분산 INGRES 데이타 베이스 시스템을 위한 초기환경 구축과 질의어 처리에 관한 연구

손 용 락, 안 순 신

고려대학교 전자 전산공학과

A Study on Initial Environment Construction and Query Processing for Distributed INGRES Database System

Sohn Yong Lak, An Sun Shin

Dept. of Electronic & Computer Science Engineering, Korea Univ.

Abstract

This paper is concerned with distributed INGRES database system. The main motivation is to provide dynamic reconfiguration and global database consistency. Dynamic reconfiguration will provide proper initial environment to newly generated system. By use of transaction characteristic, database can be maintained in consistent state.

1. 서론

분산 데이타 베이스는 컴퓨터 네트워크상에 구축된 통합된 형태의 데이타 베이스라고 표현할수 있다.[1] 분산 데이타 베이스를 구성하는 데이타들은 네트워크상의 여러 호스트에 저장되어 있다. 그러나 사용자에게는 원하는 데이타가 어느 호스트에 위치하고 있는지에 대해서는 인식하지 않은채 접근이 가능하도록 분산 데이타 베이스 시스템이 이 기능을 제공해 주어야 한다. 그리고 데이타들이 전체 호스트에 걸쳐 일관성 있는 값을 유지할수 있도록 환경 구축 및 데이타관리가 필요하다.[1]

본 연구에서는 동질 (Homogeneous) 의 시스템에서 기존의 INGRES 데이타 베이스 시스템에 동적 재구성 (Dynamic Reconfiguration) 기능 및 질의어에 대한 분산처리기능을 추가하여 분산 INGRES 데이타 베이스 시스템으로의 전환을 위한 설계 및 부분적 구현을 이루고 있다.

본 연구는 아래의 기본 가정하에 이루어 지고 있다.

1) 사용자 릴레이션은 중복되지 않는다. 그러므로 데이타 베이스에 대한 consistency 문제는 시스템 카탈로그에 국한한다.

2) 릴레이션은 분리되지 않는다. 전체 릴레이션은 한 호스트에만 존재한다.

3) 통신 선로 상에는 고장이 발생하지 않는다.

4) 몇몇 특정한 작업에 대하여는 atomic action 이 가능하다.

2. 분산 INGRES 데이타 베이스 시스템의 전체적 구조

호스트에는 각 데이타 베이스를 위한 serverdb 프로세스가 각 데이타 베이스별로 동작한다. 이를 위하여 호스트가 ON 될때 각 데이타 베이스에 대한 serverdb 를 생성시킨다. 이를 위하여 각 호스트의 "dblist" 파일에 현 호스트에 있는 데이타 베이스이름을 기록한다. serverdb 프로세스는 "dbname_host" 파일에서 데이타 베이스가 존재하는 호스트이름을 취하여 각 호스트의 동일한 데이타 베이스를 위한 serverdb 프로세스에게 현 호스트가 ON 되어 serverdb 프로세스가 생성되 었음을 알려준다. 이들 dblist, dbname_host 파일들에 대한 편집작업은 super user 만이 가능하며 새로운 데이타 베이스를 추가할 경우에는 미리 dblist 파일에 데이타 베이스 이름을 추가하고 dbname_host 파일에 호스트 이름들을 추가하여야 한다. 새로이 추가된 데이타 베이스에 대한 serverdb 를 생성 시키려면 호스트를 OFF 시킨후 reboot 시킴으로서 가능하다.

각 호스트의 데이타 베이스에 대한 데이타 베이스 시스템의 동작은 아래 그림과 같다.

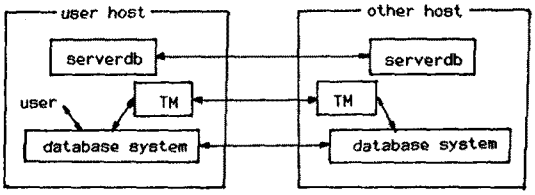

그림 1. 전체 시스템의 동작
Fig. 1. Feature of global system

위 그림에 나타나는 각 요소들의 역할은 다음과 같다.

2.1 serverdb 의 역할

1) 각 데이타 베이스마다 탄생한다.

2) 네트워크상의 각 호스트의 상태 및 데이타 베이스의 상태를 파악한다.

3) 현 호스트 및 나머지 호스트의 consistency 를 유지한다.

4) 사용자 호스트로부터 요청된 명령어를 수행한다.

2) 와 3) 을 위하여 serverdb 는 데이타 베이스의 상태를 나타내는 "dbstate" 표를 유지하여야 한다. dbstate 표의 내용은 각 호스트가 ON 될때 초기화 된다.

dbstate 표의 구조는 다음과 같다.

dbname	ldb	dbupdate	ldbs	cuh	luh	lmsh	cons	waitsys
	y/n	y/n	y/n	y/n	y/n	coord/parti	y/n	y/n

dbname : 데이타 베이스 이름
ldb : 현 호스트에 데이타 베이스의 존재 여부
dbupdate : 데이타 베이스의 사용 여부
ldbs : 현 호스트에 데이타 베이스 시스템의 존재 여부
cuh : 현 호스트에 사용자 존재 여부
luh : 현 호스트가 지난번에 ON되었을때 사용자의 존재 여부

lmsh : 현 호스트가 지난번에 ON되었을때 TM의 자격
cons : 현 호스트가 지난번에 ON되었을때 데이타 베이스의 상태
waitsys : 현 호스트가 현재 consistency가 보장될수 있는 시스템 카탈로그의 도착을 기다려야 하는가에 대한 선택

그림 2. dbstate표 의 구조
Fig. 2. dbstate table

2.2 데이타 베이스 시스템의 역할

1) 사용자로 부터 "ingres dbname"이라는 명령어가 도착하면 사용자 호스트에는 주 데이타 베이스 시스템이 나머지 호스트에는 종 데이타 베이스 시스템이 생성된다.

2) 주 데이타 베이스 시스템은 사용자로 부터 도착한 질의어를 종 데이타 베이스 시스템과 협력하여 처리한다.

2.3 데이타 베이스

1) 사용자로 부터 "creatdb dbname"명령어가 도착하면 모든 호스트에 새로운 데이타 베이스가 생성된다.

2) 새로운 데이타 베이스는 시스템 카탈로그만을 가지며 사용자로부터 릴레이션의 생성요청이 있으면 새로운 릴레이션들이 추가된다.

전체 데이타 베이스를 구성하고 있는 릴레이션들은 다음과 같이 구분될수 있다.

a) 타겟 릴레이션
질의어의 처리결과 내용이 변하는 릴레이션

b) 정보 릴레이션
질의어의 처리를 위하여 주어진 조건에 대한 정보만을 제공하는 릴레이션

2.4 호스트

네트워크상의 각 호스트는 아래와 같이 구분된다.

1) 사용자 호스트
사용자가 위치하고 있는 호스트이다.

2) 타겟 (Target) 호스트
타겟 릴레이션이 존재하는 호스트이다.

3) 정보 (Information) 호스트
정보 릴레이션이 존재하는 호스트이다.

4) 일반 호스트
시스템 카탈로그에 대한 consistency 를 맞추어 주기위한 작업만을 하는 나머지 호스트이다.

2.5 TM (트랜잭션 메니저)

TM 의 구분은 다음과 같이 이루어 진다.

1) 조정자 TM
타겟 릴레이션이 위치한 호스트에 존재하는 TM으로 한 질의어의 처리를 위해 전체 시스템에 걸쳐 한개만 존재한다.

2) 참가자 TM
조정자 TM이 존재하는 호스트를 제외한 나머지 호스트들의 TM 이다.

이들 TM 은 데이타 베이스 시스템의 탄생과 함께 생성되며 질의어의 처리시 트랜잭션 동작을 관장한다.

2.6 프로세스들간의 통신을 위한 프리미티브

1) sendb("데이타 베이스 이름", 메세지 종류, 메세지, 메세지 크기, 모드, "호스트이름");

이는 네트워크상의 동일한 데이타 베이스가 존재하는 호스트로 메세지를 보낸다. 데이타 베이스이름은 동일한 데이타 베이스가 존재하는 호스트들과 그룹 통신이 가능하게 해준다. 메세지 종류는 sendb 가 내부적으로 메세지 종류에 따라 적절히 선택되어 동작하도록 하여준다. 모드는 "브로드 캐스트", "싱글호스트", "로칼호스트"중 하나로 선택되어진다. 브로드 캐스트일경우 그룹에 등록된 모든 호스트로 메세지를 보내게 되며, 싱글 호스트일 경우에는 주어진 호스트 이름으로 메세지를 보낸다. 로칼호스트일 경우에는 메세지 종류로 선택된 현 호스트의 프로세스에게 메세지를 보낸다.

2) recvdb("데이타 베이스 이름", 메세지 종류, 메세지, 메세지크기, 모드, "호스트이름", 타임아웃 크기);

이는 현 호스트에 도착한 정보를 메세지로 읽어 들인다. 데이타 베이스 이름과 메세지 종류, 메세지 크기, 모드는 sendb 와 거의 유사하게 사용된다. 타임아웃 크기는 원하는 종류의 메세지가 주어진 타임아웃 동안에 도착하지 못하면 메세지를 보내야할 호스트가 OFF 되었다고 간주되어 그룹에서 제거된다. serverdb 가 새로운 호스트로 부터 메세지를 받으면 그 호스트를 그룹에 추가 한다. 메세지를 받을때마다 그룹유지에 관한 작업을 함으로서 네트워크의 상태를 파악할수 있게된다.

2.7 프로세스들간의 메세지

1) TM간의 메세지
TR_START, TR_RESP, COM_ABT

2) TM과 데이타 베이스 시스템간의 메세지
TR_REQ, PROC_QRY, UP_SYS, PROC_RESP, COM_ABT_PROC

3) serverdb간의 메세지
rise, rise_resp, sys_cat, sc_resp, command, com_resp

3. 분산 INGRES 데이타 베이스 시스템의 초기환경 구축

분산 데이타 베이스 시스템에서는 전체 데이타 베이스를 consistent 한 상태로 유지하는것이 매우 중요하고 어려운 작업이다. 이를 위한 작업은 두가지로 나눌수가 있다. 첫째는 호스트가 ON 될때 serverdb 는 네트워크상의 각 호스트 및 데이타 베이스의 상태를 확인하여 전체 데이타 베이스를 consistent 한 상태로 전환시켜 주는 것으로서 이때 비로소 데이타 베이스에 대한 접근이 가능해진다. 둘째는 데이타 베이스 시스템이 질의어를 처리할때 그결과로 전체 데이타 베이스가 consistent 한 상태로 유지 하는것으로서 이를 위하여 트랜잭션 개념이 필요하다.

본 절에서는 첫째 작업에 대하여 언급하며 다음 절에서는 둘째 작업에 대하여 언급한다.

발생 할수 있는 여러 상태로는 아래와 경우들이 있을수 있다.

1) 네트워크상에 호스트1만이 ON되어 있고 아직 데이타 베이스는 만들어지지 않은 상태

2) 네트워크상에 호스트1만 ON된 상태에서 호스트2가 ON되어 serverdb가 새로이 생성되는 상태 (아직 데이타 베이스는 만들어 지지 않은 상태)

3) 네트워크상에 호스트1과 호스트2가 ON된 상태에서 호스트1에 사용자로부터 "creatdb A" 명령어가 도착했을 경우.

4) 네트워크상에 호스트1 과 호스트2만 ON 상태에 있으며 데이타 베이스는 시스템 카탈로그만을 지닌 상태에서 데이타 베이스가 존재하지 않은 호스트3가 새로이 ON되는 경우

5) 네트워크상의 여러 호스트들중 변경된 데이타 베이스가 존재하는 경우

6) 네트워크 상의 호스트1과 호스트2에 데이타 베이스 시스템이 실행되고 있는중에 호스트3이 ON 되는 경우

여기서는 5) 와 6) 의 경우를 살펴 본다.

3.1 네트워크 상의 여러 호스트들중 변경된 데이타 베이스가 존재하는 경우

각 호스트가 ON 될때 데이타 베이스에 대한 consistency 를 맞추어주어야 한다. 현 호스트에 있는 데이타 베이스가 consistency 를 보장해줄수 없을 경우에는 이미 ON 되어 있는 다른 호스트로 부터 consistency 를 보장해줄수있는 시스템 카탈로그를 받아 현 호스트의 시스템 카탈로그를 consistent 한 상태로 바꾸어 주어야 한다. 현재 ON 상태에 있는

호스트들중 어느 호스트도 consistency 를 보장해줄수가 없으면 각 호스트는 consistency 를 보장해줄수 있는 호스트가 ON 되기를 기다려야한다.

새로이 ON되는 호스트에 데이타 베이스가 존재하지 않을 경우에는 이미 ON되어 있는 호스트들중 consistency를 보장해 줄수있는 호스트가 있을 경우에는 그 호스트로부터 시스템 카탈로그를 받아야 한다. 그 반대의 경우에도 마찬가지이다. 이들에 대한 판단은 호스트가 ON 될때 serverdb에 의해 이루어 진다.

그림 3은 이 경우의 전체 시스템의 동작을 나타내고 있다.

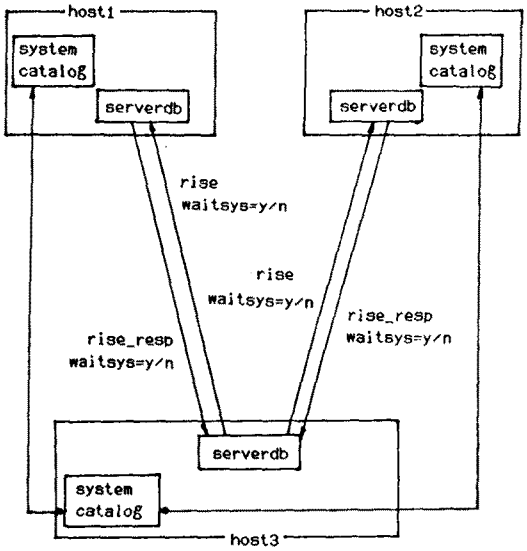

그림 3 전체 시스템의 동작
Fig 3 Feature of global system

호스트가 ON 될때 데이타 베이스의 상태는 dbstate 표에 기록되어야 한다. 이들에 대한 정보는 호스트가 OFF 될때 데이타 베이스의 상태를 나타내주는 dbname_dbst 파일로부터 얻을수 있으며 이로부터 consistency 여부를 판단할수 있다. dbname_dbst 파일은 userhost, membership, cons 로 구성되어 있으며 데이타 베이스 시스템이 질의어 처리를 위한 트랜잭션을 수행하고 있을때 데이타 베이스의 상태가 dbname_dbst 파일에 기록 되어져야 한다. 즉 호스트가 사용자 호스트일 경우에는 userhost 에 "yes" 를 그렇지 않으면 "no" 를 기록한다. 트랜잭션을 시작하면 cons 를 "no" 로 트랜잭션을 마치면 cons 를 "yes" 로 기록한다. membership 에는 각 트랜잭션을 위한 TM 이 조정자로 동작하였으면 "coord" 로 참가자로 동작하였으면 "parti"로 기록된다. 결론적으로 Userhost/Parti/Incons일때와 호스트가 OFF 되기전 사용자 호스트로 동작하고 있지 않았을 경우에는 consistency 를 보장할수가 없다. 사용자 호스트가 아니었을 경우에는 현 호스트가 OFF 되어도 사용자 호스트와 나머지 호스트들은 사용자로 부터 도착하는 질의어를 계속 처리하여 데이타 베이스의 상태를 바꿀수 있으므로 consistency를 보장할수가 없으며 사용자 호스트였을지라도 "Parti/Incons" 일 경우에는 그 질의어에 대하여 타겟 호스트 및 나머지 호스트들이 처리를 계속 진행하여 데이타 베이스의 상태를 바꿀수 있으므로 consistency 를 보장할수가 없다. 이때는 새로이 ON 되는 호스트의 serverdb 는 waitsys 를 "yes" 로 기록하여 각 호스트의 serverdb 로 보내며 이에 대한 응답도 각 호스트의 dbname_dbst 파일의 상태에 따라 기록된 waitsys 를 받는다. 이때 waitsys 가 "no" 인 정보를 보내온 호스트들의 시스템 카탈로그는 consistency 를 보장할수 있다. 그러므로 이들중 최소 비용으로 시스템 카탈로그의 내용을 전송할수 있는 호스트를 선택하여 consistent 한 시스템 카탈로그를 받아 consistency 를 맞추어 준다.

3.2 네트워크상의 호스트1 과 호스트2에 데이타 베이스 시스템이 실행되고 있는중에 호스트3가 ON되는 경우

호스트1 및 호스트2에는 데이타 베이스 시스템이 실행중이므로 이들 호스트에서는 consistency 를 보장할수가 있다. 호스트3는 다른 호스트에서 이미 데이타 베이스시스템이 동작중인 가운데 ON 되었으므로 consistency 를 보장해줄수 없다. 그러므로 호스트3는 호스트1이나 호스트2로 부터 시스템 카탈로그를 가져와야 한다. 그러나 호스트1과 호스트2의 데이타 베이스 시스템이 트랜잭션내에서 동작하고 있는 동안에는 데이타 베이스가 inconsistent 한 상태이므로 시스템 카탈로그를 옮길수가 없다. 그러므로 TM 과 serverdb 는 시스템 카탈로그에 대해 아래 그림의 형태로 동시성 제어가 이루어 진다.

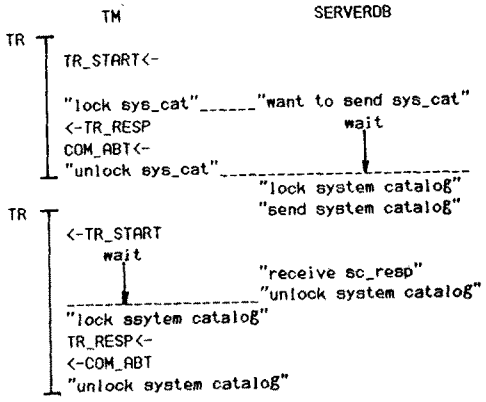

그림 4 TM과 serverdb사이의 시스템 카탈로그에 대한 동시성 제어
Fig 4 Concurrency control between TM and serverdb for system catalog

4 분산 INGRES 데이타 베이스 시스템 환경에서의 질의어 처리

질의어 처리는 여러 호스트의 데이타 베이스 시스템과 TM들의 협력으로 수행된다. 사용자는 질의어의 처리가 분산 데이타 베이스의 환경에서 수행되고 있다는것을 인지하지 못한채 사용할수 있어야 한다. 이를 위해서는 질의어의 문법에 분산의 성격이 나타나지 않아야 한다.

트랜잭션의 동작은 릴레이션 및 시스템 카탈로그의 내용을 변화 시키는 질의어의 처리에 필요로 한다. 이를 위하여 TM은 아래의 그림 5에 나타난 2 phase commit 프로토콜의 형태로 동작한다.

사용자로부터 질의어의 요청이 처음 들어오는 사용자 호스트의 TM은 조정자 TM이 되며 나머지 호스트의 TM은 참가자 TM으로 동작한다. 이들 TM의 자격은 질의어의 처리도중 타겟 릴레이션의위치에 따라 변하게 된다. 즉 타겟 릴레이션이 위치한 호스트는 조정자 TM으로 나머지 호스트의 TM은 참가자 TM으로 동작한다.

질의어의 처리가 끝나면 처음의 상태로 환원된다.

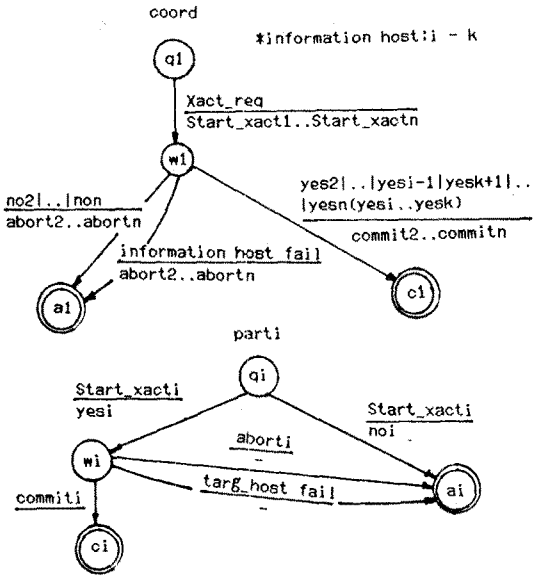

그림 5 2 phase commit 프로토콜의 동작

Fig 5 Feature of 2 phase commit protocol

각 호스트의 데이타 베이스 시스템과 TM의 동작을 살펴보면 아래와 같다.

4.1 사용자 호스트

1) 데이타 베이스 시스템은 사용자로부터 들어온 질의어의 처리가능을 판단한다. 처리가 가능하면 질의어에 나타나는 타겟 릴레이션이 존재하는 호스트이름을 TR_REQ 메세지의 targ_host 에 실어 TM에게 보낸다. 데이타 베이스 시스템은 TM으로 부터 COM_ABT_PROC 메세지가 도착하기를 기다리고 있다가 메세지의 flag 내용에 따라 commit 혹은 abort 한다. commit 할경우 질의어의 처리결과로 데이타 베이스의 상태를 변화 시킨다. abort 할 경우에는 데이타 베이스의 상태를 변화시키지 않은채 질의어의 처리를 중단한다.

2) TM 은 항상 init_coord 상태에 있다. 데이타 베이스 시스템으로부터 TR_REQ 메세지가 도착하면 각 호스트의 TM에게 TR_START 메세지를 보낸다. TR_REQ 에 실려있던 targ_host 의 호스트 이름이 사용자 호스트이면 TM은 wait_coord 상태가 된다. targ_host 가 사용자 호스트가 아니면 TR_RESP 메세지의 flag 에 "yes" 를 실어 타겟 호스트의 TM에게 보낸후 wait_parti 상태가 된다.

3) TM이 wait_coord 상태에 있을경우 TM은 각 호스트들로부터 TR_RESP 메세지가 도착하기를 기다린다. TR_RESP 의 flag 가 "no" 인 메세지가 도착하였으면 COM_ABT 메세지의 flag 에 "abort" 를 기록하여 각 호스트로 보낸후 abort 상태가 된다. 모든 정보 호스트들로 부터 도착한 메세지의 flag 가 모두 "yes" 로 기록되어져 있으면 COM_ABT 의 flag에 "commit"를 기록하여 모든 호스트의 TM에게 보낸후 commit 상태가 된다.

TM이 wait_parti 상태에 있을 경우 조정자 TM 으로부터 도착한 COM_ABT 의 flag 의 내용에 따라 "commit" 혹은 "abort" 상태가 된다.

4) TM이 commit 상태가 되면 TM이 조정자일경우 TM은 COM_ABT_PROC 의 flag 를 "commit" 로 기록하여 데이타 베이스 시스템에게 보낸후 init_coord 상태로 환원된다. TM이 commit 상태이며 참가자로 동작하였을 경우에는 COM_ABT 에 실려온 시스템 카탈로그의 변화된 내용 및 "commit" 를 COM_ABT_PROC 에 실어 데이타 베이스 시스템에게 전한후 위와같이 환원된다. TM이 abort 상태가 되면 COM_ABT_PROC 의 flag 를 "abort" 로 기록하여 데이타 베이스 시스템에게 전한후 위와 같이 환원된다.

4.2 나머지 호스트의 TM과 데이타 베이스 시스템의 동작

1) TM은 init_parti 상태에서 조정자 TM 으로부터 TR_START 메세지가 도착하기를 기다린다. TR_START 로 부터 PROC_QRY 메세지를 구성하여 데이타 베이스 시스템에게 보낸후 데이타 베이스 시스템으로부터 PROC_RESP 를 받아 flag 가 "CAN_PROC" 이면 앞서 도착한 TR_START 의 targ_host 를 확인한다. 타겟 호스트가 현 호스트이면 wait_coord 상태로 그렇지 않으면 TR_RESP 의 flag 에 "yes" 를 실어 타겟호스트의 조정자 TM에게 보낸후 wait_parti 상태가 된다. PROC_RESP 의 flag 가 "NOT_PROC" 이며 타겟호스트이면 COM_ABT 의 flag 에 "abort" 를 실어 각 호스트의 TM에게 보낸후 abort 상태가 된다. 타겟 호스트가 아니면 TR_RESP 의 flag 에 "no" 를 기록하여 조정자 TM에게 보낸후 abort 상태가 된다.

2) TM 이 wait_coord 상태이면 사용자 호스트의 wait_coord 상태와 유사하게 동작하며 init_parti 상태로 환원된다. wait_parti 상태일 경우에도 사용자 호스트의 wait_parti 상태와 유사하게 동작하며 init_parti 상태로 환원된다.

3) 데이타 베이스 시스템은 TM 으로부터 PROC_QRY 를 받아 질의어의 수행가능여부를 PROC_RESP 의 flag에 기록 하여 TM에게 보낸다. TM 으로부터 COM_ABT_PROC 를 받아 사용자 호스트의 데이타 베이스 시스템과 동일하게 동작한다.

5. 결론

본 연구에서는 제한된 형태의 분산 INGRES 데이타 베이스 시스템의 설계 및 구현을 하고있다. 사용자로부터 도착하는 print, help, create, destroy, quit 질의어와 ingres 명령어에 대하여 분산 INGRES 데이타 베이스 시스템에서 부분적으로 구현되고 있다.

호스트간에 왕래하는 정보의 양을 최대한 줄이는 작업이 전체적으로 구현될 시스템의 성능에 많은 영향을 줄것이다. 또한 릴레이션의 중복과 분리가 가능한 상태에서의 분산 데이타 베이스 시스템의 구축에 관한 연구가 계속 진행되어야 할것이다.

6. 참고 문헌

1] Stefano Ceri, Giuseppe Pelagatti, "Dristributed Databases principles & systems", Mc Graw Hill, 1984
2] Michael Stonebraker, "The INGRES papers:Anatomy of Relational Database System", Addison Wesley, 1984
3] John A. Stankovic, "Reliable Distributed Software", IEE Computer Society, 1985
4] C.J.Date, "A Guide to INGRES", Addison Wesley, 1987
5] C.J.Date, "An Introduction to Database Systems Vol I, II", Addison Wesley, 1987
6] James Martine, "Managing the Database Environment", Prentice Hall, 1983