임베디드 DBMS에서 바이트패킹과 Lempel-Ziv 방법을 혼합한 무손실 벡터 데이터 압축 기법

A Lossless Vector Data Compression Using the Hybrid Approach of BytePacking and Lempel-Ziv in Embedded DBMS

문 경 기* 주 용 진** 박 수 홍***
Gyeong Gi Moon Yong Jin Joo Soo Hong Park

요 약 최근 무선 인터넷 환경의 발달로 실시간 교통정보안내와 지도를 통해 목적지까지의 경로 안내를 해주는 차량 항법 서비스 등 공간 데이터를 활용한 위치기반서비스가 증가되고 있다. 하지만, 현재 시스템 환경에서는 대용량의 공간 데이터를 파일 시스템 기반으로 관리하기 때문에 실시간적인 데이터 저장과 관리측면에서 많은 제약을 가진다. 이를 보완하기 위해 임베디드 데이터베이스를 바탕으로 대용량의 공간 데이터를 구조적으로 관리할 수 있는 연구가 요구된다. 이에 본 연구는 임베디드 시스템에서 대용량의 공간 데이터의 효율적인 저장을 위해 데이터베이스에 적용 가능한 바이트패킹과 Lempel-Ziv 압축기법을 혼합 개선한 무손실 압축 기법을 제시하고자 하였다. 이렇게 제시된 공간 데이터 압축 기법을 실제 대도시권 데이터(서울·인천)에 적용하여 실험해 보고 동일 데이터에 대하여 실험을 통해 재구성이 되기까지의 질의 처리 시간을 분석을 통해 선행 연구에서 제시한 방법을 적용한 결과와 비교 하였다. 연구결과로 본 연구에서 제시된 압축 방법이 높은 위치 정확도를 요구하는 데이터에 대해 더 나은 성능을 보이는 것을 확인 할 수 있었다.

키워드 : 벡터 데이터 압축, 위치기반서비스, 임베디드 DBMS, 모바일 GIS

Abstract Due to development of environment of wireless Internet, location based services on the basis of spatial data have been increased such as real time traffic information as well as CNS(Car Navigation System) to provide mobile user with route guidance to the destination. However, the current application adopting the file-based system has limitation of managing and storing the huge amount of spatial data. In order to supplement this challenge, research which is capable of managing large amounts of spatial data based on embedded database system is surely demanded. For this reason, this study aims to suggest the lossless compression technique by using the hybrid approach of BytePacking and Lempel-Ziv which can be applicable in DBMS so as to save a mass spatial data efficiently. We apply the proposed compression technique to actual the Seoul and Incheon metropolitan area and compared the existing method with suggested one using the same data through analyzing the query processing duration until the reconstruction. As a result of comparison, we have come to the conclusion that suggested technique is far more performance on spatial data demanding high location accuracy than the previous techniques.

Keywords : Vector Data Compression, LBS, Embedded DBMS, Mobil GIS

1. 서 론

최근 유선 인터넷 환경에서 제공되고 있는 다양한 위치기반서비스(LBS : Location Based System)는 무선 인터넷 환경이 발전함에 따라 GPS와 차량용 단말기와 같은 공간 데이터를 활용하여 제공되

† 이 연구는 국토해양부 첨단도시기술개발사업 - 지능형국토정보기술혁신 사업과제의 연구비지원(07국토정보-B01)에 의해 수행되었음.

* 엠앤소프트 기술연구소 주임연구원, starmkk@gmail.com
** 서울시립대학교 도시과학연구원 융합도시연구센터 연구교수 yjjoo75@uos.ac.kr(교신저자)
*** 인하대학교 지리정보공학과 교수 shpark@inha.ac.kr

는 사례가 증가하고 있다[1]. 지도를 제공하는 서비스 또는 목적지까지의 경로 안내를 해주는 네비게이션 서비스 등으로 구성된 다양한 검색서비스는 편리한 교통정보안내를 비롯한 생활의 유익한 정보를 제공해 준다.

현재 공간 데이터를 활용한 다양한 서비스는 높은 위치 정확도 유지를 바탕으로 빠른 검색 속도를 제공하기 위해 데이터 처리 기술을 개발하여 사용하고 있다. 특히 미국의 GDF, 일본의 KIWI 등 상용화 된 차량 항법용 데이터 구조는 파일 기반의 미디어 포맷으로 설계 되어있다[8]. 하지만, 이러한 파일 기반의 데이터 관리는 데이터에 대한 변경이 발생 했을 때 실시간적인 데이터 동기화와 관리 기능에 많은 제약사항을 지니게 된다[7]. 이런 문제점으로 인해 주기적으로 웹(Web)이나 CD와 같은 저장 매체의 배포를 통해 전체 데이터를 업데이트 하는 방법을 사용하고 있다[17].

대용량 GIS 공간 데이터의 구조와 관리 측면에서 데이터베이스와 관련한 연구는 크게 공간 인덱스와 압축에 관한 주제로 분류될 수 있다. 즉, 공간 인덱싱에 관한 연구는 주로 쿼드트리(Quadtree)나 알트리(R-tree)와 같은 다차원 공간 인덱스를 실시간적인 사용자의 검색을 지원하기 위해 최적화된 형태로 검색의 성능을 제공하는 연구이다[2, 10]. 또 다른 주제는 공간 데이터를 압축하는 부문으로 저장공간과 질의 처리과정을 보다 효율적으로 관리하여 응용 시스템에게 최적화된 데이터 관리 기능을 제공하는 것이다[3, 6, 12, 16]. 이에 본 연구에서는 데이터베이스를 활용하여 공간 데이터를 효율적으로 압축하기 위한 부문으로 벡터 맵 데이터를 압축할 수 있는 무손실 압축 방법을 설계, 제시하고자 한다. 즉, 기존 데이터 중복성을 초래하는 기존 바이트패킹 압축 방법의 문제점을 보완하여 객체 별로 적용함으로서 압축된 효과로 인한 질의 처리의 향상을 가능하게 하는 향상된 압축 알고리즘 개발을 목적으로 한다.

공간 데이터에 대한 압축 기술은 많은 양의 지도 데이터를 모바일 기기에 저장할 수 있게 하며, 문자 데이터와 같은 다른 종류의 데이터의 저장을 위한 공간을 마련할 수 있다. 데이터의 전송 시에도 전송에 필요한 비용을 최소화시켜, 보다 빠른 시간에 많은 양의 공간 데이터를 전송하게 한다.

이를 위한 연구의 내용과 방법은 다음과 같다. 우선, 관련 이론 및 선행 연구 검토를 통해 연구에서 설계·제시할 압축 알고리즘을 분석하고 본 연구에 적용하기위한 시사점을 도출한다. 또한 현행 연구의 분석을 통해 파악된 문제점과 한계를 극복 할 수 있는 벡터 맵 데이터 압축 기법을 설계한다. 둘째, 제시된 압축 알고리즘을 검증하기 위한 실험 단계로 실험 대상지역을 선정하고 데이터를 구축한다. 각각의 압축 방법에 대한 성능 실험은 디스크 기반의 모바일용 관계형 데이터베이스 SQLite을 사용한다. SQLite는 질의 처리 면에서 기존 MySQL, PostgreSQL보다 성능이 우수하고, 다양한 플랫폼, Native C/C++ 표준 API와 프로그램 인터페이스를 지원하므로 상호운용이 가능한 소프트웨어의 개발과 복잡하고 다양한 공간 질의 구현이 용이한 특징을 가진다. 셋째, 실험을 통하여 얻어진 결과를 분석한다. 크게 질의 처리에 관한 관점에서 압축이 된 데이터를 원본 데이터로 복원되는 처리 시간을 분석한다. 또한 선행 연구에서 제시된 압축 알고리즘과 본 연구의 방법론을 비교 분석을 통하여 제안된 압축 방법의 성능을 검증한다. 마지막으로 분석된 결과를 바탕으로 제안된 압축 방법의 타당성과 실제 적용 가능성을 검토하고, 더불어 제안된 압축 방법의 문제점과 향후 연구 방안에 대하여 제시한다.

2. 관련 연구

공간 데이터베이스에 공간 데이터를 저장하기 위한 방법은 대표적으로 OGC(Open Geospatial Consortium)의 OpenGIS Implementation Specification for Geographic Information-Simple feature access 통해 제안하고 있지만[13,14,15], 공간 객체의 좌표에 대한 정보를 8바이트(byte) 단위로 저장을 하고 있다. 이는 압축을 고려하지 않는 구조로서 많은 저장 공간이 필요하게 되며, 질의 대상이 되는 공간 객체의 크기에 따라 질의 처리가 오래 걸린다. 따라서 가변적인 길이를 갖는 공간 객체를 압축하여 저장할 수 있는 방법을 통해 저장구조와 질의 처리를 효율적으로 처리할 수 있는 방법들이 제안되고 있다.

2.1 Precision 압축

Precision 압축 방법이란 객체의 좌표 정보에 대한 정밀도를 조절함으로서 압축의 효과를 얻는 프

로그래밍 기법이다. 원본 데이터를 실제적으로 정의할 때에는 Precision 값에 의해 정수형으로 바뀌어서 데이터를 저장하게 된다. 그 결과 좌표 정보 하나마다 필요한 8바이트의 저장 공간을 4바이트의 크기로 저장을 할 수가 있게 된다. 해당 데이터를 출력 할 때에는 Precision 값으로 나누어 원래의 데이터로 복원을 시켜 응용 프로그램에 제공한다.

2.2 바이트 패킹(BytePacking) 압축 방법

바이트패킹 압축 방법은 공간 데이터를 최소 단위인 비트(bit) 단위로 저장을 하여 압축된 데이터를 원본 데이터로 완벽하게 복원하는 무손실 압축 방법으로서 상용 공간 데이터베이스(Spatial DBMS)인 ArcSDE에서도 사용되는 방법이다[9]. 바이트패킹 압축 방법은 공간 데이터의 저장 공간을 효율적으로 사용하기 위하여 실제 좌표 정보를 저장하지 않고, 공간 객체가 처음 시작하는 시작점을 기준으로 차분 벡터(Differential Vector)의 값을 구해 상대 좌표로 변환을 시켜 저장한다. 즉, 실제 좌표들을 공간 객체의 시작점을 기준으로 상대 좌표로 변환 시킨다. 그 후 변환된 상대 좌표에 대해 압축 방법을 적용하여 비트 단위로 저장을 한다. 또한, 컨트롤 비트(Control bit)와 사인 비트(Sign bit)는 공간 데이터를 위한 추가적인 정보로 사용되는 플래그 정보로써. 컨트롤 비트는 인접해 있는 바이트가 연속적인 정보일 때 1이라는 값을 통해 바이트에 대한 연관성 정보를 나타낸다. 사인 비트는 차분 벡터 값의 부호 정보를 관리한다. 즉, 음수일 경우는 1, 양수일 경우는 0으로 부호 정보를 저장한다. 리틀 엔디안(Little Endian)의 형식으로 되어 있기 때문에 복원시에는 오른쪽부터 진행이 되고, 압축 시에는 왼쪽부터 진행이 되는 것이 특징이다.

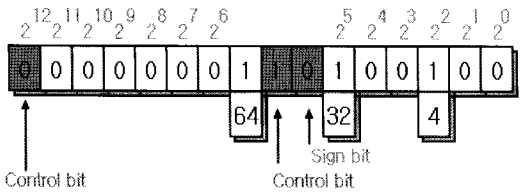


그림 1. 바이트 패킹을 이용한 정보(100) 저장

2.3 Lempel-Ziv 압축 방법

Lempel-Ziv는 공간 데이터 뿐만 아니라 동영상, 문서, MP3와 같은 파일 시스템에서도 적용 가능한 압축 방법이다. Lempel-Ziv 압축 방법은 현재의 위치에서 중복이 되는 패턴이 존재한다면 그것에 대한 상대적 위치와 그 패턴의 길이를 구해서 패턴을 대치하는 방법이다. 아래 표 1은 원래 문자열을 Lempel-Ziv 압축 방법에 의해 데이터를 압축 한 것을 나타낸다. 원래 문자열에서 "03 02 04 8a d1"라는 패턴이 뒤에서 반복되는 것을 볼 수 있다. 이 때 뒤의 패턴을 앞에 9번째 데이터부터 5번까지는 같은 정보라는 코드를 삽입함으로서 압축을 할 수도 있고, 복원도 할 수 있다. Lempel-Ziv 압축 방법은 그 실제 구현에서 여러 가지 다양한 방법이 있다. 즉, 가장 대표적인 방법은 정적 사전(Static Dictionary)법과 동적 사전(Dynamic Dictionary)법이다. 정적 사전법은 출현될 것으로 예상되는 패턴에 대한 정적 테이블을 미리 만들어 두었다가 그 패턴이 나올 경우 정적 테이블에 대한 참조를 하도록 하여 압축하는 방법이다. 이 방법은 압축하고자 하는 파일의 내용이 예상 가능한 경우에 매우 좋은 방법이다. 동적 사전법은 파일을 읽어 들이는 과정에서 패턴에 대한 사전을 만든다. 즉 동적 사전법에서 패턴에 대한 참조는 이미 그 전에 파일 내에서 출현한 패턴에 제한된다. 동적 사전법은 파일을 읽어 들이면서 사전을 구성해야 하는 부담이 생기기 때문에 속도가 느리다는 단점이 있으나, 임의의 파일에 대해 압축률이 좋은 경우가 많다(이재규, 1996).

표 1. Lempel-Ziv 압축 방법

데이터	
원본 데이터	0a 03 02 04 8a d1 08 2d 59 47 03 0204 8a d1
압축 데이터	0a 03 02 04 8a d1 08 2d 59 47 <9, 5>

2.4 연구의 착안점 도출

선행 연구로 검토한 바이트패킹 압축 방법은 공간 데이터의 압축 시 최소 단위인 비트로 데이터를 저장하게 된다. 하지만 실제적으로 객체가 가지고 있는 차분 벡터 값이 같을 때마다 비트 단위로 저장되는 압축 정보가 반복이 되어서 저장하게 된다. 이 특성 때문에 객체가 가지는 포인트 수가 많을수록 중복 저장을 하는 경우가 많아지게 된다. 이러한 문제점으로 인해 질의 처리를 수행하는데 있어서 중복된 정보의 수만큼 복원하기까지의 비용이 추가

적으로 요구된다. 이러한 문제점을 보완 하고자 본 연구에서는 기본적인 바이트패킹 압축 방법으로 각각의 차분 벡터를 추출하여 반복되어 저장이 되는 패턴을 찾아 그 위치에 대한 정보를 헤더로서 관리를 하는 압축하는 방법을 수행한다. 이 때 반복된 정보에 대한 절대 위치를 보완된 Lempel-Ziv 압축 방법으로 다시 재압축을 한다. 이는 Lempel-Ziv 압축 방법은 압축된 데이터가 복원 시 데이터의 크기에 따라 O(N)의 효율로 복원되는 문제가 있기 때문이다. 따라서 이를 보완하고자 헤더 정보를 통해 O(1)의 효율로 복원 정보를 찾는 것을 제공하기 위해 Lempel-Ziv 압축 방법을 보완하였다. 결과적으로 바이트패킹 압축 방법의 중복 저장에 대한 문제점을 보완된 Lempel-Ziv 압축 방법으로 다시 재압축을 하는 혼합 방식을 설계한다.

3. 백터 데이터 압축 기법 설계

3.1 벡터 데이터 변환

벡터 데이터 압축 기법 설계의 첫 번째 단계는 설계에 사용될 저장 모델을 선정하고, 선정된 모델을 이용하여 공간 좌표계에서 객체의 절대적인 위치를 나타낼 수 있는 차분 벡터를 구하였다. 이 과정에서는 데이터 압축의 과정에서 사용될 데이터 모델로 OGC에서 제안하는 기하 모델을 수정하여 사용하였다. OGC는 공간 데이터 분야에서 가장 공인된 표준화 기관이지만, 공간 데이터를 저장하기 위해서는 상대적으로 많은 양의 저장 공간이 필요하는 문제가 있다. 따라서, 기본적인 내용을 준수하되 공간 데이터를 위한 최소화된 정보만을 저장할 수 있도록 데이터 모델을 재구성 하였다. 아래 표 2는 연구에서 사용된 벡터 저장 모델이다.

표 2. 벡터 데이터 구조체

데이터	데이터 타입	바이트
데이터 타입	Unsigned Char	1
포인트 수	Unsigned Long	4
x좌표(시작점)	Long	4
y좌표(시작점)	Long	4
길이	Unsigned Short	2
상대좌표	Bit	가변

OGC의 Simple Feature Specification For SQL v1.1[15]에서 제안하는 모델과는 달리 엔디안(Endian) 정보를 제외시키고 시작점을 기준으로 상대 좌표로 변환하여 저장한 것이 특징이다.

차분 벡터는 각각의 점 좌표에 대한 상대적인 위치를 표현하는 벡터이다. 차분 벡터를 계산 하는 방법은 i번째 저장된 점의 위치를 i-1번째 점과의 차이를 이용하여 계산하는 방법과, i번째 저장된 점의 위치를 첫 번째 위치에 저장된 점과의 차이를 이용하여 계산하는 방법이 있다. 차분 벡터를 계산하는 방법에 따라 표 3과 같은 표기를 이용한다.

표 3. 차분 벡터 표현법

	표현	
이전 점 좌표와 차이	$\Delta_{i,i-1}$	Delta(i, i-1)
첫 번째 점 좌표와 차이	$\Delta_{i,0}$	Delta(i, 0)

$\Delta_{i,0}$는 하나의 공간객체에서 시작점 좌표를 기준으로 나머지 각 점의 좌표를 가리키고 있는 벡터의 집합이다. 이와는 달리 $\Delta_{i,i-1}$의 경우에는 기준이 되는 점이 각 객체의 시작점이 아닌 위치를 표현하고자 하는 점의 이전에 저장되는 점이 된다. 위의 두 가지 방법을 이용하여 그림2의 폴리곤으로부터 차분 벡터를 추출해 보면 표 4와 같다.

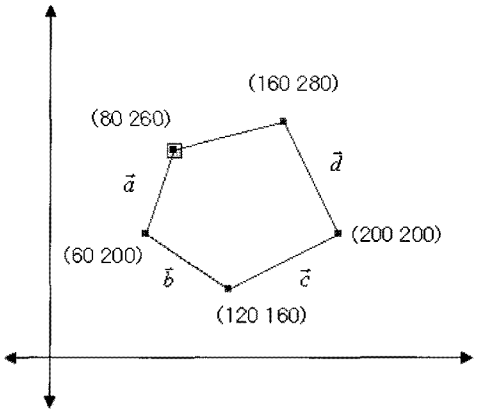


그림 2. 차분 벡터 예 -
POLYGON ((80 260, 60 200, 120 160, 200 200, 160 280, 80 260))

표 4. 차분 벡터 추출

	차분 벡터
$\Delta_{i,i-1}$	(-20 -60), (60 -40), (80 40), -40 80)
$\Delta_{i,0}$	(-20 -60), (40 -100), (120 -60), (-40 80)

3.2 벡터 데이터 압축 알고리즘 설계

연구에서 제안하는 압축 방법은 바이트패킹 압축 방법으로 차분 벡터를 압축하고, 압축된 데이터를 읽어 들이면서 반복되는 정보를 찾아내 다시 재압축을 하는 방법을 사용한다. 선형 검색을 통해 압축을 하기 때문에 걸리는 시간이 오래 걸린다는 점이 단점이지만, 객체마다 서로 다른 중복되는 정보를 찾을 수 있는 장점이 있다. 또한 복원에 관한 정보를 해쉬 테이블로 관리하기 때문에 복원 시 압축에 관한 정보를 O(1)의 속도로 검색하여 복원에 관한 속도를 높일 수 있다. 압축 방법에 대한 알고리즘은 표 5와 같다.

표 5. 개선된 압축 알고리즘

Designed Algorithm
1) initialize(code table); 2) initialize(hash table); 3) buffer=string consisting of character from the file. 4) for each buffer do 5) _string= buffer[i] 6) search for _string in the code table; 7) if found 8) then begin 9) store in the hash table 10) ret = compress(buffer,i,matchlength,pattern); 11) buffer += ret; 12) buffer=string consisting of one character c; 13) matchlength++; 14) else 15) then begin 16) send the code associated with buffer; 17) assign a code to _string; 18) store both in the code table; 19) buffer=string consisting of one character c; 20) end 21) send the code associated with buffer 22) end

우선적으로 텍스트 파일의 첫 부분인 문자에 하나의 코드를 할당하고 (압축 알고리즘 3행) 코드 표에 저장한다. 다음으로 루프를 실행하고 파일로부터 한 번에 한 문자씩 읽어 들인다(제 4행). 이 때는 파일로부터 받은 문자들과 연쇄시켜 만들어진 버퍼 스트링을 이용한다. 그 후 루프의 각 패스에서는 한 문자를 읽고 버퍼 스트링에 덧붙여 새로운 임시 스트링을 만든다(제5행). 만약 그 임시 스트링을 전에 만났었다면(즉 코드 표에 있다면), 해당 버퍼 위치의 값을 해쉬 테이블에 저장하고 길이 정보를 증가시킨다(제7행, 제13행). 하지만 임시 문자 행이 코드 표에 없다면, 임시 스트링에 코드를 할당하여 코드와 스트링을 코드 표에 저장한다(제19행). 끝으로 버퍼 스트링을 읽은 문자로 다시 초기화한다(제21).

3.3 공간 데이터 저장 스키마 설계

본 절은 압축 방법에 따른 질의 처리를 실험하기 위해 설계된 관계형 테이블을 정의한다. 맵 데이터 저장 구조는 관계형 인덱싱을 지원하기 위한 것으로서 SQL만으로도 공간 질의가 가능하도록 설계하였다. 공간 인덱스 테이블은 공간 질의 시 인덱스 역할을 하는 테이블로서 해당 객체의 MBR 정보와 공간 인덱스인 Fixed Grid Index의 Grid ID를 저장한다. Grid ID는 객체의 MBR이 지나는 모든 그리드 영역을 계산해 저장을 하기 때문에 객체 하나는 최대 4개까지 중복이 되어서 저장이 된다. 이는 SQL의 IN연산자를 지원하기 위함으로서, 보다 질의 처리를 향상 시킬 수 있는 효과를 얻을 수 있다. 표 6은 S테이블의 스키마를 나타낸 것으로 공간 질의 시 필터링(Filtering)과 리파인(Refine)에 대한 기능을 지원한다.

표 6. 공간 인덱스 테이블 스키마

필드명	필드 내용	필드타입	길이	제약조건
ID	Geometry ID	VARCHAR	9	PK
Level	표현 레벨	CHAR	1	NN
MinX	MBR	INTEGER	4	NN
MinY		INTEGER	4	NN
MaxX		INTEGER	4	NN
MaxY		INTEGER	4	NN
GX	Grid ID	INTEGER	4	NN
GY		INTEGER	4	NN

피쳐 테이블은 실제적인 데이터가 저장되는 테이블(표 7)로서 객체에 대한 클래스 코드로 색상, 디스플레이 관련정보를 저장한다.

표 7. 피쳐 테이블 스키마

필드명	필드 내용	필드타입	길이	제약조건
GID	Geometry ID	VARCHAR	9	PK
ClassCode	표현코드	VARCHAR	8	NN
Shape	객체	BLOB		NN

3.4 공간 질의(Sptial Quiery) 구조의 설계

관계형 데이터베이스를 바탕으로 공간 질의를 하기 위해서는 최적화된 SQL구문이 필요하다. 표 8은 공간 검색을 위한 SQL의 설계를 나타낸 것으로서 인라인 뷰를 사용하였으며, 필터링 단계와 정제 단계로 공간 검색을 하도록 설계하였다. 4행과 5행은 필터링 단계로서 해당 Grid ID를 속성으로 가지는 객체에 대해 후보 셋(Candidate set)을 형성하게 된다. 형성된 후보 셋에 대해 다시 리파인이 이루어지는데 7행과 8행이 그 역할을 하게 된다. 검색된 객체는 10행과 11행의 네추럴 조인(Natural-Join)에 의해 F테이블에 저장된 객체를 검색하게 된다.

표 8. 공간 검색을 위한 공간 질의 구문

SQL구문
1) select BgF562615.Shape, BgClassSymbol.ClassCode,BgClassSymbol.RGBColor 2) from (select distinct(ID) as BgSID 3) from BgS562615 4) where GX IN (280,281) and 5) GY IN (519,520) and 6) level = 0 and 7) BgS562615.minx <= 303275 and BgS562615.minY<= 551626 and 8) BgS562615.maxx >= 302564 and BgS562615.maxy >=551093 9)) as BgStable, BgF562615, BgClassSymbol 10) where BgStable.BgSID = BgF562615.GID and 11) BgF562615.ClassCode = BgClassSymbol.ClassCode 12) Order by BgClassSymbol.DispOrder

4. 알고리즘의 적용 및 분석

제시된 압축 방법의 성능을 정량적으로 평가하고, 선행 연구와의 비교를 통하여 타당성과 적용 가능성을 판단하고자 다음과 같은 실험을 하였다. 첫 번째 단계는 실험에 적합한 데이터를 선정하는 것이다. 실험에 사용된 데이터는 데이터의 수에 따라 모든 도엽이 구축된 전체 데이터이고, 또 다른 데이터는 특정 지역의 도엽 데이터를 선정하였다. 이 과정에서 선정된 두 종류의 데이터를 기존의 압축 방법과 연구에서 제시된 압축 방법을 적용하여 각각의 결과를 도출하였다. 마지막 단계는 결과물을 분석하여 연구의 타당성과 실제 데이터에 대해 적용 가능성을 부여하였다. 이 과정에서 선정된 두 데이터에 적용된 압축 방법에 대한 압축률, 질의 처리 시간을 비교 분석하였다.

4.1 실험 데이터

제안된 벡터 데이터 압축 방법에 대한 성능 평가를 위하여 원본 데이터, 바이트패킹 압축 방법, Lempel-Ziv 압축 방법과의 비교 실험을 수행 하였다. 각각의 방법에 대하여 두 번의 실험을 하였는데, 첫 번째 데이터로는 인천광역시, 수도권 지역이며 두 번째 데이터는 인천광역시 지역의 데이터이다. 두 데이터를 위해 1:1000 수치지도에서 면으로 표현되는 지형, 지물과 선으로 표현되는 도로, 행정경계, 부지계 등에 대해 레이어를 추출하여 실험에 적용하였다. 수치지도에서 추출된 데이터는 256180개의 폴리곤 형태를 가지며, 이를 구성하는 포인트의 수는 2214391개이다. 라인의 경우는 381076개이며, 포인트의 수는 1884702개이다. 두 번째 실험에서는 전체 데이터의 대상 수를 줄인 데이터를 사용하였다. 사용된 데이터는 용현동 주변의 인천 일부 지역으로서, 폴리곤의 수는 15682개, 포인트 수는 114028개이다. 라인의 경우는 21670개, 포인트 수는 87892개이다.

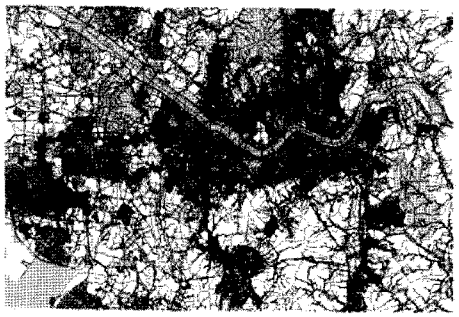
위와 같이 두 개의 다른 데이터에 대해 기존의 압축 방법과 본 연구에서 제안하는 압축 방법을 각각 적용해 보았다. 첫 번째 실험 데이터는 밀집해져 있는 공간적 구성과 희소적인 공간적 구성이 잘 반영이 되어 있는 구조이다. 따라서 다양한 포인트의 수와 레코드의 수에 따른 성능을 비교하기 위해 선정하였다. 두 번째 데이터는 폴리곤, 라인과 같은 데이터의 유형에 따른 압축 방법에 대한 성능을 비교하기 위해 선 정하였다. 표 9는 실험에 사용된 데이터의 특징이다.

4.2 제안된 압축 방법 실험

연구에서 제시한 압축 방법은 바이트패킹으로 압축을 한 뒤 Lempel-Ziv 압축 방법을 적용하여 다시 압축을 한다. 하지만, 복원 속도를 고려하여 Lempel-Ziv 압축 방법에서 사용되는 패드 캐릭터(Pad Character)를 사용하지 않고 키-주소(Key-Address)방식의 해쉬 테이블(Hash Table)을 사용하는 새로운 방법을 적용하여 보았다. 이런 방법은 O(N)의 성능을 가지는 Lempel-Ziv 압축 방법의 복원 속도가 데

표 9. 실험 지역 공간 데이터

	인천광역시, 수도권 데이터	인천광역시 일부 지역
데이터 수	637,256	37,352
전체 포인트수	4,099,093	201,920
데이터 분류 대상 지역	배경, 도로 인천광역시·수도권	폴리곤, 라인 인천광역시 일부



이터의 크기와 관계없이 O(1)의 성능을 가질 수 있기 때문이다.

표 10은 특정 공간 객체를 제안한 방법으로 저장한 데이터이다. 이 구조는 패턴 정보가 시작되는 정보의 위치를 헤더 쪽에 저장하게 된다. 앞쪽에 위치한 Hexa Code인 "0b"는 10진수 값으로 변환하여 레코드 부분의 11번째 위치를 가리키게 된다. 그리고 2바트 단위로 데이터를 읽어 패턴이 시작되는 위치인 "00 07"를 읽게 된다. 이 위치로부터 '08'만큼 데이터를 복원 시켜 원본 데이터로 복원하게 된다. Hash Key를 저장하는 헤더 부분으로 인해 Lempel-Ziv 압축 방법보다는 항상 1byte 내지 2byte의 크기를 갖는 것이 단점이지만, O(1)의 성능으로 저장된 정보를 찾는 것이 가장 큰 장점이다. 연구에서 제안하는 방법에 대한 실험은 바이트패킹 압축 방법과 Lempel-Ziv 압축방법으로 설계된 데이터 모델에 대해 질의 처리 성능을 비교 분석하는 방법으로 실험 하였다.

표 10. 제안된 압축 방법으로 저장한 객체

Hexa Code
0b | 04 5b 6d 08 2c d5 01 8d 01 bf 00 07 08

4.3 결과 분석

우선 서로 다른 방법으로 인천광역시·수도권 지역의 실험 데이터를 구축하였을 때 압축된 데이터의 크기를 살펴보았다. 412MB의 크기를 갖는 원시 데이터(*.shp)를 각각의 방법으로 저장하였을 때 저장된 데이터의 크기는 표 11, 표 12와 같다. 저장된 방법은 파일 포맷과 데이터베이스의 방식을 기준으로 비교한 것이다. 제안된 압축 방법의 압축률은 기존의 방법과 비교 하였을 때 크게 향상 되지는 않았다. 다음은 질의에 관한 실험을 하였다. 인천광역시와 수도권 일대의 지역에 대하여 공간 질의를 하였을 때 이를 복원하여 원본 데이터로 회복이 되기까지 걸리는 시간을 포인트 수와 데이터의 수에 따라 분석하였다.

표 11. 압축률 비교 - 원시 데이터 기준

저장 방법	압축된 크기 (MB)	위치정확도 (100%)	압축률 (%)
바이트패킹	103,020	100	74.3
Lempel-Ziv	98,983	100	75.62
본 연구 방법	98,328	100	75.62

표 12. 압축률 비교 - WKB(OGC) 기준

저장 방법	압축된 크기 (MB)	위치정확도 (100%)	압축률 (%)
바이트패킹	103,020	100	53
Lempel-Ziv	98,983	100	57
본 연구 방법	98,328	100	57

그림 3은 질의가 발생할 시 검색 대상이 되는 데이터의 수에 따라 복원 시간 까지 걸리는 시간을 나타낸다. Lempel-Ziv 압축 방법이 복원까지 걸리는 질의 처리 시간이 가장 오래 걸리는 것으로 확인할 수 있다. 또한, 압축을 하지 않고 저장하는 방

법과 바이트패킹 압축 방법이 거의 비슷한 복원 시간이 소요되는 것을 알 수 있었다. 하지만, 제안된 압축 방법은 기존의 저장 방법인 바이트패킹 방법보다는 평균 31% 정도 복원 속도가 빠름을 확인할 수 있다.

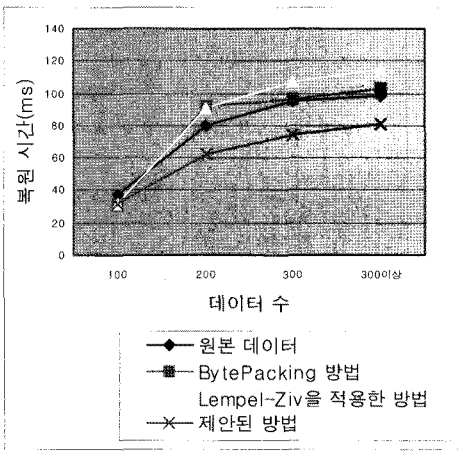


그림 3. 데이터 수에 따른 복원 속도

그림 4는 포인트 수에 따른 복원 속도에 대한 관계를 나타낸 것으로써, 제안된 방법이 다른 방법보다 25%의 정도로 복원에 관한 속도가 향상 된 것을 나타낸다.

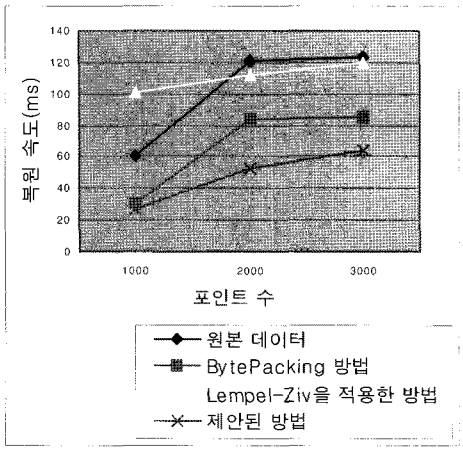


그림 4. 포인트 수에 따른 복원 속도

다음으로 압축 방법에 있어서의 복원 시간과 관련된 질의 처리 성능을 비교해 보면 그림 5의 결과를 얻을 수 있었다. 우선적으로 Lempel-Ziv 압축 방법으로 중복에 대한 문제를 보완할 경우는 압축률이 높을수록 복원까지 걸리는 시간이 큰 폭으로 증가했다. 하지만 본 연구에서 제안된 키-주소 방식으로 보완한 방법은 압축률에 따른 시간이 Lempel-Ziv 압축 방법 보다는 안정적으로 증가한 것을 볼 수 있다. 일반적으로 질의를 실행하는 단계에 있어서 제안된 방법은 바이트패킹 압축 방법과 비교 했을 때 평균적으로 8ms 만큼 질의 처리에 대한 성능이 향상되었다. 일반적으로 제안된 압축 방법은 기존 압축 방법 보다 질의를 처리하는데 있어서 효과적으로 처리하는 것을 볼 수 있었다. 하지만, 중복 저장에 대한 문제점의 보완을 하였던 제안된 압축 방법으로서는 압축률의 향상을 얻을 수 없었다.

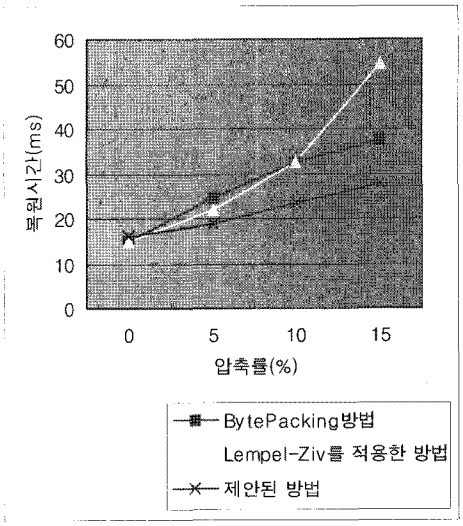


그림 5. 압축률과 복원 속도 관계

5. 결론 및 고찰

본 연구에서는 혼합 방식을 적용한 무손실 벡터 압축 기법을 설계하고 결과를 선행 연구와 비교 분석을 통하여 질의 처리에 대한 검증을 하였다. 연구의 궁극적인 목적은 무손실 벡터 데이터 압축 기법으로 질의 처리의 향상을 얻는 것이었다. 따라서 차분 벡터들을 바이트패킹 압축 방법으로 압축을 하고 중복이 되서 저장하는 정보를 찾아 다시 재압축을 하는 혼합 방식을 사용하였다.

무손실 압축 방법은 데이터를 압축하여 복원 시 원본 데이터로 완전히 복원하는 방법이다. 기술적인 면으로 볼 때, 저장하려고 하는 정보를 대상으로 반복이 되는 패턴 정보를 찾아내어 압축하는 알고리

즘을 기반으로 한다. 하지만 무손실 압축 방법은 압축률이 일정한 수준이 되면 더 이상 압축하기가 어렵다. 압축이 될 때 복원을 위한 정보도 같이 저장이 되기 때문이다. 즉 일정수준의 압축률에 도달하면, 복원을 위한 정보도 그만큼 증가되어 저장이 된다. 연구에서는 바이트패킹 압축 방법에 대해 발생되는 중복 정보의 문제를 제안된 압축 방법으로 다시 재압축을 하는 방법을 제안했다. 이 때 압축되는 정보의 위치를 헤더 정보로서 관리를 하게 하였다. 이는 중복 저장에 대한 문제와 복원을 위해 소요되는 비용 문제를 최소화하는 장점을 얻을 수 있었지만, 헤더 정보가 저장되는 추가적인 저장 공간으로 인하여 압축률에 대한 성능을 얻을 수 없었다.

본 연구에서는 기존에 제시된 바이트패킹 압축 방법과 비교 실험을 하여본 결과, 압축률의 향상은 얻을 수 없었다. 하지만 전반적으로 질의 처리에 의해 복원이 되기까지의 걸리는 시간이 기존 압축 방법 보다 향상되는 것으로 나타났다. 데이터 수와 포인트 수가 많은 지역에 대한 질의 실험은 혼합 방식이 질의 처리에 있어서 가장 빠르게 처리하는 것으로 나왔다. 또한, 대상 수가 작은 실험 데이터에 대한 질의 실험도 혼합 방식이 기존 압축 방법보다 빠르게 처리하는 것을 볼 수 있었다. 특히 원본 데이터를 압축을 하지 않는 데이터에 대해 실험을 비교하여 볼 때 압축 방법을 적용한 데이터가 질의 처리에 있어서 더 향상 된 것을 볼 수 있었다. 이는 압축 방법이 단순히 저장 공간의 측면에서의 중요성뿐만 아니라 질의 처리에 있어서도 중요한 요소로 영향을 미치는 것을 확인 할 수 있었다.

질의 처리 속도에 있어서 기존의 압축 방법보다 제안된 방법이 나은 이유는 다음과 같이 설명될 수 있다. 우선적으로 관계형 데이터베이스 내에서 처리되는 질의 실행(Query Execution) 과정을 최소화하였다는 점이다. 제안하는 압축 방법은 질의가 실행되는 데이터에 대한 중복 정보를 제거함으로서, 객체 별로 압축의 효과를 얻을 수 있었다. 이런 점으로 인해 기존의 바이트패킹 압축 방법 보다는 포인트·데이터 수가 많을수록 데이터의 크기를 줄일 수 있었으며, 결국 질의 실행에 대한 물리적 비용을 최소화하여 질의 처리를 빠르게 처리 할 수 있었다. 혼합 방식을 이용한 벡터 데이터 압축 방법은 기존 압축 방법 보다 질의를 처리하는데 있어서 보다 높은 효율성을 보였다. 또한 무손실 압축 방법을 바탕으로 하기 때문에 위치에 대한 정확도가 정밀히 유지된다. 따라서 정밀한 공간 연산이 필요한 응용 프로그램에서 실시간적인 데이터의 업데이트가 요구되는 경우 효율적으로 사용 가능한 압축 방법일 것이다.

참 고 문 헌

[1] 김미란, 최진오, 2002, "모바일 벡터지도 서비스를 위한 클라이언트/서버 시스템의 설계및 구현", 정보처리학회논문지 9-D 제5호.

[2] 이민우, 2005, "임베디드 시스템의 객체 관계형 DBMS에 적합한 공간 인덱스 방법", 인하대학교 대학원 석사학위논문:26-29.

[3] 이동헌, 전우제, 박수홍, 2005, "K평균 군집화를 이용한 벡터 데이터 압축 방법," 한국공간정보시스템학회지, 제7권, 제3호, pp. 45-53.

[4] 이석호, 2003, "데이터베이스 시스템", 정익사.

[5] 이재규, 1996, "C로 배우는 알고리즘", 세화 pp 1225-1296.

[6] 전우제, 주용진, 문경기, 이용익, 박수홍, 2005, "A GIS Vector Data Compression Method Considering Dynamic Updates," 한국GIS학회지 제13권, 제4호, pp. 355-364.

[7] 주용진, 박수홍, 2010, "동적 경로안내시스템에서 벡터 지오데이터의 관리를 위한 다중 해상도 모델" 한국지형공간정보학회지 제18권 제4호 2010년 12월 ,pp. 101-107.

[8] 함창학, 주용진, 2010, "차량 항법용 원도로 활용하기위한 교통 주제도 데이터 모델 전환에 관한 연구," 한국지형공간정보학회지 제18권 제3호, pp. 67-74.

[9] Greg Cunningham, James Gough, Gillian Silvertand, 2004, "ArcSDE : Administration for Oracle(Final)". ESRI.

[10] Hans-Peter Kriegel, Martin Pfeifle, Marco Potke, Thomas Seidl, 2003, "The Paradigm of Relational Indexing:A Survey", Database Systems for Business, Technology and Web,.

[11] Hector Garcia-Molina, Jeffery D. Ullman, Jennifer Widom, 2000, "Database System Implementation".

[12] Shashi Shekhar, Yan Huang, Judy Djugash,

Changqing Zhou, 2002, "Vector Map Compression: A Clustering Approach", Proceedings of the tenth ACM international symposium on Advances in geographic information systems, pp. 74-80.

[13] Shashi Shekhar, Sanjay Chawla, 2000, "Spatial Databases A Tour" Prentice Hall, 3.

[14] Silberschatz, Korth, Sudarshan, 2002, 김형주역, "Database System Concepts," pp 577-578..

[15] OGIS, 1999, "Open GIS Consortium : Open GIS simple features specification for SQL (Revision 1.1)".

[16] Khalid Sayood, 2000, "Introduction to Data Compression," Morgan KaufMann Publishers.

[17] Yong-Jin Joo, Soo-Hong Park, 2006, "Design and Implementation of Map Databases for Telematics and Car Navigation Systems using an Embedded DBMS," The Journal of GIS Association of Korea, Vol 14. No.4.

논문접수 : 2011.01.17
수 정 일 : 2011.02.21
심사완료 : 2011.02.24

문 경 기

2004년 수원대학교 도시공학과 졸업 (공학사)
2006년 인하대학교 지리정보공학과 졸업 (공학석사)
2006년 국립공주대학교 군사과학연구소 연구원
2008년 파인원커뮤니케이션즈 전임연구원
2009년~현재 엠앤소프트 주임연구원
관심분야는 LBS, 공간DB, 공간데이터압축, 공간인덱싱

주 용 진

2001년 인하대학교 지리정보공학과 공학 졸업(공학사)
2003년 인하대학교 지리정보공학과 공학 졸업(공학석사)
2004년 한국교통연구원 국가교통DB 센터 연구원
2009년 인하대학교 대학원 졸업(공학박사)
2009년~현재 서울시립대학교 도시과학연구원 융합도시연구센터 연구교수
관심분야는 위치기반서비스, 공간DB, 공간추론 및 온톨로지, 도시-교통 통합모형

박 수 홍

1989년 서울대학교 지리학과 졸업(학사)
1991년 서울대학교 지리학과 졸업(석사)
1996년 Univ. of South Carolina at Columbia 졸업(지리학박사)
1998년~2000년 서울시정개발연구원/연구위원
2000년~현재 인하대학교 지리정보공학과 교수
관심분야는 u-GIS Service Model, Spatial Database, Spatial Data Models