The Journal of GIS Association of Korea, Vol. 14, No. 4, pp. 379-389, December 2006

Design and Implementation of Map Databases for Telematics and Car Navigation Systems using an Embedded DBMS

Yong-jin Joo* · Jung-yeop Kim** · Yong-ik Lee*** · Kyung-ky Moon**** · Soo-hong Park*****

ABSTRACT

Map databases for CNS (Car Navigation System) can be accessed quickly and compressed efficiently due to that these are usually recorded as in a PSF (Physical Storage Format). However, it is difficult to create and manage data storage based on a file-system. To solve these problems, DBMS needs to be combined with spatial data management. Therefore, we developed an embedded DBMS with which to store data and conduct quick searches in CNS. Spatial data could be easily managed and accessed using the compression method, Multi-Link, spatial index, and spatial division. In the result, the proposed embedded DBMS searched quickly and stably supported data management. If synchronization is applied in DBMS, it is expected to utilize the advantages of an embedded DBMS.

Keywords : embedded spatial DBMS, car navigation system, multi-link, spatial index

요 약

현재 차량 항법 서비스를 위한 데이터는 물리적인 저장형식(PSF : Physical Storage Format)에 따라 설계되어 대용량의 GIS 원시 데이터를 작은 용량으로 압축하고 빠른 매체 접근이 가능하도록 변환된 서비스용 지도 포맷을 제공한다. 하지만, 파일 시스템 기반의 복잡한 구조는 데이터 생성과 관리를 어렵게 하고 시스템 간 상호 호환성이 결여되는 문제점을 가진다. 이러한 문제를 해결하기 위해 최근 대두되고 있는 차량용 항법 시스템에서는 견고한 데이터 관리, 데이터의 동기화, 그리고 실시간 데이터 처리

* Ph. D. Candidate, Department of Geoinformatic Engineering, Inha Univ. (dccmdrum@daum.net)
** Ph. D. Candidate, Department of Geoinformatic Engineering, Inha Univ. (jyfloo@empal.com)
*** Master Course, Department of Geoinformatic Engineering, Inha Univ. (a78leekey@naver.com)
**** Researcher, Research Institute for Military Science (starmkk@naver.com)
***** Professor, Department of Geoinformatic Engineering, Inha Univ. (shpark@inha.ac.kr)

의 필요성에 때문에 데이터베이스 시스템으로 공간 데이터 관리의 기능을 접목하려고 한다. 따라서, 본 연구에서는 차량 항법 서비스를 위한 데이터의 저장과 빠른 검색을 지원하는 차량용 Embedded DBMS 모듈을 개발하였다. 이를 위해 압축 기법, Multi-Link, 공간 인덱스, 공간분할을 적용하여 대용량의 공간 데이터를 효율적인 관리와 접근이 용이하도록 하였다. 또한, 어플리케이션에서의 데이터 검색과 표현에 필요한 API 개발을 하였다. 결과적으로, 개발된 Embedded DBMS는 적은 용량과 빠른 검색 구조인 PSF의 장점을 그대로 유지하면서 안정적인 데이터의 관리에 적합한 구조를 지원한다. 향후 DBMS모듈에 동기화 기법을 적용한다면 데이터의 현시성이 중요한 차량항법용 데이터의 유지관리 측면에서 강력한 데이터 관리가 용이한 Embedded DBMS의 장점을 더욱 활용할 수 있을 것으로 기대된다.

주요어 : 임베디드 공간 데이터베이스, 네비게이션 시스템, 멀티-링크, 공간 인덱스

1. Introduction

The practical uses of location information as well as user requirements are increased in a mobile environment that supports portability. Especially, a telematics service for the movement information usage of a location information foundation is utilized with one of the u-IT839 strategy. The GIS DB technique is an important element of a telematics service. It involves a location-based service for car navigation, spatial data schema, a spatial index and management through the spatial reference of spatial data.

So far, CNS (Car Navigation System) data is stored in formats such as HDD, CD-Rom, DVD-ROM, and CF/SD memory card. Data designed in PSF (Physical Storage Format) are service maps that can be accessed quickly and compressed efficiently. PSF, however, has a complex file format, and is difficult to manage due to offset values. Therefore, it is difficult to transform original map data into service map data, and neither is it easy to update data. Also, applications not based on PSF present the problem that there is no compatibility.

To solve these problems, we need an efficient method of utilizing DBMS to store large size spatial data. Also, we need a management function for robust data management of spatial data in CNS, data synchronization, and real-time processing. Therefore, we constructed a map database for CNS using an Embedded DBMS to supplement the structure of the PSF foundation system and compensate for it many faults. The database maintained the advantages of stable data management and added efficient query processing of large data through spatial index, space division, Multi-Link and compression. Additionally, we simplified the complex file structure and constructed a flexible schema and Multi-Link structure, and we supported an efficient and consistent developmental environment and guaranteed system interoperability through standard APIs such as map displays which support multi-scale, POI search, compression and spatial index. Consequently, these applications with DBMS provide not only the merits of the PSF format but also those of proper format in data management.

We introduce the necessity of the map database technique for CNS service using DBMS in Chapter 1. In Chapter 2, we describe the DBMS requirement of a Map Database and the design considerations necessary to satisfy this. In Chapter 3, we explain the developmental environment for system implementation, spatial module and functions to be implemented such as data loader, compression, Multi-Link, spatial index and the clustering method. In Chapter 4, we perform a test of the efficiency of spatial division and Multi-Link in order to improve query processing. Finally, Chapter 5 offers conclusions and suggests possible future work.

2. Requirements and Design Considerations

An embedded database is database software that is used for data management in embedded systems. Although a general database is focused on complex query processes, an embedded database performs only queries, and is designed to a small size. An embedded database, also, should be compatible with various operating systems[1, 5].

To use spatial data in CNS, we combined a database system with the management function of spatial data. It costs less to develop an embedded DBMS because there is no need to provide every function of a commercial DBMS[4]. However, the embedded DBMS for CNS should provide functions that can manage spatial data.

A GIS Database for CNS service stores multiple objects such as road, background, annotation and POI. Spatial data occupies much storage space and has a complex storage structure between levels. So, the map database requires a method providing processing speed for large spatial data with efficient query processing. And the database schema must be simple and support a flexible structure. Consequently, Spatial DBMS for CNS requires modular development, unlike general spatial databases.

We considered the relevant particulars in applying the embedded database to the CNS. First of all, the embedded database should involve a compression method that can manage enormous spatial data, and should reduce data using a spatial index. Second, the embedded database should perform within a proper query time using a spatial clustering method due to the load that query processing gives. Finally, the embedded database should optimize queries to execute services in CNS.

3. Implementation of a Prototype Embedded Spatial DBMS

This chapter introduces the developmental environment for system implementation and explains methods that enhance data simplification and query processing.

3.1 Design of the Prototype System for CNS Service

We designed a spatial database for CNS using embedded DBMS (SQLite). SQLite is an embeddable SQL-driven database engine that implements both the database engine and its interface as a C/C++ library. SQLite is extremely efficient, benefiting from a highly optimized internal architecture and a small memory footprint. SQLite supports a large subset of the ANSI SQL-92 standard, and can manage a large spatial database because it can store 2TB data.

The SQLite library includes a very powerful mechanism for adding user-defined functions to the SQL command set. Custom functions can even be written in many of the supported language APIs, not just C/C++.

Because SQLite has these advantages, it provides good performance in a mobile device for CNS. We ported SQLite-wince-3.3.5 on WinCE to design a spatial database, and developed APIs that can show map data and change levels using Pocket PC 2003 SDK. Figure 1 shows the structure of the embedded spatial database.

The original map data is divided by levels and is converted. Also, spatial data are loaded in the database using compression Data loader applies input data to the schema and performs the conversion function. This eases data conversion as well as the process of constructing a database. Additionally, the compression method is applied to decrease volume of the spatial data. We developed Multi-Link to support an efficient network-data hierarchy structure, because network data needs a stage presentation of various levels and a hierarchy entity relation.

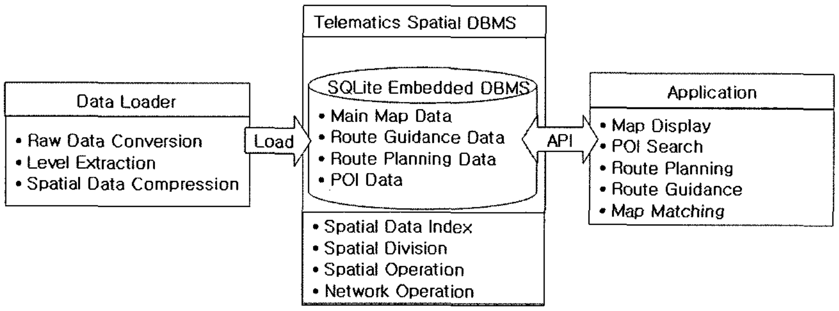
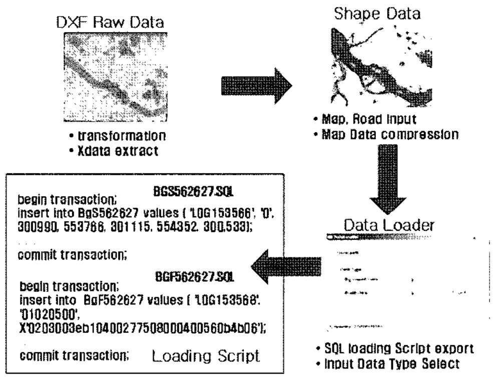


Figure 1. The structure of the prototype system

Using that structure, we utilized various database management functions. Expanded SQLite DBMS can perform simple spatial or non-spatial queries, and can add or delete data. Expanded SQLite DBMS improved search performance through the spatial data index and spatial division, and guaranteed the independence of the DB. Also, it can communicate with other CNS applications through APIs.

3.2 Data Loader

The spatial data model should store the spatial database in the embedded database. In particular, the model should be designed with the most suitable form for efficient storage. And spatial data should be transformed to fit the internal table schema of the RDBMS when queried. Data loader is a tool that can satisfy these requirements, and play a part in storing the original map data into the embedded spatial database. As shown in Figure 2, raw data such as that of the network, background, annotation, and POI is extracted by level and is converted to shape files. The loader applies input data to the schema specification to be defined and fills in the script for database storage. Each table consists of an S table, which is a spatial index, and an F table, which is stored data.



```
BGS562627.SQL
begin transaction;
insert into BgS562627 values ( 'L0G153566', '0',
300990, 553786, 301115, 554352, 300,533);
. . .
commit transaction;

BGF562627.SQL
begin transaction;
insert into BgF562627 values ( 'L0G153568',
'01020500',
X'0203003eb1040027750800040056004b06');

commit transaction;
```

Figure 2. Data loading procedure

3.3 Data Compression

Large-sized spatial data occupies much storage space in compact mobile devices for which resources have been restricted. We require the compression of spatial data to enhance the utility of query processing for storage space. As shown in Table 1, we customized the WKB of the OGC foundation to the model in order to store spatial data, and used a bytepacking-compression method to reduce the storage space of coordinate information. The absolute starting point of the object that expresses the location and differential vectors is saved.

Given a line string or polygon in an uncompressed map form, we first convert the coordinates in the uncompressed map into base points followed by a sequence of differential vectors. The differential vectors

Table 1. Structure of Vector Data

Data Field	Data Type	Byte
Spatial Data Type	Unsigned Char	1
Sum of Point	Unsigned Long	4
X (first point)	Long	4
Y (first point)	Long	4
length of feature	Unsigned Short	2
differential vector	Bit	variable

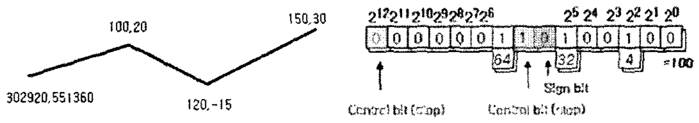


Figure 3. Embedded Spatial Database-based compression

may be determined by the vector differences between the current point and the previous point or the first point. The series of differential vectors produced can then be encoded to produce the compressed data by Bytepacking-Compression[7]. As a result, the compressibility by this method was improved by 40 percent.

3.4 Multi-Link

Multi-Link is an important technique that enhances the query performance for a hierarchical road network. Multi-Link reconstructs the lower-level links through common attributes corresponding to the upper-level links. It supports an efficient hierarchy for retrieval of the network and in presentation. Generally, the upper-level road data are constructed hierarchically according to the criterion of the lower-level network data. Because the upper-level data consist of lower-level nodes and links, the upper-level data share the complex lower-level data structure. This results in a large database and the accompanying difficulty of management[2, 6]. In the result, this structure diminishes the efficiency of the search in DBMS. To solve these problems, we designed the Multi-Link structure after integrating the road data of the same attributes[3].

We assigned the lowest-level link with incremental number to constitute Multi-Link using dynamic segmentation. The unique ID number can be specified for a road link at the lowest level. They are numbered in sequence within a link and sorted in ascending order. Link IDs are specified according to links for lower level road displays. An integrated link of roads at a higher level is represented using link id at its origin link and link id at its end link. The lowest-level link can be used in road data, path-computation data, and road-guidance data. In a preprocessing step, we merge all layers of the unit parcel to extract the common attribute of the links. Afterwards, we provide topological information on the node at the link. To yield link numbers in order, we produce the set at the upper-most level with the node and link according to the rank and road number. A

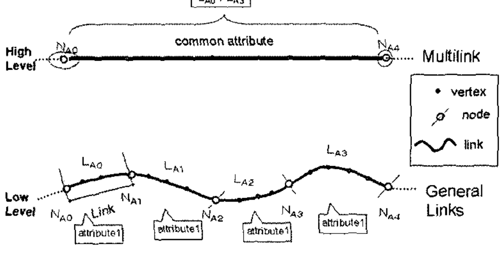


Figure 4. The structure of the General-Link and Multi-Link

common attribute of Multi-Link comes to make with origin link ID and end link ID. Also, the vertex of a Multi-Link is made of the coordinate of nodes.

3.5 Spatial Index

There are R-tree and Grid indexing methods for efficient searching. The R-tree method consists of hierarchical MBR (Minimum Boundary Rectangle), and is efficient in updating an index. The Grid method stores the target area divided into grid cells. The Grid method is efficient in area querying and performs with fewer disks I/O. Implementation is easy with a simple data structure. Also parcel, the unit of data management in navigation, is similar to grid. A grid index of various sizes can enhance query performance for a hierarchical parcel structure. Therefore, the present study used a spatial index for spatial mobile devices using a global fixed-grid index, efficient for simple displays and calculations of the spatial data of points and ranges. The following formula (1) is the hashing function used for the global fixed-grid index.

$$\begin{aligned}\text{Grid ID}(x) &= \text{int}(x\ \text{Minx}) \ / \ \text{SizeX} \\ \text{Grid ID}(y) &= \text{int}(y\ \text{Miny}) \ / \ \text{SizeY}\end{aligned} \qquad (1)$$

3.6 Spatial Division using a Quad-Tree Method

We used a quad-tree method to enhance database query speed excepting the spatial index. The spatial index returns all entities for the retrieval area. Retrieval speed will be remarkably slow if many metropolitan area entities are retrieved. Therefore, we need a clustering method that considers the number of entities.

In the case that a table is constructed with each parcel, the amount of data influences the query. Therefore, the spatial data needs to be divided in order that queries can be performed efficiently. We conducted spatial division only when the number of the objects exceeded the critical value. Theses division is conducted until the number of the objects does not exceed the critical value.

When the number of objects exceeded the critical value, we divided the area into four equal areas. The object is stored in the division line in duplicate. If the critical value is too low owing to the fact that there is much duplication, it is important to select a proper critical value. In the present study, we determined the critical value (a=5,000) according to the number of duplicate objects and retrieval time.

4. Performance Test

In this chapter, we test the efficiency of spatial division and Multi-Link in order to improve query processing. The performance assessment compared query processing time for spatial division. We also accomplished a correlation analysis of the factors influencing query processing time. Finally, the performance assessment of Multi-Link for query performance and stability was executed.

4.1 Spatial Division Efficiency Estimation

Performance is irregular according to the number of parcel entities when we store all of the entities in one table. To solve these problems, we executed the spatial division by the Quad-Tree technique. As shown in Table 2, data size increased by 13% before the spatial division was applied, and the number of total individuals was confirmed to be increased by 5%.

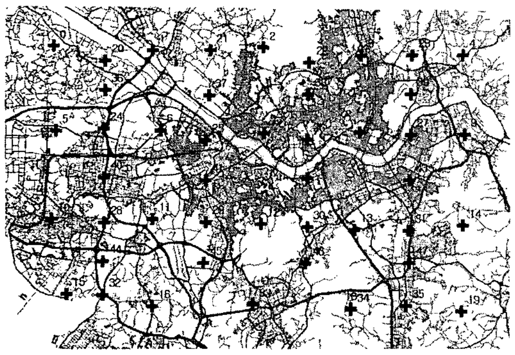


Figure 5. Query Location in Study Area (+ : 49 Point)

We constructed statistical data to analyze the query processing time for 49 points within a research area through spatial division. We produced statistical data for the query time of background data (BgTime), the number of entities to be displayed (BgSum), and the number of the vertex of the entity (BgNumPoint). We queried 10 times for each point. We calculated the arithmetic average and computed the central tendency from the result.

As shown in Figure 6, the query processing time was reduced by applying spatial division.

However, because Figure 6 cannot explain that spatial division is more efficient in query performance, we also conducted an advanced analysis. That is, we searched, through a

Table 2. Comparison of Spatial Division before with after (Critical value = 5,000)

Spatial Division	Data Size (byte)	Rate of Increase (%)	Total Objects	Rate of Increase (%)
After	81702.912	113	359826	105
Before	93921.28	100	339576	100

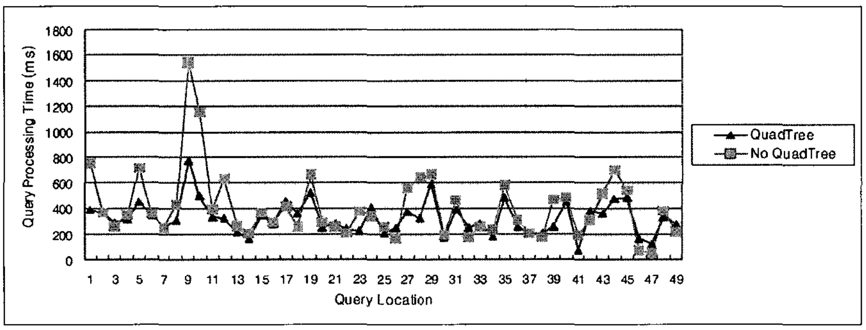


Figure 6. Query Processing Time for each Query Location with Quad-Tree and Not Quad-Tree

correlation analysis, the item most related to query time (Table 3).

We can see that query time (BgTime) is concerned with the number of objects that is displayed in the table (BgSum). We conducted regression both for spatial division and NOT spatial division.

BgTime = 4.6376×Bgsum+237.1064 (NOT spatial division)

BgTime = 2.5736×BgSum+229.0901 (spatial division)

The number of objects using the two regressions was the same.

Figure 7 shows that query time in spatial division is faster than query time in NOT spatial division. Therefore, retrieval time of entities was reduced by applying quad-tree and it is a technique that can increase query efficiency.

4.2 Multi-Link Efficiency Estimation

The road data were divided according to attribute similarity. Multi-Link is a technique that considers divided links as one object. It shows complex road data easily, and decreases query time. Figure 8 shows the results for the query time in level 3.

As can be seen, Multi-Link is more efficient than NOT Multi-Link. That is, it maintained

Table 3. Correlation analysis for statistical items

	BgTime	BgSum	BgNumPoint
BgTime	1		
BgSum	0.818177098	1	
BgNumPoint	0.444577704	0.503255	1

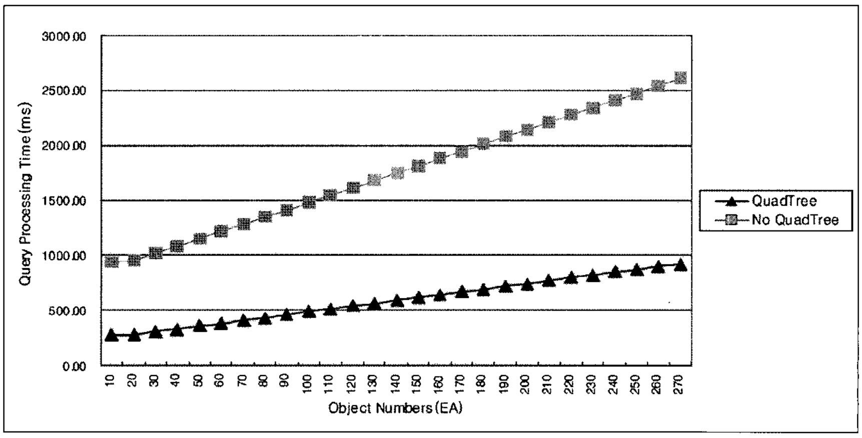


Figure 7. Compare Query Processing Time with space division before with after

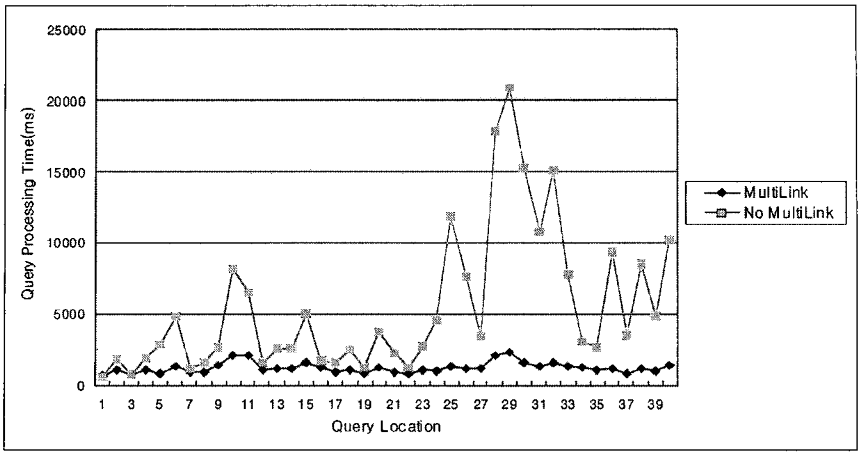


Figure 8. Query Processing Time for each Query Location with Multi-Link and NOT Multi-Link

query time stably and conducted queries quickly. Therefore, we can confirm that Multi-Link improved query performance.

5. Conclusions

In this paper, we construct a map database for CNS using an Embedded DBMS to supplement the structure of the PSF system. We developed an Embedded DBMS module that supports data storage for CNS services and an indexing technique for retrieval. We developed compression, Multi-Link, and spatial index to manage complex and large-sized spatial data. We considered the load that query processing applies to the embedded system and improved query processing time through spatial division and the Multi-Link technique. We maintain the advantages of a database offering stable management of data and added efficient query processing of large data through spatial index, space division Multi-Link, and compression. Also, we eliminated the complex file structure and substituted a simple schema and Multi-Link structure.

The independence of the DB was guaranteed through implementation of APIs for retrieval and representation of data. And, we supported an efficient and consistent developmental environment and guaranteed system interoperability through standard API. Consequently, the developed DBMS maintains the advantages of existing PSF through indexing and compression, and supports efficient management of data.

We hereafter will apply a synchronization technique requiring research on real-time updating of data. The efficient management of data can apply the advantage of easy embedded DBMS more on the maintenance side. Thus we will commence research on the updating of traffic information processing and real-time map data.

Reference

[1] A. Tesanovic, D. Nystrom, J. Hansson, and C. Jan. 2002, Nystrom, Embedded Databases for Embedded Real-Time Systems: A Component-Based Approach, tech. rep., Department of Computer Science, Linkoping University and Department of Computer Engineering, Malardalen University

[2] Bo Huang, 2002, An ODMG-based object model for dynamic segmentation, IEEE ITS International Conference

[3] Dimitis Papadias, Jun Zhang, 2003, Query Processing In Spatial Network Database

[4] Huang, B., Fwa, T. F., and Chan, W. T., 2004, Pavement distress data collection system based on mobile GIS. Transportation Research Record. 1889: 54-62.

[5] M.A. Olson, September 2000, Selecting and implementing an embedded database system, IEEE Computer Society, 33(9): 27-34,

[6] Maarten Vermeij, Peter Van OoSterom, 2000, Storing And Using Multi-Scale Topological Data Efficiently In A Client-Server DBMS Environment

[7] 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