비공유 공간 데이터베이스 클러스터에서 확장성 해싱 기반의 로그를 이용한 회복 기법

[0]장일국*, 장용일*, 박순영*, 배해영*
*인하대학교 전자계산공학과
e-mail : j98221042@dblab.inha.ac.kr

Recovery Method Using Extendable Hashing Based Log in A Shared-Nothing Spatial Database Cluster

[0]Il-Kook Jang*, Young-Il Jang*, Soon-Young Park*, Hae-Young Bae*
*Dept. of Computer Science and Engineering, Inha University

요 약

회복기법은 비공유 공간 데이터베이스 클러스터에서 고가용성을 위해 매우 중요하게 고려되고 있다. 일반적으로 데이터베이스 클러스터의 회복기법은 노드의 오류가 발생한 경우 로컬 로그와는 별도로 클러스터 로그를 생성하며, 이를 기반으로 해당 노드에서의 회복과정을 수행한다. 그러나, 기존의 기법은 하나의 레코드를 위해 다수의 갱신정보를 유지함으로써 클러스터 로그의 크기가 증가되고, 전송비용이 증가된다. 이는 회복노드에서 하나의 레코드에 대해 여러 번의 불필요한 연산을 실행하여 회복시간이 증가되고, 전체적인 시스템의 부하를 증가시키는 문제를 발생시킨다.

본 논문에서는 비공유 공간 데이터베이스 클러스터에서 확장성 해싱 기반의 로그를 이용한 회복기법을 제안한다. 제안기법에서의 클러스터 로그는 레코드 키값을 이용한 확장성 해싱을 기반으로 레코드의 변경사항과 실제 데이터를 가리키는 포인터 정보로 구성된다. 확장성 해싱 기반의 클러스터 로그는 크기와 전송비용이 감소하며, 회복노드는 하나의 레코드에 대해 한번의 갱신연산만 실행하므로 빠른 회복이 가능하다. 따라서 제안 기법은 확장성 해싱 기반의 클러스터 로그를 이용하여 효율적인 회복처리를 수행하며, 시스템의 가용성을 향상시킨다.

1. 서론

최근 무선 인터넷과 모바일 장치가 발전하고 대중화됨에 따라 지리정보와 같은 공간 데이터를 제공하는 서비스가 증가 하였다. 공간 데이터의 특성상 대용량의 데이터 저장과 관리를 위해 확장성과 가용성이 뛰어난 비공유 구조의 공간 데이터베이스 클러스터가 개발되었다[1]. 비공유 구조의 공간 데이터베이스 클러스터는 독립적으로 동작하는 노드들을 고속의 네트워크로 연결시키는 구조를 가지며 분할 정책과 복제 정책의 사용으로 사용자에 대한 질의응답 속도를 빠르게 제공하는 고성능, 예측할 수 없는 사용자를 수용하기 위한 고확장성, 급격한 사용자 증가에 장애가 발생하여도 지속적인 서비스를 제공하는 고가용성을 지원한다[2,3]. 회복기법은 비공유 데이터베이스 클러스터에서 고가용성을 위해 매우 중요하게 고려되고 있다. 일반적으로 데이터베이스 클러스터의 회복기법은 노드의 오류가 발생한 경우 노드 자신의 회복을 위한 로컬 로그와 클러스터 구성에 대한 회복을 위한 클러스터 로그를 생성하며, 이를 기반으로 해당 노드에서의 회복과정을 수행한다. 기존의 회복 기법은 하나의 레코드를 위해 다수의 갱신정보를 클러스터 로그에 유지함으로써 클러스터 로그의 크기가 증가되고, 회복노드에 전송되는 비용이 증가된다. 이는 회복노드에서 하나의 레코드에 대해 여러 번의 불필요한 갱신연산을 수행하여 회복시간이 증가되고, 전체적인 시스템의 부하를 증가시키는 문제를 발생시킨다.

본 논문에서는 비공유 공간 데이터베이스 클러스터에서 확장성 해싱 기반의 로그를 이용한 회복기법을 제안한다. 제안기법에서의 클러스터 로그는 레코드 키값을 이용한 확장성 해싱을 기반으로 레코드의 변경사항과 실제 데이터를 가리키는 포인터 정보로 구성된다. 확장성 해싱 기반의 클러스터 로그는 하나의 레코드를 위해 최신버전의 클러스터 로그 하나만을 유지하므로 클러스터 로그의 크기가 감소하고, 회복노드에 전송되는 비용이 감소한다. 회복노드는 하나의 레코드에 대해 한번의 갱신연산만 실행하므로 빠른 회복이 가능하다. 따라서 제안기법은 확장성 해싱 기반의 클러스터 로그를 이용하여 효율적인 회복처리를 수행하며, 시스템의 가용성을 향상시킨다.

본 논문의 내용 구성은 다음과 같다. 2장에서 관련연구를 다루고, 3장에서 제안기법의 기반이 되는 시스템을 기술한다. 그리고 4장에서 확장성 해싱 기반의 로그를 이용한 회복기법을 설명하며, 마지막으로 5장에서 결론을 내린다.

1) 본 연구는 대학 IT연구센터 육성·지원사업의 연구결과로 수행되었음

2. 관련연구

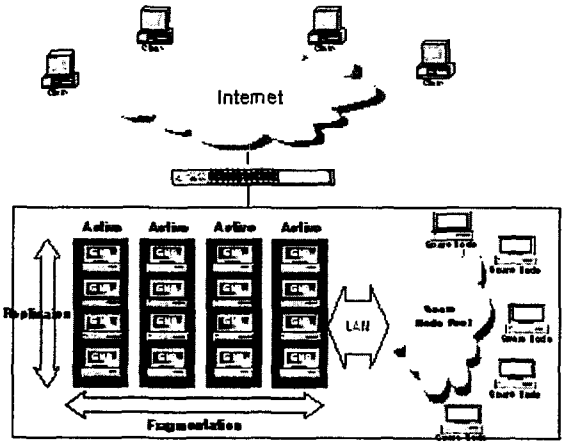
ClustRa는 메인 메모리 기반의 DBMS로 독립적인 질의 처리가 가능한 노드를 고속의 네트워크로 연결하고, 2개의 노드가 하나의 그룹을 유지하며, 그룹 내에서는 마스터 노드와 백업 노드가 존재한다. 그룹 내에 적용되는 분할정책과 복제정책의 사용으로 고성능, 고가용성을 지원한다. ClustRa에서의 회복기법은 노드 자체의 일관성 유지를 위한 내부로그와 노드간 일관성 유지를 위한 분산로그를 이용하여 회복을 수행한다. 메모리 기반의 분산로그는 연산이 발생하지 않더라도 항상 메모리에 유지하므로 메모리 낭비가 심하다. 또한 오류가 발생한 노드가 회복되기 전까지 빈번한 갱신연산이 발생할 경우 분산로그의 크기가 기하급수적으로 증가하고, 분산로그를 회복노드에 전송하기 위하여 하나의 큐를 유지한다. 즉, 갱신연산 처리 시 하나의 순차적인 번호를 갖는 분산로그를 생성하고, 큐에 넣게 된다. 따라서 모든 분산로그를 위한 순차번호를 생성할 때 동시성제어가 필요하며 이를 위해 락(Lock)을 사용하므로 성능이 저하되고, 큐에 유지한 분산로그를 순차적으로 보내게 되므로 전송비용이 증가한다. ClustRa에서의 회복과정은 내부로그를 이용하여 자체회복을 수행 후 분산로그를 이용하여 클러스터 구성을 위한 회복을 수행한다. 그러나 자체회복이 오래 걸릴 경우 분산로그의 크기가 증가하게 되고, 클러스터 구성에 대한 회복 시 큐에 유지한 분산로그를 순차적으로 전송하므로 회복시간이 느려지게 된다[5,6].

3. 시스템 개요

본 논문의 기반이 되는 전체적인 시스템 구조와 회복기법에 대해서 설명한다.

3.1 비공유 공간 데이터베이스 클러스터

본 논문에 기반이 되는 시스템은 비공유 공간 데이터베이스 클러스터인 GMS/Cluster 이며, GMS/Cluster의 구조는 [그림 1]과 같다.

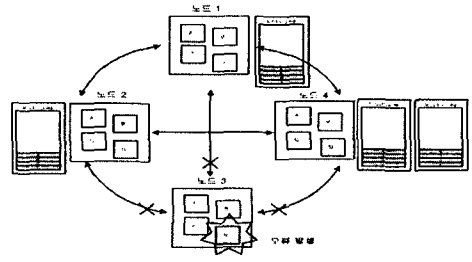


[그림 1] GMS/Cluster의 구조

비공유 구조의 공간 데이터베이스 클러스터는 기본적으로 노드와 그룹, 유휴 노드로 구성된다. 노드는 독립적인 DBMS 기능을 수행할 수 있으며, 각각의 노드들은 고속의 네트워크로 연결되어 있다[4]. 그룹은 노드들을 2~4개로 구성되며, 그룹 내에서는 하나의 마스터 노드와 여러 개의 백업노드가 존재 한다. 유휴 노드는 아무런 처리도 하지 않는 노드로 활동 노드의 붕괴나 온라인 확장 등에 사용된다. 그룹 내에 적용되는 복제 정책은 각 데이터에 대한 복사본을 다른 노드에서 유지하는 것으로 질의를 처리하던 노드에 오류가 발생할 경우 해당 노드의 데이터 복사본을 유지하는 다른 노드가 서비스를 지속할 수 있는 고가용성을 제공한다. 그룹 간에 적용되는 분할 정책은 하나의 데이터를 몇 개의 작은 데이터로 분할하여 각각 다른 노드에서 관리하는 것으로 대용량 데이터의 효율적인 관리와 데이터의 갱신연산에 대하여 동시 처리량을 향상시킴으로서 고성능을 제공한다. 노드에 부하가 집중되는 경우 유휴 노드가 클러스터에 추가되어 활성화 상태가 되고 기존에 분할 및 복제되어 있던 데이터를 재배치함으로써 예측할 수 없는 사용자를 수용할 수 있는 고확장성을 제공한다[7,8].

3.2 비공유 공간 데이터베이스 클러스터에서의 회복

GMS/Cluster는 클러스터 로그를 이용함으로써 결함 허용과 오류가 발생한 노드의 회복을 지원한다. 클러스터 로그는 특정 노드의 오류가 발생한 시점부터 기록하기 시작한다. 특정 노드의 정지를 즉시 감지하기 위하여 그룹을 형성할 때 각 노드간 망형 연결을 구축하여 어느 한 노드가 정지되면 그룹내의 나머지 노드들은 이를 바로 알게 되어 정지된 노드를 대체할 대체 노드를 결정한다. 결정된 노드는 오류가 발생한 노드를 대체함으로써 연속적인 서비스를 제공하고 그룹내 나머지 노드들은 클러스터 로그를 기록하기 시작하고, 오류가 발생한 노드가 재 시작을 하면 나머지 노드들은 자신이 갖고 있는 클러스터 로그를 전송함으로써 오류가 발생한 노드의 회복을 지원한다. 이를 그림으로 나타내면 [그림 2]과 같다.



[그림 2] 임의 노드의 오류 발생

오류가 발생한 노드의 회복과정은 정지된 노드가 회복되어 다시 그룹에 참여하는 경우와 유휴노드가 오류가 발생한 노드 대신 그룹에 참여 하는 경우가 있다. 오류가 발생한 노드가 회복되어 그룹에 참여하는 경우에 수행되는 과정은 로컬로그를 이용하여 자체회복을 수행한 다음 클러스터 구성의 회복을 위하여 그룹 내 다른 노드에게 자신이 살아났음을 알리고 클러스터 로그를 전송받아 회복을 수행한다. 유휴노드가 오류가 발생한 노드를 대신하여 그룹에 참여하는 경우에 수행되는 과정은 오류가 발생한 노드를 대신하여 유휴 노드중 하나를 선택하여 그룹의 일원으로 대체시키고, 유휴 노드에게 데이터베이스의 내용을 완전 복제 시킨 후 클러스터 그룹의 일원으로 서비스를 시작하게 된다. 유휴 노드가 그룹에 참여하여 정지된 노드의 역할을 대신하게 될 때 자신이 가지고 있던 모든 클러스터 로그를 삭제 시킨다. 또한 붕괴되었던 노드는 시스템 관리자에 의해 데이터베이스 초기화를 거쳐 유휴노드로 재 시작을 하게 된다[8].

4. 확장성 해싱 기반의 로그를 이용한 회복기법

본장에서는 확장성 해싱 기반의 클러스터 로그의 구조를 설명하고, 클러스터 로그의 기록 및 전송을 설명하며, 클러스터 로그를 이용하여 해당 노드에서의 회복을 설명한다.

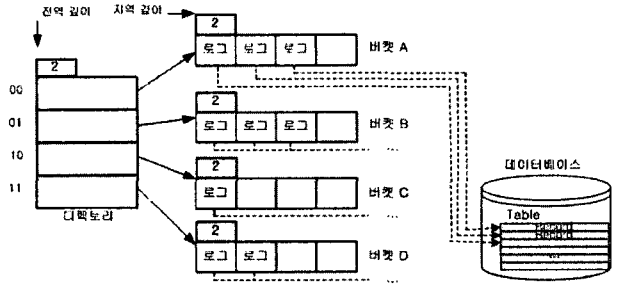
4.1 확장성 해싱 기반의 클러스터 로그 구조

클러스터 로그는 비공유 데이터베이스 클러스터에서 클러스터 구성의 회복에 필요한 로그로서 노드의 오류가 발생한 경우 로컬 로그와는 별도로 클러스터 로그를 생성하며, 이를 기반으로 해당 노드에서의 회복과정을 수행한다. 클러스터 로그의 구조는 [그림 3]와 같다.

Primary Key	RID	Log Content
기본키	레코드 식별자	레코드의 변경사항

[그림 3] 클러스터 로그의 구조

<Primary Key>는 레코드의 생성 시 주어지며 모든 레코드를 구별할 수 있는 유일키 이고, <RID>는 레코드의 물리적인 위치를 가리키는 식별자로서 연산이 발생한 레코드를 구별하기 위한 유일키로 사용되며, <Log Content>는 연산이 발생한 레코드의 변경사항을 기록한다. 다음 [그림 4]는 클러스터 로그를 효율적으로 관리하기 위하여 제안된 확장성 해싱 기반의 클러스터 로그의 구조이다.

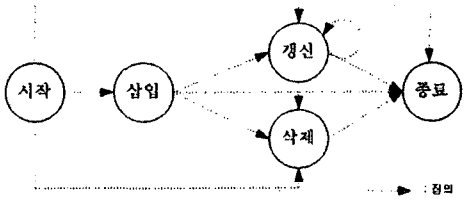


[그림 4] 확장성 해싱 기반의 클러스터 로그 구조

전역깊이는 디렉토리에 대한 인덱스로서 현재 디렉토리의 크기를 나타내고, 지역깊이는 해당 버켓의 오버플로우 발생 여부를 파악하기위해 유지되고, 디렉토리는 버켓을 가리키는 포인터를 저장하며, 버켓은 클러스터 로그를 유지한다. 자료 저장소는 연산이 발생하는 실제 데이터를 관리한다. 클러스터 로그의 관리는 연산이 발생한 해당 레코드의 RID를 해시함수에 적용하여 결과의 이진표현으로 디렉토리 주소를 나타내고, 디렉토리에 저장되어 있는 포인터가 가리키는 버켓에 클러스터 로그를 유지한다. 하나의 레코드에 삽입연산이 발생할 경우 새로운 클러스터 로그를 생성하여 기록하고, 삭제연산이 발생할 경우 해당 레코드의 RID를 이용하여 클러스터 로그를 검색하여 해당 클러스터 로그를 삭제하며, 갱신연산이 발생할 경우 해당 레코드의 RID를 해시함수에 적용하여 그 결과의 이진표현에 해당하는 디렉토리를 검색하고 디렉토리에 저장되어 있는 포인터가 가리키는 버켓에 유지되어 있는 클러스터 로그를 찾아 해당 클러스터 로그를 최신의 변경내용 하나만을 저장한다.

4.2 클러스터 로그의 최신버전 기록 및 전송

노드의 오류가 발생한 경우 해당 노드에서의 회복과정을 수행하기 위해 그룹 내의 다른 노드들은 클러스터 로그를 생성하여 기록하기 시작한다. 클러스터 로그의 갱신 내용은 다수의 갱신 연산이 발생하여도 최신의 변경내용 하나만을 유지한다. 클러스터 로그 생성 후 발생하는 연산은 삽입연산, 갱신연산, 그리고 삭제연산이 있다. 다음 [그림 5]는 클러스터 로그 생성 후 발생하는 다수의 연산과정에 대한 설명이다.



[그림 5] 클러스터 로그 기록 시 다수의 연산

클러스터 로그 생성 후 발생되는 다수의 연산과정은 삽입, 갱신, 삭제연산이 발생할 수 있으며, 삽입연산은 갱신, 삭제연산이 발생할 수 있고, 갱신연산은 갱신과 삭제연산이 발생할 수 있다. 시작은 클러스터 로그를 생성하며, 종료는 클러스터 로그 기록의 끝을 나타낸다.

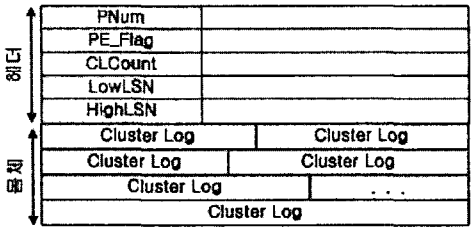
[그림 6]은 클러스터 로그 생성 후 발생하는 다수의 연산과정을 거쳐 최신 버전의 변경내용 하나만 클러스터 로그에 유지하는 결과를 나타낸다.

	최신버전 클러스터 로그의 내용		
질의	INSERT	UPDATE	DELETE
INSERT	-	-	-
UPDATE	UPDATE	UPDATE	-
DELETE	IGNORE	DELETE	DELETE

[그림 6] 클러스터 로그의 최신버전 갱신내용

삽입연산은 새로운 클러스터 로그를 생성하고, 갱신연산은 클러스터 로그를 갱신하며, 삭제연산은 클러스터 로그를 삭제시킨다. 삽입연산 이후 발생하는 갱신연산은 클러스터 로그를 갱신하고, 삭제연산은 무시한다. 갱신연산 이후 발생하는 갱신연산은 클러스터 로그를 갱신하고, 삭제 연산은 클러스터 로그를 삭제한다. 하나의 레코드에 다수의 연산이 발생하여도 최신 버전의 변경내용을 유지함으로서 클러스터 로그의 크기를 줄일 수 있다.

클러스터 로그의 전송은 고속의 네트워크를 통해 패킷 단위로 전송하며, 자료구조는 다음 [그림 7]과 같다.



[그림 7] 클러스터 로그의 전송 자료구조

클러스터 로그는 헤더와 몸체로 구성되고, 헤더는 클러스터 로그를 전송하는데 필요한 정보들을 유지하며, 몸체는 변경된 실제 데이터를 유지하고 있는 클러스터 로그를 전송하는 것으로 클러스터 로그를 순차적으로 패킷의 몸체 크기만큼 복사한다. 헤더의 구성에 필요한 정보는 다음과 같다. <PNum>는 클러스터 로그 전송 패킷번호를 나타내는 것으로 패킷의 직렬성을 보장하고, <PE_Flag>는 클러스터 로그 전송 패킷의 마지막 여부를 알려 주는 플래그 이다. <CLCount>는 패킷에 존재하는 클러스터 로그의 수를 나타내며, <LowLSN>는 현재 패킷에서 가장 작은 클러스터 로그 번호로 패킷내의 클러스터 로그의 시작 번호와 같으며, <HightLSN>는 현재 패킷에서 가장 큰 클러스터 로그 번호로 패킷내의 로그 끝 번호와 같다.

발생되는 연산에 따라 전송되는 클러스터 로그의 정보가 달라지며, 연산에 따른 클러스터 로그 전송 정보는 다음 [그림 8]과 같다.

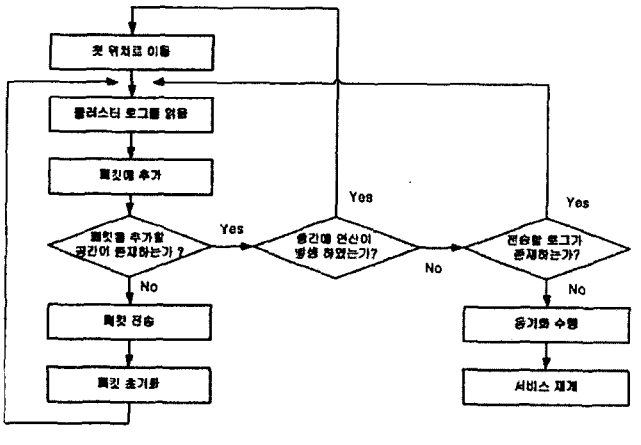
	Cluster Log		
INSERT	Primary Key	Data	
UPDATE	Old Primary Key	New Primary Key	Data
DELETE	Primary Key		

[그림 8] 연산에 따른 클러스터 로그 전송 정보

삽입연산의 클러스터 로그는 Primary Key와 실제 Data를 전송하고, 갱신연산의 클러스터 로그는 이전 데이터의 Primary Key와 새롭게 갱신된 데이터의 Primary Key와 갱신된 새로운 데이터를 전송하며, 삭제연산의 클러스터 로그는 Primary Key를 전송한다.

4.3 클러스터 로그를 이용한 회복

오류가 발생한 노드가 회복되는 경우 로컬로그를 이용하여 자체회복을 수행하고, 자체회복이 끝난 노드는 클러스터 구성에 대한 회복을 위해 그룹내 다른 노드들과 연결하여 클러스터 로그 전송을 요청한다. 마스터 노드는 오류가 발생한 노드에게 클러스터 로그를 패킷 단위로 전송하고, 패킷을 전송 받은 회복노드는 해당 패킷에 대한 확인응답을 전송하며, 회복노드로부터 확인응답을 받은 패킷은 삭제된다. 손실된 패킷은 재전송하고, 회복노드에서 패킷번호를 기반으로 정렬하여 순차적으로 수행한다. 패킷 전송 중 연산이 발생할 경우 마스터 노드에서 해당 연산을 수행한 후 클러스터 로그를 생성하며, 전송되어야 할 클러스터 로그의 크기가 하나의 패킷 크기보다 작아지면 발생하는 연산에 대한 처리를 잠시 멈추는 일시적인 트랜잭션 대기 상태가 되고, 대기상태 동안 발생하는 연산은 큐(Queue)에 유지하며, 남아 있는 클러스터 로그를 회복노드로 전송한다. 회복노드로 모든 클러스터 로그가 전송되면 트랜잭션 대기상태 동안 큐에 유지되어 있는 연산을 회복노드로 전송하여 수행한다. 이와 같은 마스터 노드와 회복노드 사이에 동기화 과정을 수행하여 오류가 발생한 노드의 회복을 완료하여 정상적인 서비스를 재개한다. 마스터 노드에서 오류가 발생하여 회복과정을 수행한 경우 동기화 과정을 마친 후 회복된 노드가 다시 마스터 노드가 되어 정상적인 서비스를 재개한다. 다음 [그림 9]는 회복을 수행하기 위하여 클러스터 로그를 전송하는 과정이다.



[그림 9] 클러스터 로그 전송

전송되어진 클러스터 로그는 발생된 연산을 기반으로 실제 Data를 반영한다. 삽입연산은 전송되어진 Primary Key와 실제 Data를 데이터베이스에 반영하고, 갱신연산은 전송되어진 Old Primary Key를 이용하여 이전의 데이터를 검색하여, 새로운 New Primary Key로 바꾸고, 이전 Data를 전송되어진 실제 Data로 갱신한다. 삭제연산은 전송되어진 Primary Key를 이용하여 해당 데이터를 검색하여 삭제 시킨다.

5. 결론

본 논문에서는 비공유 데이터베이스 클러스터에서 확장성 해싱 기반의 로그를 이용한 회복기법을 제안하였다. 제안기법에서의 클러스터 로그는 레코드 키값을 이용한 확장성 해싱을 기반으로 해당 클러스터 로그의 위치를 빠르게 검색하여 레코드의 변경사항과 실제 데이터를 가리키는 포인터 정보로 구성된다.

확장성 해싱 기반의 클러스터 로그는 하나의 레코드를 위해 최신버전의 클러스터 로그 하나만을 유지하므로 클러스터 로그의 크기가 감소하고, 회복노드에 전송되는 비용이 감소한다. 또한 회복노드는 하나의 레코드에 대해 한번의 갱신연산만 실행하므로 빠른 회복이 가능하다. 따라서 제안기법은 확장성 해싱 기반의 클러스터 로그를 이용하여 효율적인 회복처리를 수행하여, 시스템의 가용성을 향상시킨다.

향후연구는 하나의 레코드에 대한 회복노드에서 롤백연산을 효율적으로 관리할 수 있는 방법이 요구된다.

참고문헌

[1] R. H. Guting, and et. al, "A Foundation for Representing and Querying Moving Objects", ACM Transactions on Database Systems, Vol. 25, No. 1, pp. 1-42, 2000

[2] Roger Bamford, Rafiul Ahad, Angelo Pruscino, "A Scalable and Highly Available Networked Database Architecture", Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999

[3] Roel Vandewall, "Database Replication Prototype", Masters thesis, Department of Mathematics and Computer Science, University of Groningen, The Netherlands, 2000

[4] David J. DeWitt and Jim Gray, "Parallel Database Systems : The Future of Database Processing or a Passing Fad?", Microsoft. http://research.microsoft .com/~gray/CacmParallel-DB.doc

[5] Svein-Olaf Hvasshovd, Oystein Torbjornsen, Svein Erik Bratsberg, "The ClustRa Telecom Database: High Availability, High Throughput, and Real-Time Response", Proceedings of the 21st VLDB Conference, 1995

[6] Oystein Torbjornsen, Svein-Olaf Hvasshovd, Young-Kuk Kim, "Towards Real-Time Performance in a Scalable, Continuously Available Telecom DBMS", ClustRa, 2001

[7] Yong-Il Jang, Chung-Ho Lee, Jae-Dong Lee and Hae-Young Bae, "Improved On-line Scaling Scheme in a Scalable and Highly Available Database", In Proceedings of the PDPTA' 02 Conference, Las Vegas, Nevada, USA, 2002

[8] 유병섭, 김명근, 김재홍, 배해영, "GMS/Cluster 설계 및 구현", 개방형 지리정보 시스템 학회, p. 225 - 231, 2003