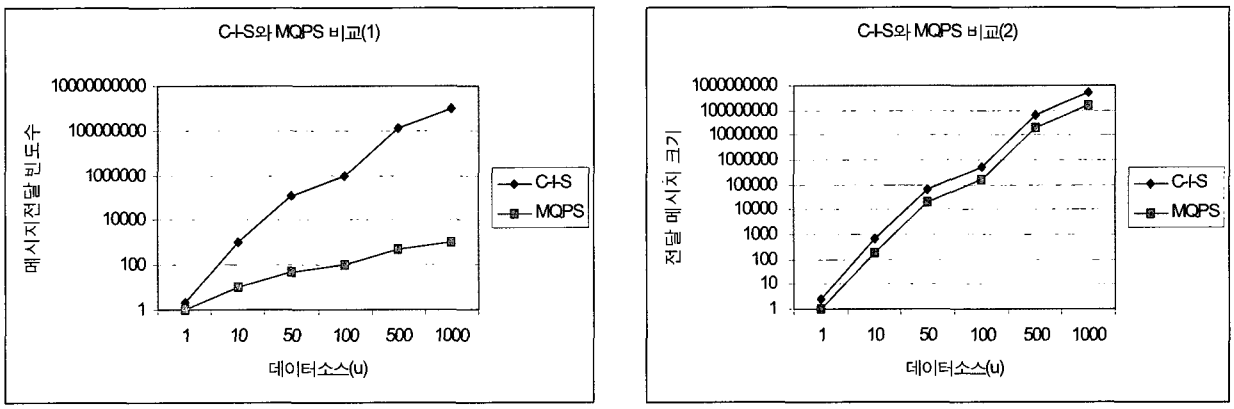
따라서 두 가지 요소를 가지고 C-I-S 구조와 MQPS 구조의 성능 비교를 하여, 사용자 증가에 따라 메시지 사용 빈도수와 메시지의 크기가 최적화가 무선 컴퓨팅 환경 서비스에 얼마만큼 중요한 영향을 미치는가를 살펴보면 <표 1>과 같다.

<표 1>은 사용자 요청 수에 따른 기존 C-I-S 구조의 메시지 사용 빈도수와 메시지 크기의 최적화를 제안한 MQPS 구조의 메시지 사용 빈도수와 메시지 크기 최적화와 비교한 결과를 보여준다. 모바일 컴퓨팅 환경에서 사용자의 증가에 따라, 안정적이고 신뢰성 있는 서비스를 위해 제안한 MQPS가 C-I-S 구조 보다 우수하다는 것을 확인할 수 있다. <표 1>을 그래프로 표현하면, (그림 15)와 같다.

〈표 1〉 사용자 데이터 처리 요청 증가에 따른 두 가지 요소의 비교표

	메시지 전달 빈도수		전달 메시지 크기	
데이터 소스(u)	C-I-S	MQPS	C-I-S	MQPS
1	2	1	3	1
10	1046	10	687	175
50	126226	50	67894	20875
100	1004951	100	523151	166750
500	125124751	500	63175600	20833750
1000	1000499501	1000	502873873	166667500

※ 여기서, 초기 메시지 크기 N =1 / 데이터 갱신 횟수 n = 1로 가정한 결과치 임.



(그림 15) 사용자 데이터 처리 요청 증가에 따른 두 가지 요소의 비교 그래프

5. 결 론

본 논문에서는 모바일 데이터베이스 환경에서 무선망의 약한 연결성과 접속단절 시에 모바일 컴퓨팅 과정에서 발생할 수 있는 데이터 처리의 안정성과 사용 데이터의 일관성 유지, 그리고 대용량 데이터 사용시의 효율적인 모바일 기기의 한계성 극복을 위해 모바일 질의 처리 시스템(MQPS : Mobile Query Processing System)에 대한 설계 방법을 제안하였다.

모바일 데이터베이스 환경을 위해 기존에 제시된 모바일 클라이언트-서버 모델들은 무선망의 약한 연결성과 접속 단절의 문제점을 해결하여 대용량의 데이터 처리를 위한 모바일 기기의 한계 극복과 현실 데이터의 이용가능성, 그리고 사용 데이터의 일관성 유지를 하려고 하였으나 완벽하게 이러한 문제점 해결을 달성할 수 있는 방안은 아니다. 무선망에서는 약한 연결성으로 인해 유선망 환경과 같이 연결 상태가 정상적이고 안정적인 경우에는 나타나지 않는 문제들이 발생하므로 무선망 환경에서의 모바일 데이터베이스 연산은 이러한 특성들을 고려하여 네트워크의 연결 상태에 적응 할 수 있어야만 한다.

본 논문에서는 이러한 단점을 보완하기 위하여 클라이언트와 서버 양쪽에 에이전트를 가지는 시스템(C-I-S)이외에 모바일 클라이언트 지향적인 데이터 웨어하우즈(Mobile Client-Oriented Data Warehouse) 기능을 가진 추가적인 개체로 모바일 질의처리 시스템(MQPS)을 설계하였고, 기존 방식과의 성능 비교를 위해 제안한 MQPS의 시뮬레이션 환경을 MC 환경을 위해 VM 제공 에뮬레이터(Emulator), MQPS 환경을 위해 Windows 2000 운영체제와 SQL Server 2000 DBMS, 그리고 JAVA (IDK1.3)를 사용하였으며, VM SDK로는 KVM SDK 1.0을 사용하여 사용자 요청 증가에 따른 전달 메시지의 빈도수와 사용 메시지의 크기 변화를 통한 성능 비교를 하였다.

본 시스템의 기대효과는 모바일 컴퓨팅 환경의 확산과 사용자 증가에 따라 모바일 기기의 한계와 불안전한 접속 등의 이유로 접속이 단절되어 데이터 처리의 일관성 유지와 이용 가능성 저하의 문제점 등을 해결할 뿐만 아니라, 상용 서비스하고 있는 모바일 비즈니스 업무의 대용량의 데이터 처리 부분에 적용도 가능하다.

향후의 과제로는 본 논문의 시뮬레이션 환경은 KVM 플랫폼에 초점을 두었기 때문에 시스템 구현시의 범용화에 있다. 즉, 현재 국내 이동통신사마다 KVM, GVM, XVM, MAP, Brew 등 VM 플랫폼이 모두 상이 하므로 앞으로 더 많은 시뮬레이션이 필요하다. 아울러 모바일 환경의 약한 연결과 접속성 단절 상황에서 C-I-S 모델보다 MQPS가 어떻게 데이터 비축 문제와 공유 데이터 일관성 문제를 해결했는지에 대한 검증을 MQPS 구현시에 검증이 필요하다.

참 고 문 헌

[1] 이재우, "모바일 데이터베이스 응용 사례", 데이터베이스연구회, 17권 3호, pp.115~118, 2001. 9.

[2] 최미선, 김영국, "이동(Mobile) 데이터베이스 개요 및 연구 현황", 데이터베이스연구회, 17권 3호, pp.3~16, 2001. 9.

[3] Margaret H. Dunham and Vijay Kummer, "Impact of Mobility on Transaction Management", Proceeding of the International Workshop on Data Engineering for Wireless and Mobile Access, pp.14~21, August, 1999.

[4] D. B. Johnson & W. Zwuenepoel, "Recovery in Distributed Systems using Optimistic Message Logging and Checkpointing Journal of Algorithms", 11(3) : 462~491, September, 1990.

[5] Sanjay Kummar Madria, Bharat K. Bhargava, "A Transaction Model to Improve Data Availability in Mobile Computing", Distributed and Parallel Databases 10(2) : 127~160, 2001.

[6] G. Walborn and P. K. Chrysanthis, "Proceeding in Mobile Database Applications", In Proceeding of the 14th Symposium on the Reliable Distributed Systems, September, 1995.

[7] E. Pitoura, B. Bharagava, and O. Wolfson, "Data Consistency in Intermittently Connected Distributed Systems", IEEE Transactions on Knowledge and Data Engineering 11(6), pp.896~915, Nov/Dec., 1999.

[8] N. Neves and W. K .Fuchs, "Adaptive Recovery for Mobile Environments Communications of the ACM", 40(1) : 69~74, January, 1997.

[9] D. Barbara, "Mobile Computing and Databases - A Survey", IEEE Transactions on the Knowledge and Data Engineering, 11(1) : 108~117, 1999.

[10] B. R. Badrin, A. Fox, L. Kleinrock, G. Popek, P. Reiher, and M. Satyanarayanan, "A Conceptual Framework for Network Adaptation", the IEEE Mobile Networks and Applications, 5(4) : 221~231, 2000.

[11] S. Mazumdar, M. Pietrzyk, and K. Chrysanthis, "Caching Constrained Mobile Data", CSD Technical Report, Univ. of Pittsburgh, 2001.

[12] C. Spyrou, G. Samaras, E. Pitoura, S. Papastavron, and P. K. Chrysanthis, "The Dynamic View System(DVS) : Mobile Agents to Support Web Views", In the 17th Int'l Conf. on Data Engeering, pp.30~31, Apr., 2001.

[13] Y. Zhuge, "Incremental Maintenance of Consistent Data Warehouses", Ph.D thesis, Department of Computer Science, Stanford Univ., 1999.

[14] S. W. Lauzac, "Utilizing Customized Materialized Views to Create Database Services Suitable for Mobile Database Applications", ph.D thesis, Pittsburgh Univ., 2001.

[15] D. VanderMeer, "Data Access and Interaction Management in Mobile and Distributed Environments", Ph.D thesis, Georgia Institute of Technology, 2003.

주 해 종

e-mail : hjjoo@hanmir.com

1988년 명지대학교 전자계산학과(공학사)

1990년 명지대학교 일반대학원 전자계산학과(공학석사)

1997년 명지대학교 일반대학원 컴퓨터공학과(박사수료)

1992년~1996년 KTNet 기술연구소 연구원

1994년~1996년 명지대학교, 성결대학교 시간강사

1996년~1997년 구미기능대학 전자기술학과 전임강사

1997년~2000년 대원과학대학 사무자동화과 조교수

2000년~2002년 KOCES 전자금융사업부 팀장

2002년~2005년 대원과학대학 멀미티디어과 조교수

관심분야 : 모바일 DB, 모바일 전자지불결제, 멀티미디어 DB

박 영 배

e-mail : parkyb@mju.ac.kr

1993년 서울대학교 컴퓨터공학과 (공학박사)

1990년~1992년 명지대학교 전자계산소장

1997년~2001년 명지대학교 산업대학원장

1981년~현재 명지대학교 컴퓨터공학과 교수

관심분야 : Spatial DB, Multimedia DB, Web DB, XML DB, Large Fingerprint DB, Data Warehousing & Data Mining 등