

A System Framework for Map Air Update Navigation Service

Kyoungwook Min, Kyoungwhan An, Insung Jang, and Sungil Jin

The quality of navigation service is determined by the accuracy of the available data. For existing navigation services, a full map update is provided in order to keep the map data of mobile devices current. As content and services of mobile devices have recently been diversifying, the size of map data managed in mobile devices has increased, reaching several gigabytes in size. It generally takes tens of minutes to write several gigabytes of data into mobile device storage. For traditional navigation systems, a complicated storage structure called a physical storage format (PSF) is used to assure maximum processing performance of map data in mobile devices within limited resources. Consequently, even though modified navigation map data actually affects only a portion of a map, the full map data is updated because partial updates are not possible. In this paper, a navigation system is studied to solve this difficult partial map update problem. The map air update navigation system, which is the result of this study, provides real-time partial map updating using wireless communications.

Keywords: Mobile navigation system, navigation map update, mobile database, version control, routing service.

I. Introduction

The latest mobile devices can provide large-scale data processing as their computing power and resources become abundant. Navigation services are favorites among mobile services, and the size of map data has reached several gigabytes. Navigation data comprises visualization of geometry data, point of interest (POI) data, and road network data. Existing navigation services in which all of navigation map data is managed in mobile device provide full map updates two to three times a year via wired Internet service. However, there is a waste in cost due to a full map update even though the actual amount of updated data is only tens of kilobytes or several megabytes at most. It is not possible for existing navigation systems to offer a partial map update, that is, a set of part of map data newly added, deleted, or changed, because the relationship among navigation map data objects use a file offset value in the physical storage format (PSF) used in the existing system. In particular, for road network data consisting of nodes and links, updating is far more difficult due to the connectivity and multilevel hierarchical structure of data. The existing format is a read-only format, making it very difficult to add, change, or delete a map data object.

While there have been studies on storage structure and a potential system for partial updating of navigation map data, no success has been reported thus far. There has been a worldwide trend in using a mobile database management system (DBMS) to solve these problems because insertion, updating, and deletion of data are fundamentally provided in such systems, and independent data management of applications is easy. However, in the commercialization of mobile DBMS thus far, the search and display performances of navigation map data in mobile devices are degraded because spatial data types, query

Manuscript received Dec. 17, 2010; revised May 13, 2011; accepted May 19, 2011

This work was supported by the Industrial Strategic Technology Development Program (10035250, Development of Spatial Awareness and Autonomous Driving Technology for Automatic Valet Parking) funded by the Ministry of Knowledge Economy (MKE), Rep. of Korea.

Kyoungwook Min (phone: +82 42 860 1708, email: kwmin92@etri.re.kr), Kyoungwhan An (email: mobileguru@etri.re.kr), and Insung Jang (email: e4dol2@etri.re.kr) are with the IT Convergence Technology Research Laboratory, ETRI, Daejeon, Rep. of Korea.

Sungil Jin (email: sijin@cs.cnu.ac.kr) is with the Computer Science Department, Chungnam National University, Daejeon, Rep. of Korea.
doi:10.4218/etrij.11.1610.0012

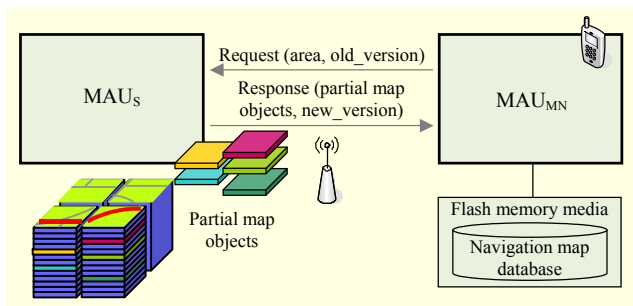


Fig. 1. System framework for map air update navigational service.

processing of spatial indexes, and spatial searches are not provided. Furthermore, repetitive query-by-query searches of nodes and links on the computing route in DBMS do not assure its performance.

In this study, a system framework for map air updating of a navigation service is developed in order to solve these problems, and navigation map data management in mobile devices, along with partial versions of server-located map data, is emphatically researched.

Figure 1 shows the system framework for the proposed map air update (MAU) navigation service. The MAU mobile navigation (MAU_{MN}) requests an update of map data in an older version to the server, and the updated map data in a new version is transmitted to the mobile navigation system via wireless telecommunications as the protocol between the mobile device and server system. The role of the MAU server (MAU_S) is managing the version of the partial map data and providing the map data to the mobile navigation device. The role of the MAU_{MN} is managing the navigation map data and providing navigation services in a mobile device.

The remainder of this paper is organized as follows. Section II provides a summary of related research. A service protocol between MAU_S and MAU_{MN} is defined in the next section. Section IV explains the major function of MAU_S , while in section V, MAU_{MN} is explained in detail, particularly mobile DBMS for navigation services. In section VI, experimentation results using large-scale map data and a performance evaluation are given. Finally, section VII offers some concluding remarks regarding this research.

II. Related Works

In this section, related works are reviewed. These related works are divided into two categories: partial updates of navigation data and mobile DBMS.

1. Research of Map Air Update for Navigation

Existing commercial navigation systems use a PSF structure

that includes the KIWI [1], GDF [2], and NAVIKEN formats [3], among others, as map data formats that are optimized for storage media of mobile devices. The reason for using a PSF is to assure the size minimization of map data, visualization performance, and route computation. However, this is a complicated structure based on a file system and has disadvantages: difficulty managing large map data and a lack of interoperability. It is also not possible to update partial map data, and therefore it consumes a lot of time and cost by updating full map data.

The ActMap project of Europe [4] developed a prototype level system by researching exchange formats, servers, and mobile devices for partial map updates, but this system was not commercialized.

To overcome the disadvantages of a PSF structure, an embedded spatial DBMS prototype was implemented in [5]. Spatial data query processing and navigation functions were added using the open source SQLite [6]. More specifically, a spatial data index operator and network operator were implemented using the SQLite engine library. Multilink and grid spatial index techniques are focused on supporting a hierarchical road network structure. However, a partial update of road network data logic becomes complicated, and the quantity of update data increases. Also, performance is degraded because a large number of tables need to be searched to compute a route. The navigation map data is managed by partition, and each table is generated by the respective partitions. In conclusion, there is a performance limit to the prototype developed in [5] because the core technique for navigation was developed not by embedding directly in DBMS but by embedding at the application level.

2. Research of Mobile Embedded DBMS

Commercial mobile DBMS has been applied and utilized in various industries. Existing research on mobile databases has issues such as data synchronization, mobile transaction processing, a small footprint, and flash memory characteristics [7]. The flash memory storage media characteristics of the mobile DBMS are described in [8]. As the storage capacity of flash memory reaches a large scale, major issues for improving the performance and lifetime of flash memory are being described. The most prominent characteristic of flash memory is the cost of read/write/erase and a difference in the processing unit. In the case of read/write, it is generally processed in sector units (sector or page = 512 B). Block units are used for deletion (block = 16 kB). The performance of an overwrite operation is very low because the erase operation is carried out first if the sector that has been already recorded is overwritten [9].

The performance of flash memory is good for sequential

write patterns, but the performance is degraded for random write patterns such as DBMS applications. Therefore, in [10]-[12], some algorithms are suggested to prevent performance deterioration in frequent random write applications of DBMS.

A spatial main-memory DBMS (MMDBMS) utilizing the main memory of a mobile device as storage media was developed in [13]. It was developed to provide the spatial data type using the extension of legacy relational main-memory-based DBMS. Spatial MMDBMS is similar with this research in terms of supporting spatial data query processing but differs in that this research uses flash memory as a storage media in order to process large-scale data.

III. Map Air Update Service Protocol

In this section, we define the service protocol using XML schema for an MAU service. This protocol defines three kinds of services. Discovery service provides summary information such as update data size and download time of data for the region the navigation user wants to update. Provision service provides data for the region the navigation user wants to update. Subscription service sets up the interested region to receive the update data through a push service rather than a pull service. If the update data exists, the server notifies the summary information of update data to the mobile device.

Figure 2 shows the MAU service protocol defined by XML. Partition, area, and so on can be used as filters for the update region. An UpdateData element can be classified as

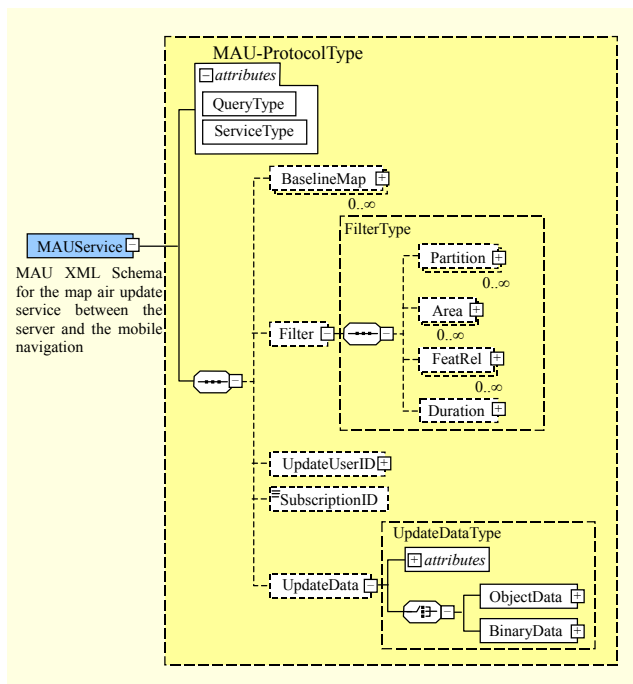


Fig. 2. XML schema for MAU service.

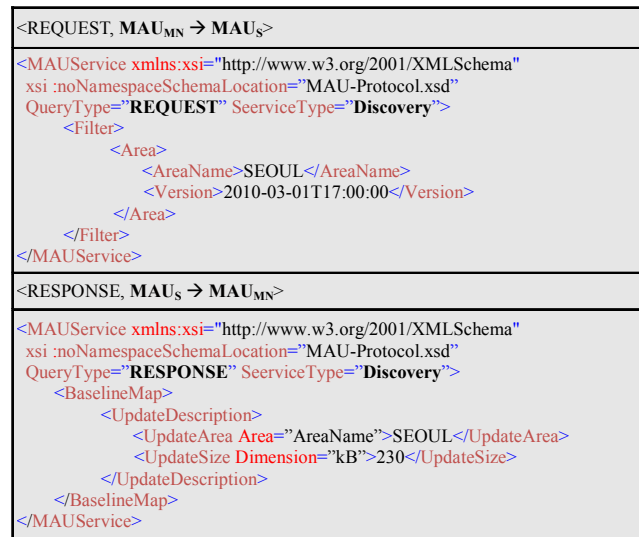


Fig. 3. Example of discovery service.

ObjectData or BinaryData. If the BinaryData element is used, the entire partition region where the update object exists will be updated in binary format. In this case, even though the number of update objects is small, the entire partition, including the object, is transmitted to the mobile device. Only the actual updated object is transmitted to the mobile device if the ObjectData element is used. The QueryType attribute is used to differentiate the service requested from the mobile device or the response from the server. The ServiceType attribute is used to specify three air update services.

Figure 3 shows an example case of a request and response for a discovery service of the MAU service. The MAU_{MN} requests the discovery service using 'SEOUL' as filter area and '2010-03-01T17:00:00' as the current version. MAU_S replies to MAU_{MN} that there is '230 kB' of update data for the corresponding 'SEOUL' region. This MAU_S protocol is a domestic standard under which MAU_S and MAU_{MN} were developed accordingly [14].

IV. Map Air Update Server

MAU_S manages the baseline map for navigation and updated partial map objects and provides this to MAU_{MN} for an MAU service. Figure 4 shows the MAU_S system architecture.

As shown in Fig. 4, MAU_S consists of two major parts: a partial map register and partial map provider. A difference map extractor extracts different objects between the new and old baseline maps. The partial map version manager then provides the version to the extracted, added, changed, and deleted objects. The partial map aggregator analyzes the past and current states of a map object in order to minimize the quantities of update data. The partial map provider provides the

three types of air update services explained in section III, and caches the update data of frequently requested regions for a fast response to the service requests from mobile devices.

1. Version Management of Partial Map Object Updates

The version control unit for partial map updates is a partition with a fixed grid cell structure. It slices the whole map region into a grid with fixed cell sizes. Map objects existing in the

same partition have the same version of the initial baseline map. A new version value is allocated to this partition if a new updated map object exists there. Figure 5 shows some schematics of the version control used in MAU_S. There are update map objects in partitions 10, 11, and 14 in the L1_Building layer, and tables of each new version are generated. For partition 10, the value of the version is v_2 , the number of newly added map objects is 20, and the number of changed map objects is 3. These updated map objects are in L1_Building_Partition10_v₂. For partition 11, object 95 is changed twice in the v_2 and v_3 versions, and object 701 is inserted in v_2 and deleted in v_3 . We are able to know the version of the updated partition and number of updated objects by easily scanning the update meta-table.

Navigation map data usually has a hierarchical level structure and is queried map data placed at the appropriate level within the layer table according to the zoom scale. Thus, navigation map data is updated by considering the hierarchical relationship among the layers. The provisioning of the updated map data requested by MAU_{MN} is executed using the update meta-table and updated version table. After extracting the partitions that are intersected by the filter area and have a higher version of the map data of MAU_{MN}, the objects in these partitions are transmitted. Figure 6 shows an example of the provisioning service. The extracted partitions are contained

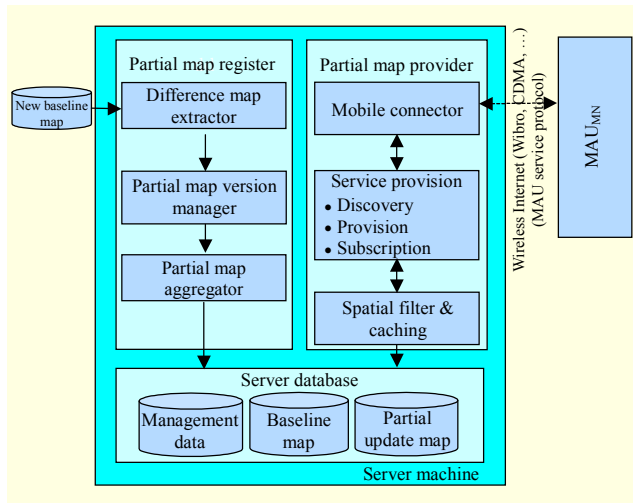


Fig. 4. Architecture of MAU_S system.

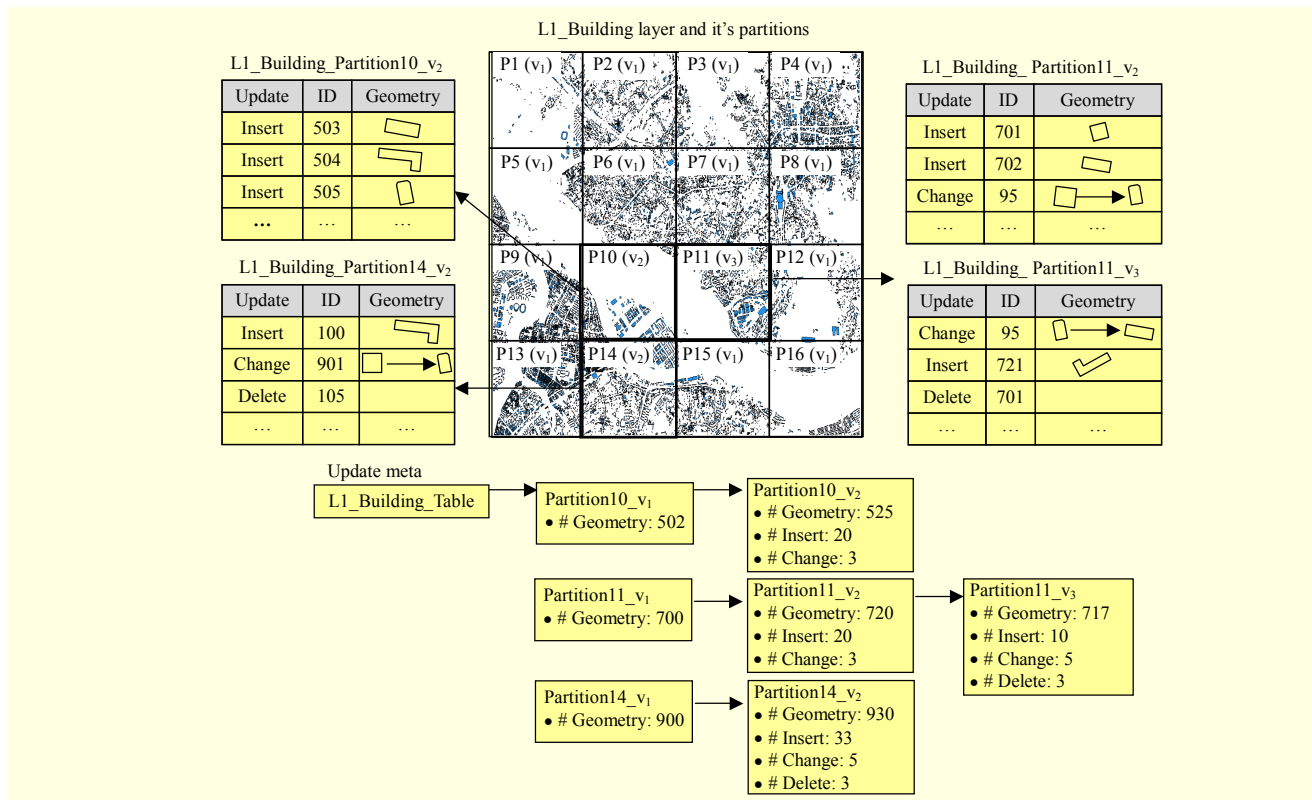


Fig. 5. Version management of update partial map data.

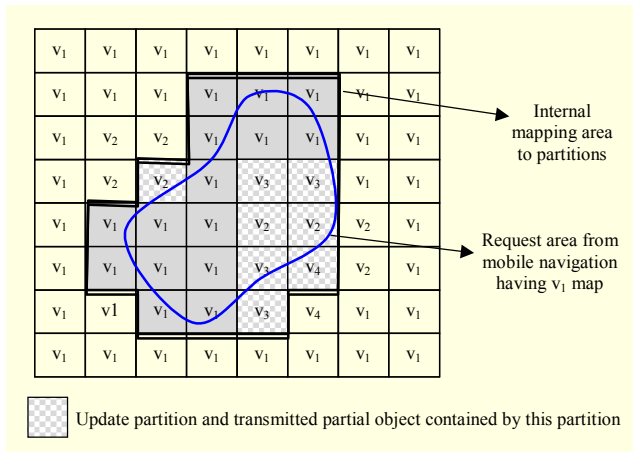


Fig. 6. Provisioning of update map data.

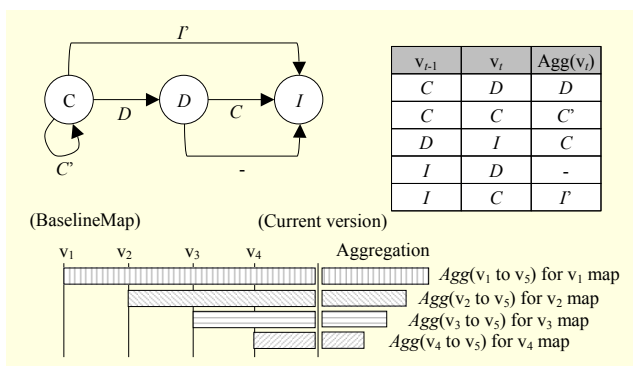


Fig. 7. Version aggregation of updated map objects.

within the requested area and have a higher version than v_1 of the map in the mobile device.

2. Aggregation of the Partial Map Data by Version

The size of partial update map data is usually within the range of tens of kilobytes or several megabytes. However, the size of the update data increases as the differences between old and new versions increase. If a navigation user has not updated their map data for a long time, more data will need to be updated to keep their mobile device current with the up-to-date version. If a mobile navigation map version is initial map version v_1 , and the most recent version is v_n , versions v_2, v_3, \dots, v_n will be updated sequentially. However, the number of updates can be reduced using an aggregation method. For example, object 701 in mesh 11 of Fig. 5 was a newly added object in v_2 . However, this object was deleted in v_3 , and therefore, the mobile navigation system using version v_1 does not need this object when updating to v_3 .

Figure 7 shows a schematic of the version aggregation method. The update transaction of a map object can be insert (I), delete (D), or change (C) within a specific version. For

updated object O_b , when the transaction of O_i is C for v_{t-1} , and the transaction of O_i is D for v_t , the resulting transaction of the aggregation for v_t is D . In this way, $Agg(C_{t-1}, C_t)$ is C'_t , $Agg(D_{t-1}, I_t)$ is C , $Agg(I_{t-1}, D_t)$ does not exist, and $Agg(I_{t-1}, C_t)$ is I'_t . Using this aggregation method, the size of the data to be transmitted to the mobile device can be reduced and the costs of update transaction processing in the mobile device can also be lowered.

V. Map Air Update Mobile Navigation

Existing navigation systems define their own proprietary formats for storing large-scale data, which is stored in several files. An application that knows its storage structure well can only access data using a file offset. Since this is not in a general format, there are many difficulties in data management. Furthermore, partial updates are difficult for data with complicated relationships, such as road network data composed of nodes and links. To solve this problem, the method proposed in this research effectively uses mobile DBMS in managing and accessing large-scale data regardless of the storage format. However, there are several problems in using commercial DBMS in mobile navigation. First, the route search performance cannot be guaranteed because large-scale node and link data have to be traversed in order to compute an optimal path, and it takes a lot of time to carry out repetitive query using the recorded logical ID. Second, it is inefficient in searching through large-scale navigation map data because spatial data types and spatial indexing are not provided. Thus, in this research, navigation-specific mobile spatial DBMS was developed to solve these problems.

Figure 8 shows the MAU_{MN} system architecture. The core engine consists of mobile spatial DBMS, a navigation

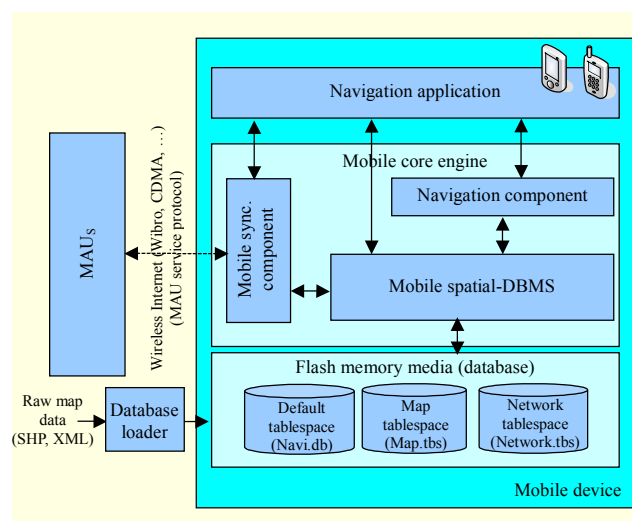


Fig. 8. Architecture of MAU_{MN} system.

component, and a mobile sync component. Navigation, mobile GIS, and other applications can be easily implemented using the mobile core engine. Mobile spatial DBMS is described in detail in subsection 1, and navigation and air update functions are described in subsection 2.

1. Mobile Spatial Database Management System.

A. Features of Mobile Spatial DBMS

The mobile spatial DBMS developed in this research has the characteristics shown in Table 1. Navigation and general relational DBMS functions are provided.

Search. The search function provides the initial sound search for Korean, in addition to searches provided through general relational DBMS. The windows, K -nearest neighbor search for spatial searches, and eight spatial relation searches (cross, disjoint, equals, overlaps, touches, within, intersects, contain) using a 9-intersection model [15] are provided. Connectivity searches using a physical record ID for road network data is also provided.

Object. The object function provides a database table composed of general and network tables. The network table takes a different structure from a general table for fast connectivity access of nodes and links when searching for a route, as it has a logical relationship as well as a physical record ID relationship for direct access. The multilevel-GRID index [16] and R*-tree index [17] are provided for fast spatial search

indexes. The multilevel-GRID index is faster in terms of spatial data updates, but the R*-tree index is faster in terms of spatial searches.

Data type. The data type function covers point, linestring, and polygon types as geometrical data types for representation of spatial data, and node and link data types as topological data types for network data.

Flash-awareness. Flash memory has different characteristics from disk memory, which is a general storage media. The most eminent characteristic is that the cost of reading and writing is asymmetric. The writing speed is very low compared to that of the reading speed. Thus, in mobile spatial DBMS, block-level clustering is provided to enhance the update performance.

B. Physical Record ID Search for High-Speed Route Searching

In DBMS, an index is usually generated at the data field table and requires fast access. Considerably fast data access is possible when using the index compared to a sequential search. Both node and link data are important for route searching, which is the most essential function in navigation systems. Both node and link data are organized in different tables in a database, and connectivity relationships between two tables are matched with their respective ID field values. If traversing through several hundreds of thousands node and link records, the performance cannot be guaranteed even for the index-generated ID field. The mobile spatial DBMS developed in this research does not use an index but allows accessing data records using the physical address (physical record ID) for a particular field.

Figure 9 shows a difference between record access by index

Table 1. Features of mobile spatial DBMS.

| Function | | Mobile DBMS |
|----------------------|----------|---|
| SQL/API | | C/C++ API |
| DDL | | create / drop table-space, table, index |
| DML | | insert, update, delete |
| Search | | attribute search (match, initial sound) spatial search (windows, K -NN, relational search) network search (physical record ID search) |
| Transaction/recovery | | begin, commit, redo, undo |
| Object | | table (general table, network table) access path (B+-tree, multilevel-GRID, R*-tree) |
| Data type | General | char, varchar, int, uint, short, double, float, binary, date, time, blob |
| | Geometry | point, linestring, polygon |
| | Topology | node, link |
| Flash-aware | | block-level spatial clustering |
| Storage media | | disk, flash memory |
| Supported platform | | Windows mobile, Android |

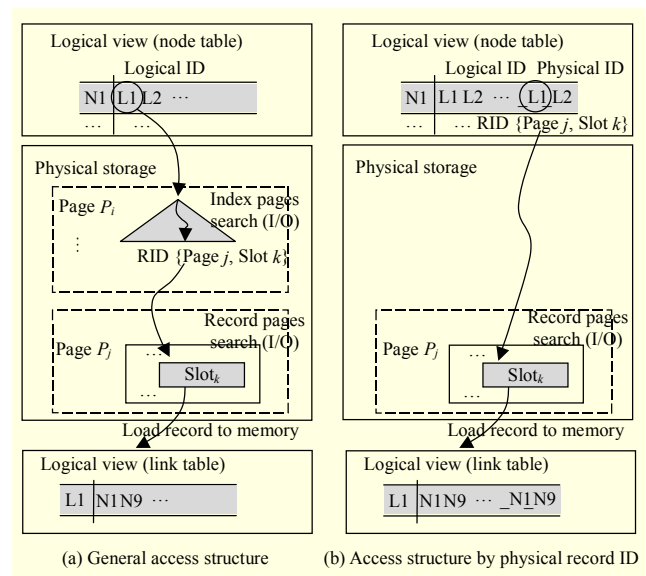


Fig. 9. Physical record ID access for large network data.

and by physical record IDs.

In Fig. 9(a), there is a node record N1 at the node table having a connectivity relationship with link records L1, L2, In order to access the link record connected to this node record, the link table record will be searched using the ID value of L1, L2, The record searching process is as follows: first, search the record ID corresponding to the indexed values of L1, L2, ... at the physical index page; second, search the data slot corresponding to the record ID at the physical data page. The search cost is $COST_{IO} = \# \text{ of IndexPage}_{IO} + \text{RecordPage}_{IO}$.

If there are too many records, the performance degrades because the number of index pages increases. Method in Fig. 9(b) proposed by this research generates a physical record ID corresponding to the logical record ID of the link connected to the node record in advance and searches quickly using this value. The search cost is $COST_{IO} = \text{RecordPage}_{IO}$.

To generate the table in Fig. 9(b), an additional process that finds and maps physical record ID values is required. For a record update, a little overhead exists in that the physical record ID needs to be updated.

C. Flash-Awareness for Partial Map Update

Recently, as smart phone use is expanding, various mobile services are becoming very popular. In particular, there are many services that use map content, such as navigation, location-based, and POI services. Flash memory media is usually used for storage media in mobile devices. This has different characteristics from general storage media. Flash memory is nonvolatile, write cost is ten times lower than read cost, and read/write cost is asymmetric. If we write on a sector that has already been recorded on, the block has to be erased; the access unit is different, and the erase count is limited. Also, the read performance is considerably faster because it requires no seek or rotational time.

Due to these characteristics, flash memory performs well for sequential writes such as copy data; however, the performance is degraded for random writes. In most cases in the past, a sequential write was generally a full map update; however, there are mobile application services requiring a random write, such as an MAU. In this research, in order to enhance the update performance of map data, block-level spatial data clustering is developed. Map data has different characteristics from other general data. The most eminent characteristic is spatial locality. Spatial locality is managed by classifying layers semantically. For example, there are building layers, POI layers, road layers, and so on. If map data at a particular location is changed, the data existing at a location corresponding to most other layers can also possibly be changed. In the same context, for road network data, there is a road table for data display, as well as link and node tables for path computation. The link and

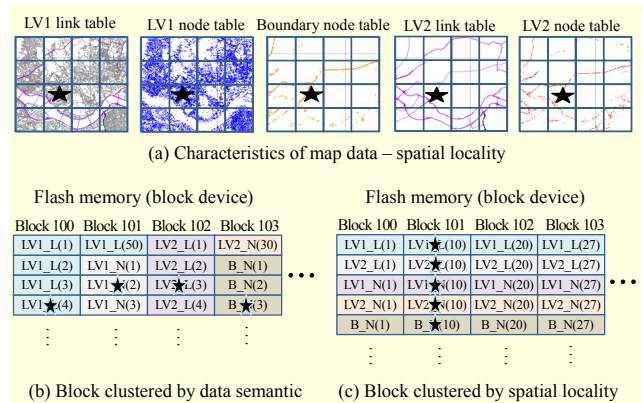


Fig. 10. Spatial locality and block clustering in flash memory.

node tables exist in multiple levels due to their hierarchical structure.

A level 1 link/node table and level 2 link/node table also exist, as in Fig. 10(a), as does a boundary node table to connect the node/link of these two levels. If the data at a particular location (an asterisk in the figure) is changed, the data at the corresponding location of all network data tables are generally changed. This is based on the spatial locality characteristic of map data. When the tables in the database are generated, it is usual to do a batch insertion record sequentially at the physical storage media. In this case, the data will be clustered with the table unit, as in Fig. 10(b). If the map data at a particular location is changed, many blocks will be erased for the update because the data at the corresponding location is scattered throughout the flash memory block as in Fig. 10(b). However, if we generate the tables by considering the spatial locality, the performance can be improved because the number of blocks to be erased is reduced as in Fig. 10(c), and the number of erase operations, which have a high cost, is reduced. Among the recent research on efficient managing methods for the blocks and sectors in FTL, applying a block-level clustering method by considering spatial locality is more effective for a partial map update.

2. Route Search and Air Update in a Mobile Device

A. Heuristic Route Searching Algorithm

The most popular algorithms used in route searches which are implemented for the car navigation systems [18] are Dijkstra, A*, bidirectional Dijkstra, and bidirectional A* algorithms. Heuristic techniques for route search algorithms are applied because of the generally limited resources available in mobile devices. The most general heuristic methods generate hierarchical road network data and limit the search space.

Multilevel network data tables. This study constitutes road network data with two levels as a heuristic method to enhance

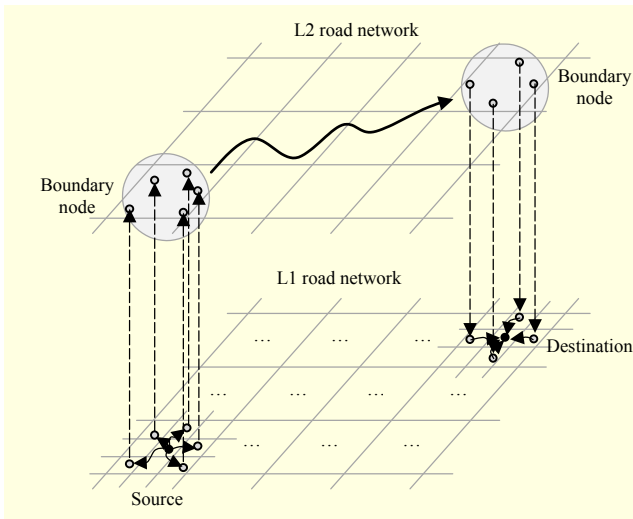


Fig. 11. Route search in hierarchical road network data.

the route search performance. The method generates node, link, and partition tables corresponding to each level, and also a boundary node table to connect low-level and high-level road network data. The level 1 network data includes whole network data, while level 2 network data includes major road data except for detailed roads. For short-distance route searches, the algorithm is executed by searching the level 1 network data table, and for mid/long-distance route searches, both the level 1 and level 2 network data tables are searched. Figure 11 shows the route search algorithm in the hierarchical road network data, the procedure of which is as follows:

- i) Source route in level 1 road network data: route search of the boundary node from the source to the upper level.
- ii) Destination route in level 1 road network data: route search of the boundary node around the destination.
- iii) Major route in level 2 road network data: route search from the result of i) to the result of ii).

Finally, the resultant route is searched by merging the results of i), ii), and iii).

Restriction of search space. During a route search in a conventional navigation system, after finding the expected partition set, all node/link data included in the partition set is loaded into the memory, and the route is searched. The mobile spatial DBMS of this study loads the expected partition identification value into memory, and the node/link records included in this partition are only traversed during the record search.

B. Air Update Road Network Data.

Generally, the most fundamental functions in DBMS are insert/update/delete of a record. In the case of a partial update of geometry and POI data in navigation map data, insert/update/delete queries can be used in the mobile spatial DBMS

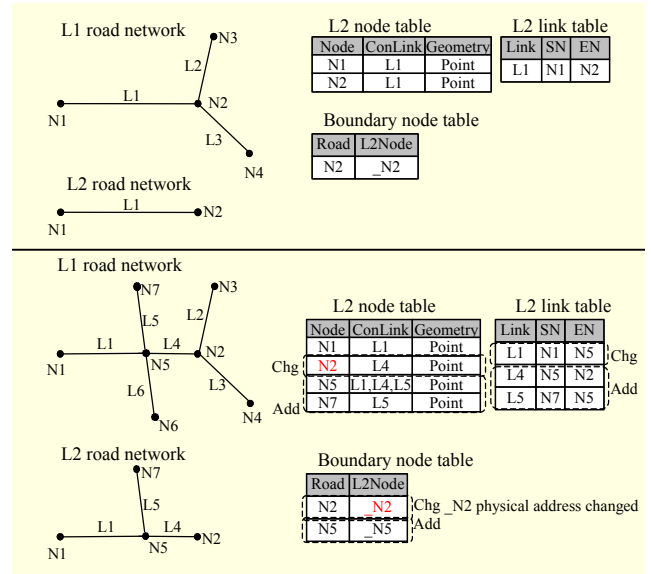


Fig. 12. Partial update road network data in MAU_{MN}.

of MAU_{MN} without particular logic. However, for road network data, the following procedure is used because MAU_S uses a level 1 network data model and MAU_{MN} uses a hierarchical data model.

- i) Extract level 1 node/link data and update tables.
- ii) Map physical record ID each level 1 on network table record.
- iii) Extract level 2 node/link/boundary node data and update tables.
- iv) Map physical record ID each level 2 on network table records.

Figure 12 shows an example of partial update network data. If a new road is added, the node/link/boundary node data corresponding to level 2 is extracted, and the table is updated. Boundary node N2 has no logical changes, but the N2 value has to be updated because the N2 node record of the level 2 node table is changed. For road network data, additional steps of record extraction corresponding to level 2 and re-mapping of the physical record ID are required. Section VI shows the experimental results of this.

VI. Experiments

The functions and performances of the MAU_S and MAU_{MN} experimentation in this study are explained in this section.

Figure 13 shows a simple navigation application for an MAU. Figure 13(a) shows the discovery service, and Figs. 13(b) and (c) show the provisional service via wireless telecommunications, a road network data update and geometrical data update, respectively.

For the performance experimentation, the following factors

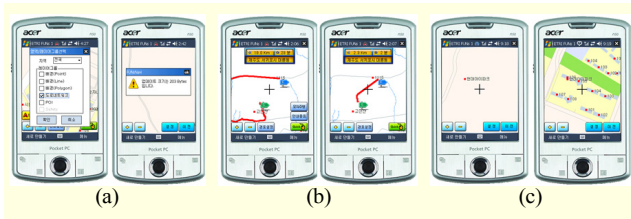


Fig. 13. MAU navigation: (a) discovery service, (b) update road network data, and (c) update geometrical data.

Table 2. Baseline map dataset.

| Data | Size (MB) | Description | DB file |
|---------|-----------|--|-------------|
| Display | 400 | 68 tables, L1 through L6 | Map.tbs |
| POI | 300 | 1 table, 500,000 records | |
| Network | 400 | 9 tables, node: 600,000 records, link: 700,000 records | Network.tbs |

were studied in a mobile device: non-spatial insert, update, delete, and select query performance, spatial search query performance, route search query performance, and air update performance.

The actual navigation map dataset used for experimentation is shown in Table 2. The model used as a test device is an LG KC1, which has a 800 MHz CPU, 128 M of DRAM, and 2 GB of flash memory.

1. Experiments on Fundamental Queries

First, experimentation on fundamental insert/delete/update/select queries is compared with commercial mobile DBMS [6]. Figure 14 shows the results of our experiments. Each query was executed about 10,000 times, with the unit of performance evaluation for the y -axis value being transaction per second (TPS). For the insert/delete/update queries, regardless of the use of index, the MAU_{MN} performance is better than in [6].

2. Experiments on Spatial Search Query

Among spatial search functions, the window query is the most popular one used in navigation. For map data visualization at a current navigation location, the window query is used for map data searches at the corresponding location. The method of experimentation is as follows. The location value is stored in the geometry field in the POI table, which stores 500,000 records, and the experiment is carried out by randomly generating query windows corresponding to 1%, 5%, 10%, and 20% of the entire map region. The experiment was repeated 1,000 times, the results of which are shown in Fig. 15. The y -axis shows the average time for executing a window

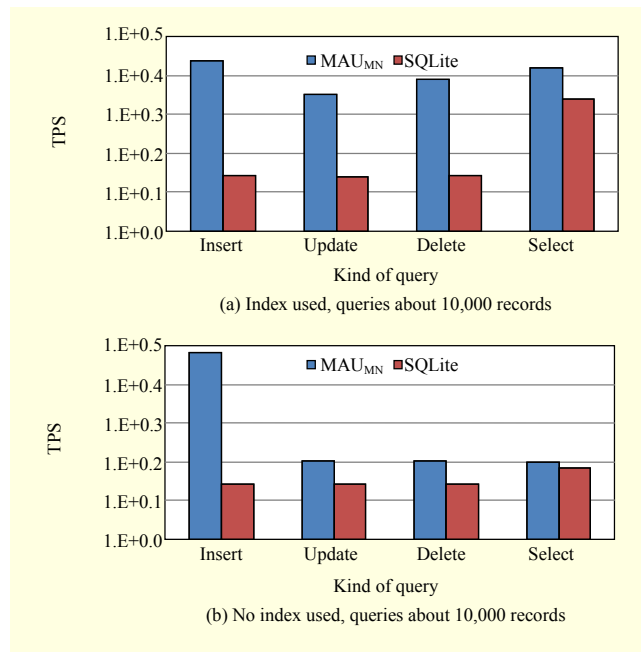


Fig. 14. Fundamental query performance comparisons.

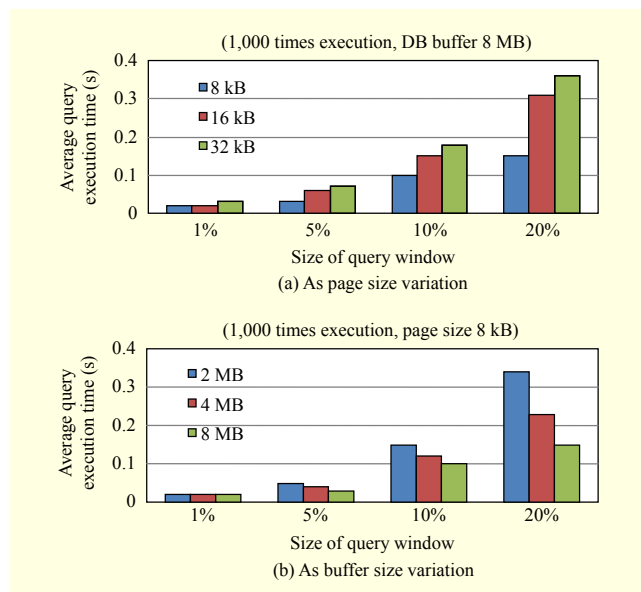


Fig. 15. Window query performance

query, while the x -axis shows the size of the query window. Figure 15(a) shows the experimental results of page size variation, while Fig. 15(b) shows the results of buffer size variation of mobile spatial DBMS. The window query performance is best when the page size is 8 kB and the buffer size is 8 MB. Another element affecting the window query performance is spatial clustering. The amount of physical pages accessed is reduced based on the spatial objects that exist spatially nearby and are stored closely to the physical storage. Consequently, the performance is enhanced.

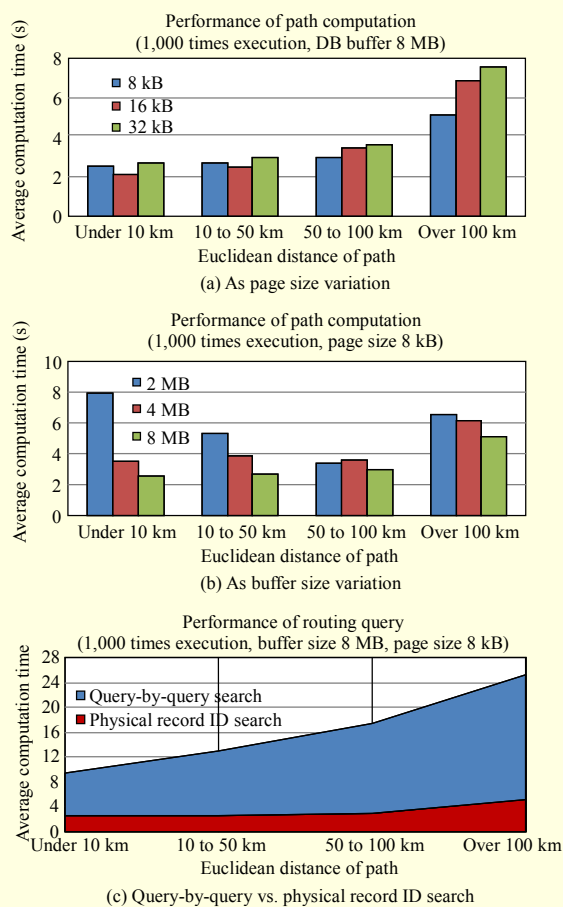


Fig. 16. Route searching performance.

3. Experiments on Route Search Query

The experiments on route search queries were carried out using real road network data. A route search was executed 1,000 times by randomly selecting the source and destination for a Euclidean distance of under 10 km, between 10 km to 50 km, between 50 km to 100 km, and over 100 km. A route search algorithm is operated such that only level 1 road network data is used for under 10 km, and a multilevel road network is used for over 10 km. Figure 16 shows the results. The y-axis shows the average time for executing a route search query, and the x-axis shows the Euclidean distance value. Figure 16(a) shows the experimental results for page size variation, while Fig. 16(b) shows the results for buffer size variation of mobile spatial DBMS. The route search query performance is best when the page size is 8 kB and the buffer size is 8 MB. As shown in Fig. 16(b), the performance is degraded because, in major cities, page I/O occurs frequently as the road network data is large and the buffer size is small.

In Fig. 16(c), a physical record ID used for searching network data is compared to a case in which a physical record

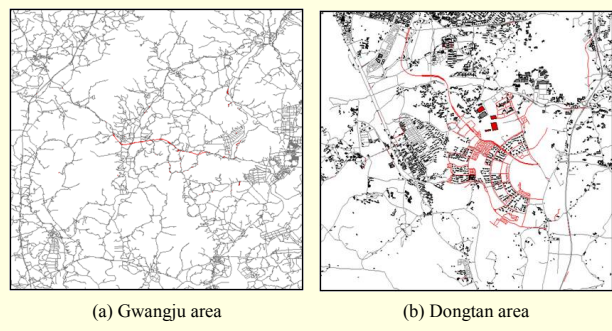


Fig. 17. Areas of MAU.

Table 3. Update dataset and update time.

| Update area | Update object | Update time |
|-------------|--|-------------|
| Gwangju | Network 40 kB (Del #: 49, Add #: 264, Chg #: 89) | 12 s |
| Dongtan | Network 100 kB (Del #: 106, Add #: 1,103, Chg #: 124) | 20 s |

ID is not used. Using a physical record ID provides more than four times the performance of a query-by-query search.

4. Experiment of Map Air Update

The partial MAU experiment using wireless telecommunications utilized real update data of Gwangju and Dongtan, Korea. As shown in Fig. 17, the Gwangju region has road network data as its update data and the Dongtan region includes road network and geometry data for visualization. The update map objects of each region are marked in red.

Table 3 provides details of the update data. A network data update takes more time compared to a geometry data update because it has to generate level 2 network data and has overhead in assigning a physical record ID value.

VII. Conclusion

In this paper, MAU_S and MAU_{MN} were developed for an air update of navigation map data, and the performance was evaluated experimentally. To provide partial map data, components, such as version control, aggregation method, and partial map data provisioning, are developed in MAU_S. The mobile spatial DBMS was developed to solve the data storage format problem, which is one reason why a partial map update is not possible in existing navigation systems. Real update map data was transmitted to a mobile device wirelessly, and the partial map update function was demonstrated for MAU_{MN}. The performance was also validated by carrying out experiments: fundamental queries, spatial search queries, and

route search queries using large-scale navigation map data.

The system framework for map air update navigation in this research is expected to provide high-quality navigation services. It is also expected that the mobile spatial DBMS of this research can be used to easily and effectively develop not only the navigation updates, but also various application updates, such as LBS, telematics, and mobile GIS.

References

- [1] H Fujimoto, "World Wide Vehicle Navigation System Using KIWI Format," *DENSO Tech. Rev. J.*, vol. 6, no. 1, 2001, pp. 29-34.
- [2] ISO GDF-Geographic Data Files-Version4.0 (Draft International Standard ISO/CD 2001-02-14): ISO/TC 204 N 34.2001.
- [3] J.B. Bullock and E.J. Krakiwsky, "Analysis of the Use of Digital Road Maps in Vehicle Navigation," *IEEE Position Location Navigation Symp.*, Las Vegas, NV, Apr.11-15, 1994, pp. 494-501.
- [4] ERTICO Project. Available: <http://www.ertico.com/actmap/>
- [5] Y.J. Joo, J.Y. Kim, and S.H. Park, "Design and Implementation of Map Databases for Telematics and Car Navigation Systems using an Embedded DBMS," *J. GIS Association of Korea*, vol. 14, no. 4, Dec. 2006, pp. 379-389.
- [6] <http://www.sqlite.org/>
- [7] EPEL, U. Grenoble, INRIA-Nancy, INT-Evry, U. Montpellier, U. Paris, U. Versailles, "Mobile Databases: a Selection of Open Issues and Research Directions," *SIGMOD Record*, vol. 33, no. 2, June 2004, pp. 78-83.
- [8] S.W. Lee et al., "Research Issues in Next Generations DBMS for Mobile Platforms," *Mobile HCI, ACM*, Sept. 9-12, 2007.
- [9] E. Gal and S. Toledo, "Algorithms and Data Structures for Flash Memories," *ACM Computing Surveys*, vo. 37, no. 2, June, 2005.
- [10] J.S. Kim et al., "A Space-Efficient Flash Translation Layer for Compact Flash Systems," *IEEE Trans. Consum. Electron.*, vol. 48, no. 2, 2002.
- [11] S.W. Lee and B.K. Moon, "Design of Flash-Based DBMS: An In-Page Logging Approach," *SIGMOD*, 2007.
- [12] S. Lee et al., "Locality-Aware Sector Translation for NAND Flash Memory-Based Storage Systems," *SIGOPS Oper. Syst. Rev.*, vol. 62, no. 6, 2008, pp. 36-42.
- [13] J.K. Yun et al., "Development of an Embedded Spatial MMDBMS for Spatial Mobile Devices," *Proc. 5th Int. Workshop Web Wireless Geographical Inf. Syst.*, 2005, pp. 1-10.
- [14] MAUS-Terminal Service Protocol for Map Air Update (TTA.KO-06.0130). Available: <http://www.tta.or.kr>
- [15] OGC Implementation Specification for Geographic Information-Simple Feature Access – Part 1: Common Architecture, vol. 2, 2006.
- [16] K.Y. Whang and R. Krishnamurthy, "The Multilevel Grid File-A Dynamic Hierarchical Multidimensional File Structure," *Proc.*

Int. Conf. Database Syst. Adv. Appl., Tokyo, Japan, Apr. 1991, pp. 449-459.

- [17] N. Beckmann et al., "The r*tree: An Efficient and Robust Access Method for Points and Rectangles," *Proc. ACM SIGMOD Int. Symp. Manag. Data*, 1990, pp. 322-331.
- [18] J. Kim et al., "Proposal for an Inundation Hazard Index of Road Links for Safer Routing Services in Car Navigation Systems," *ETRI J.*, vol. 32, no. 3, June 2010, pp. 430-439.



Kyoungwook Min received the BS and MS in computer engineering from Pusan National University, Rep. of Korea, in 1996 and 1998, respectively. Since 2001, he has been a senior member of research staff with ETRI, Rep. of Korea, and he is also working toward the PhD in computer engineering from Chungnam National University, Rep. of Korea. His main research areas are navigation service, unmanned ground vehicle, and location-based service.



Kyoungwan An received the BS, MS, and PhD in computer engineering from Pusan National University, Rep. of Korea, in 1997, 1999, and 2004, respectively. Since 2005, he has been a senior member of research staff with ETRI, Rep. of Korea. His main research areas are platforms for unmanned ground vehicle, navigation service, and location-based service.



Insung Jang received the BS and MS in computer engineering from Pusan National University, Rep. of Korea, in 1999 and 2001, respectively. Since 2001, he has been a senior member of research staff with ETRI, Rep. of Korea, and he is also working toward the PhD in computer engineering from Pusan National University, Rep. of Korea. His main research areas are platforms for geo-sensor service, navigation service, and location-based service.



Sungil Jin received the BS in computer engineering from Seoul National University in 1978, and the MS and PhD in computer engineering from Korea Advanced Institute of Science and Technology (KAIST), Rep. of Korea, in 1980 and 1994, respectively. He joined the faculty of the Department of Computer Science at Chungnam National University, Rep. of Korea, in 1981. His research interests are database management systems and geographic information systems.