J Inf Process Syst, Vol.13, No.5, pp.1259~1276, October 2017
https://doi.org/10.3745/JIPS.04.0047

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Spatio-temporal Sensor Data Processing Techniques

Jeong-Joon Kim*

Abstract

As technologies related to sensor network are currently emerging and the use of GeoSensor is increasing along with the development of Internet of Things (IoT) technology, spatial query processing systems to efficiently process spatial sensor data are being actively studied. However, existing spatial query processing systems do not support a spatial-temporal data type and a spatial-temporal operator for processing spatial-temporal sensor data. Therefore, they are inadequate for processing spatial-temporal sensor data like GeoSensor. Accordingly, this paper developed a spatial-temporal query processing system, for efficient spatial-temporal query processing of spatial-temporal sensor data in a sensor network. Lastly, this paper verified the utility of System through a scenario, and proved that this system's performance is better than existing systems through performance assessment of performance time and memory usage.

Keywords

Multi-dimensional Operator, Multi-dimensional Spatio-temporal Data Type, Sensor Networks, Sensor Query Processing System

1. Introduction

Recently, sensor query processing systems (such as TinyDB, TikiriDB, etc.) have been studied to query the sensor data in sensor networks [1-4]. Furthermore, we extended spatial extension of TinyDB (SE TinyDB), spatial TinyDB, and TikiriDB, which extend TinyDB to enable two-dimensional query processing for query processing of two-dimensional spatial sensor data such as GeoSensor Query processing systems such as spatial extension of TikiriDB (SE TikiriDB) have been studied [5]. In GeoSensor, space-time sensor data, which is multi-dimensional data, is collected as well as two-dimensional space sensor data. However, existing query processing systems do not support data types and operators for handling these multi-dimensional data, and thus are not sufficient for processing multi-dimensional sensor data [6].

Therefore, in this paper, we have developed spatio-temporal TinyDB (STTinyDB) to efficiently provide query processing for multi-dimensional data in sensor networks. STTinyDB developed in this paper extends and develops TinyDB, a representative sensor query processing system, to process multi-dimensional sensor data. For interoperability, "Simple Features Specification for (Open Geospatial Consortium)" proposed by OGC SQL standard [7,8]. The overall system configuration for STTinyDB in

Manuscript received April 7, 2017; first revision May 22, 2017; accepted June 4, 2017.
Corresponding Author: Jeong-Joon Kim (jjkim@kpu.ac.kr)
* Dept. of Computer Science & Engineering, Korea Polytechnic University, Siheung, Korea (jjkim@kpu.ac.kr)

this paper consists of interface manager, data communication manager, multi-dimensional data manager, multi-dimensional operation manager, and multi-dimensional data stream manager.

Spatio-temporal data type, space-time operator, filtering, etc., which are core technologies of spatio-temporal sensor data processing technology researched and developed in this paper, require sensor data processing technology and various application services such as IOT, big data, cloud and can contribute to revitalization of related industries.

2. Related Research

2.1 SE TinyDB

To handle spatial queries in sensor networks, SE TinyDB was developed by extending TinyDB into space [9]. SE TinyDB adds three spatial operators, Distance, Inbox, and Beyondboundary to TinyDB. Table 1 shows the types of spatial operators provided by SE TinyDB.

Table 1. SE TinyDB spatial operator

Operator	Explain
Distance	Distance(X1, Y1, X2, Y2) : Double
Inbox	Inbox(Xmin, Ymin, Xmax, Ymax) : Boolean
Beyondboundary	Beyondboundary(Xmin, Ymin, Xmax, Ymax) : Boolean

As shown in Table 1, the Distance operator returns the difference in coordinate distance between objects, the Inbox operator returns whether it is contained within a specific area (rectangle), and the Beyondboundary operator returns whether it is outside the boundary of a specific area (rectangle) do. Table 2 shows examples of spatial queries using spatial operators of SE TinyDB.

SE TinyDB provides three spatial operators as described above to support query processing for specific regions or movement trajectories. However, since it does not provide space-time data type and space-time operator for space-time sensor data processing, it is insufficient to process space-time query.

Table 2. Spatial TinyDB Spatial Data Type

Data type	Example
point	point(10 10)
lineString	lineString(10 10, 20 20, 30 40)
polygon	polygon(10 10, 10 20, 20 20, 20 15, 10 10)
polyhedralSurface	polyhedralSurface((10 10, 10 20, 20 20, 20 10, 10 10), (20 10, 40 20, 20 20, 20 10))
multiPoint	multiPoint(10 10, 20 20)
multiLineString	multiLineString((10 10, 20 20), (15 15, 30 15))
multiPolygon	multiPolygon((10 10, 10 20, 20 20, 20 15, 10 10), (60 60, 70 70, 80 60, 60 60))

2.2 Spatial TinyDB

In order to provide various services related to u-GIS, u-LBS, u-Learning, u-City, u-transportation, u-logistics, etc., in ubiquitous sensor network (USN) environment, Spatial TinyDB has been developed [10]. Spatial TinyDB provides two-dimensional spatial data types and spatial operators conforming to the "Simple Features Specification for SQL" standard specification proposed by OGC in TinyDB. Table 2 shows examples of spatial data types provided by Spatial TinyDB.

As shown in Table 2, the two-dimensional spatial data type supports seven data types such as point, lineString, polygon, polyhedralSurface, multiPoint, multiLineString, and multiPolygon. Table 3 shows the types of operators provided by Spatial TinyDB for query processing in sensor nodes.

As shown in Table 3, eight relational operators, six analytic operators, and five trajectory operators are supported.

Table 3. Spatial TinyDB operator

Operator type	Operator
Relation operator	Equals, Disjoint, Touches, Within, Overlaps, Crosses, Intersects, Contains
Analysis operator	Distances, Intersection, Difference, Union, Buffer, ConvexHull
Trajectory operator	Enters, Insides, Leaves, Meets, Passes

2.3 SE TikiriDB

In order to process two-dimensional spatial data queries in sensor networks, SE TikiriDB was developed by extending TikiriDB [11]. SE TikiriDB provides some of the data types and operators that conform to the "Simple Features Specification for SQL" standard specification presented by OGC in TikiriDB. Table 4 shows the types of operators provided by SE TikiriDB.

Table 4. SE TikiriDB operator

Operator	Explain
Distance	Distance(geometry A, geometry B) : Double
Disjoint	Disjoint(geometry A, geometry B) : Boolean
Contains	Contains(geometry A, geometry B) : Boolean
Within	Within(geometry A, geometry B) : Boolean

As shown in Table 4, the Distance operator returns the coordinate distance of the spatial objects A and B, and the Disjoint operator returns whether the spatial objects A and B did not meet in space. The Contains operator returns whether the spatial object A spatially includes the spatial object B, and the within operator returns whether or not the spatial object B spatially includes the spatial object A.

3. System Design and Implementation

The STTinyDB developed in this paper is a query processing system that efficiently processes multi-dimensional sensor data by extending TinyDB to provide various services related to military, medical,

weather, environment, transportation, home, and company in sensor network. Fig. 1 shows the overall system structure of STTinyDB designed in this paper.

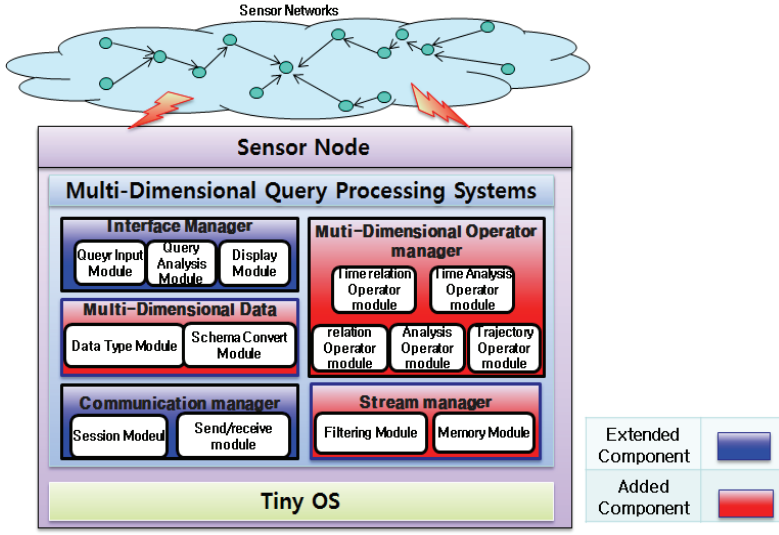

Fig. 1. Architecture of system.

As shown in Fig. 1, STTinyDB consists of a data communication manager, a data stream manager, a multi-dimensional data manager, an interface manager, and a multi-dimensional operation manager. The interface manager consists of a query input module that receives a query from the user, a query analysis module that analyzes the query token/parse, and a display module that displays the final result of the query processed by each sensor node. Provides a function to generate spatio-temporal query by selecting sensor/space-time property and operator through interface created by GUI. It also provides a function to set the query execution cycle, filtering condition, query ID, and the function to transfer the input space-time query to the space-time query analysis module. Table 5 shows the syntax and query examples of spatio-temporal queries supported by STTinyDB.

As shown in Table 5, various time and space-time queries are possible using time relation / analysis operator and space-time relation / analysis / trajectory operator.

The data communication manager consists of a session management module that manages sessions between the sensor nodes in the area to query the area entered by the user, and a data transmission / reception module that manages communication between the sensor nodes in the area.

The data stream manager is a filtering module that filters various filtering conditions to reduce the input load due to the constantly inserted multi-dimensional sensor data that is inserted when processing the query. To improve the memory efficiency among multiple queries, and a memory management module for managing the memory management module.

The filtering processing module provides a function to filter data satisfying a specific condition while minimizing the accuracy of the data in order to solve the overload problem by reducing the amount of the data stream. The condition can be set by applying the time interval difference, position coordinate difference, etc. to the attribute. For example, if the current data is compared with the previous data for the same object, the data is filtered if the data is equal to or less than the allowable range.

Table 5. Type and example of spatio-temporal query

Spatio-temporal query syntax	
SELECT select-list [FROM sensors] WHERE where-clause [EPOCH DURATION]	
Spatio-temporal query classification	**Spatio-temporal query example**
Time relational operators	SELECT nodeid, time, temp FROM sensors WHERE tDisjoint(tPeriod(2015/05/01 12:00:00, 2015/05/01 13:00:00), time)
Time analysis operator	SELECT nodeid, time, temp FROM sensors WHERE tContains(tUnion(tPeriod(2015/05/01 12:00:00, 2015/05/01 14:00:00), tPeriod(2015/05/01 13:00:00, 2015/05/01 15:00:00)), time)
Spatio-temporal relational operators	SELECT nodeid, temp, stLoc FROM sensors WHERE stDisjoint(stPolygon(tPeriod(2015/05/01 14:00:00, 2015/05/01 15:00:00), 0 0, 0 700, 700 700, 700 0, 0 0), stLoc)
Spatio-temporal analysis operators	SELECT nodeid, temp, stLoc FROM sensors WHERE stContains(stIntersection(stPolygon(tPeriod(2015/05/01 12:00:00, 2015/05/01 18:00:00), 0 0, 0 400, 400 400, 400 0, 0 0), stPolygon(tPeriod(2015/05/01 16:00:00, 2015/05/01 19:00:00), 100 100, 100 500, 500 500, 500 100, 100 100)), stLoc)
Space-time trajectory operator	SELECT nodeid, temp, stLoc FROM sensors WHERE stEnters(stPolygon(tPeriod(2015/05/01 12:00:00, 2015/05/01 23:00:00), 0 0, 0 100, 100 100, 100 0, 0 0), stLoc)

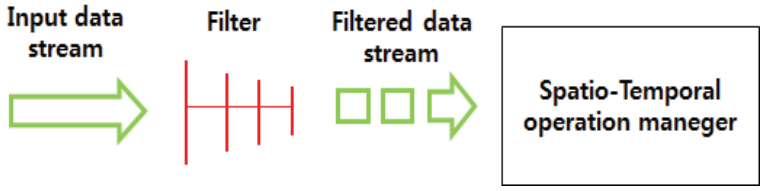

Fig. 2. Filtering process.

Fig. 2 shows the process of filtering the collected sensor data. As shown in Fig. 2, the input data stream (sensor data) passes through the filter. In the filter, it checks whether the filtering condition set by the user meets the filtering condition, and transmits the sensor data over the allowable range to the space time operation manager. Fig. 3 shows an example of filtering space-time sensor data by applying time difference condition and position coordinate difference condition.

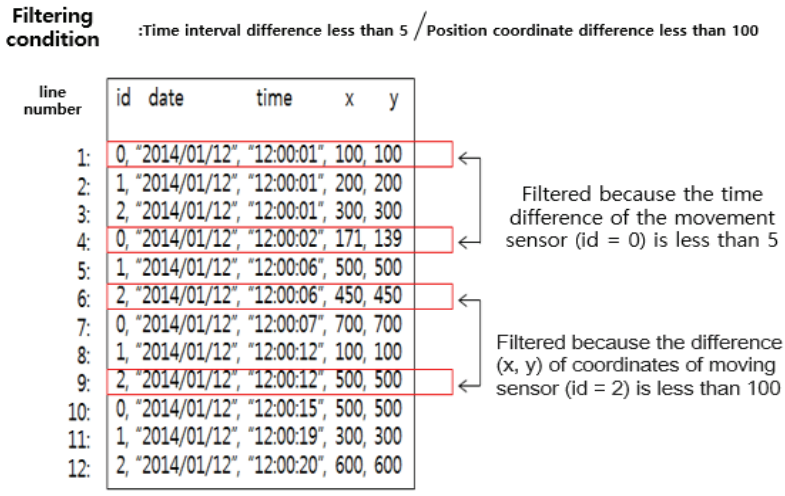

Fig. 3. Filtering example.

In Fig. 3, two filtering conditions are applied. First, the time interval difference is less than 5, and the time unit is second (s) or min (m). The second condition is a coordinate distance difference of 100 or less, and the distance unit is centimeter (cm) or meter (m). These conditions are applied sequentially. The line number 4 is filtered because the time difference of the previous time is less than 5, and the line number 9 is filtered because the time difference of the previous time is 5 or more, but the position coordinate difference is less than 100.

The memory management module provides a function to share necessary data in common in one memory area because the memory capacity is small when a plurality of space-time queries are simultaneously performed on one sensor node. This way, you can use memory efficiently because you do not have memory areas for each query. In this case, the shared memory is managed by information such as a tuple number, a reference count, and a reference time for each data tuple through a space-time reference table.

In the space-time reference table, information of a data tuple is linked to a linked list, and when the reference count is '0' in the space-time reference table, information of the data tuple is deleted. If the space-time reference table is full and information of the new data tuple is added to the space-time reference table, the information of the data tuple having the smallest reference count among the data tuple information of the space-time reference table is deleted. If the number of references is the same, delete the oldest tuple of the same one with the lowest reference time.

Fig. 4 shows an example of memory sharing for multiple queries. As shown in Fig. 4, when the space-time query comes in the order of stContains, stDisjoint, and stEnters, the memory management module operates basically. (n9, t9, g9)} (n10, t10, g10), (n11, t11, g11), and (n12, t12, g12) are stored in the storage when the first space time query 1 (stContains) (Four data tuples). '(2)' updates the space-time reference table with respect to space-time query 1. '(3)' checks if the space-time query 2 (stDisjoint) arrives and can share the data already stored in the storage. Here, {(n10, t10, g10), (n11, t11, g11), (n12, t12, g12)} are shared. '(4)' updates the space-time reference table for space-time query 2. '(5)' checks whether the space-time query 3 (stEnters) is shared among the data already stored in the storage. Here we share {(n10, t10, g10)} data. '(6)' updates the space-time reference table for space-time query 3. If

space-time query 1 is completed, tuple number 9 of the space-time reference table becomes '0', and is deleted from the space-time reference table.

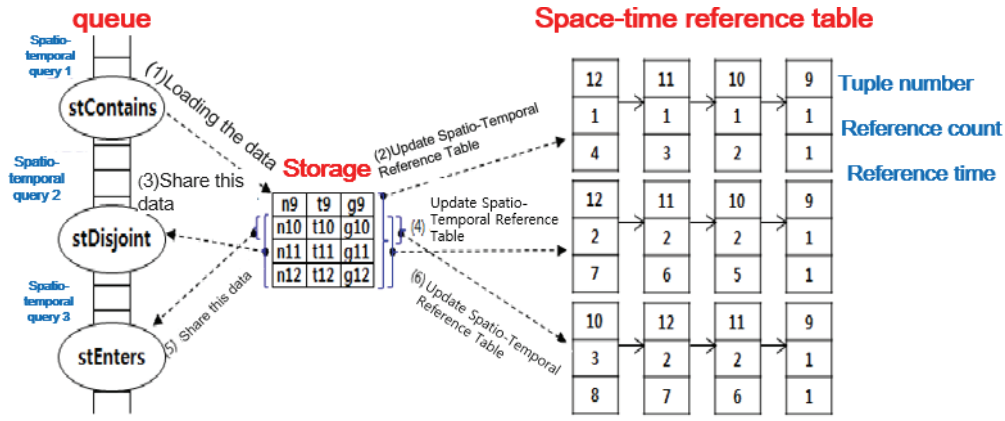

Fig. 4. Memory sharing example.

The multi-dimensional data manager is a multi-dimensional data type module based on the two-dimensional spatial data type of the OGC "Simple Features Specification for SQL" standard, and it quickly converts the attribute information provided by TinyDB into a multi-dimensional property according to a multi-dimensional schema mapping rule Dimensional schema conversion module.

The Spatio-temporal Data Type module provides a spatio-temporal data type based on the spatial data type specified in "Simple Features Specification for SQL", one of the OGC standards, to support spatio-temporal queries.

Fig. 5 shows the hierarchy of spatio-temporal data types supported by the spatio-temporal data type module.

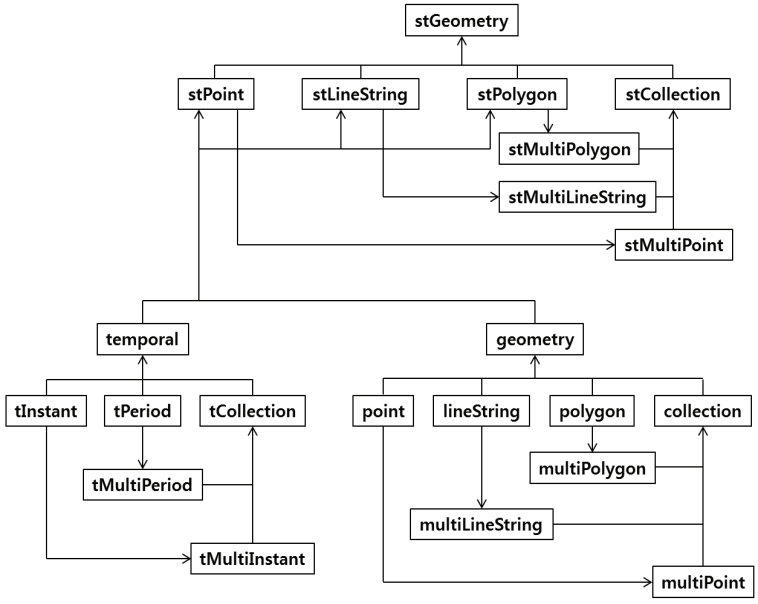

Fig. 5. Spatio-temporal data type.

As shown in Fig. 5, there is stGeometry type, which means space-time in the uppermost layer, and a geometry type, which means temporal type and space, which means time. The temporal type consists of an instant (tInstant) which means time according to the characteristic of time, a period (tPeriod) which means start time and end time (stTime, edTime), a tMultiInstant which means instant set which is bundled into one or more, TMultiPeriod, which means a set of one or more chained periods, and tCollection, which is a mixture of instant and period. The geometry type provides point, lineString, polygon, multiPoint, multiLineString, multiPolygon, and collection conforming to the OGC "Simple Features Specification for SQL" standard specification. The stGeometry type provides the stPoint, stLineString, stPolygon, stMultiPoint, stMultiLineString, stMultiPolygon, and stCollection extensions to space-time based on temporal and geometry types.

The stGeometry type provides a spatio-temporal data type based on temporal and geometry types.

Table 6 shows examples of spatio-temporal data types in the stGeometry hierarchy.

Table 6. Spatio-temporal data type

Spatio-temporal data type	Example
stPoint(temporal A, geometry B)	stPoint(tInstant(2015/01/15 09:00:00), 10 10)
stLineString(temporal A, geometry B)	stLineString(tInstant(2015/01/15 09:00:00), 10 10, 20 20)
stPolygon(temporal A, geometry B)	stPolygon(tInstant(2015/01/15 09:00:00), 10 10, 10 20, 20 20, 20 15, 10 10)
stMultiPoint(stPoint A, stPoint B, ...)	stMultiPoint(stPoint(tInstant(2015/01/15 09:00:00), 10 10), stPoint(tInstant(2015/01/15 10:00:00), 20 20))
stMultiLineString(stLineString A, stLineString B, ...)	stMultiLineString(stLineString(tInstant(2015/01/15 09:00:00), 10 10, 20 20), stLineString(tInstant(2015/01/15 10:00:00), 30 30, 40 40))
stMultiPolygon(stPolygon A, stPolygon B, ...)	stMultiPolygon(stPolygon(tInstant(2015/01/15 09:00:00), 10 10, 10 20, 20 20, 10 10), stPolygon(tInstant(2015/01/15 10:00:00), 60 60, 70 70, 80 60, 60 60))
stCollection(stGeometry A, stGeometry B, ...)	stCollection(stPoint(tInstant(2015/01/15 09:00:00), 10 10), stLineString(tInstant(2015/01/15 09:00:00), 10 10, 20 20))

As shown in Table 6, the spatio-temporal data types include stPoint, which is a spatio-temporal point object, stLineString, which is a space-time line object, stPolygon, which is a spatio-temporal polygon object, stMultiPoint, which is a multi-StMultiLineString, stMultiPolygon, which means multiple space-time polygon objects, and stCollection, which represents a space-time object as a collection.

The multi-dimensional operation manager is composed of a time relation/analysis operator processing module that performs time operation processing, a space time relation/analysis operator processing module for multi-dimensional space time operation processing, and a space time trajectory operator processing module that processes a moving trajectory of a moving sensor node.

The Spatio-temporal Operator Processing Module extends the standard specification of OGC's "Simple Features Specification for SQL" to provide space-time relational operators to support space-time operations in sensor nodes. This operator is divided into a timestamp operator and an interval operator, and returns true or false as the return value for the operation.

Table 7 shows the types of time stamp/interval time-space relational operators.

Table 7. Time stamp/interval spatio-temporal relation operator

Operator	Explain
stEquals(stGeometry A, stGeometry B) : Boolean	Returns whether a space-time object is equal to a specific time or a specific time interval A and a specific time or a specific time interval B
stCrosses(stGeometry A, stGeometry B) : Boolean	Returns whether a space-time object intersects a specific time or a specific time interval A and a specific time or a specific time interval B
stAppears(stGeometry A, stGeometry B) : Boolean	Returns whether a spatio-temporal object of a specific time or a specific time interval B appears at a specific time or a specific time interval A
stDisappears(stGeometry A, stGeometry B) : Boolean	Returns whether a spatio-temporal object at a specific time or a specific time interval B has disappeared at a specific time or a specific time interval A
stContains(stGeometry A, stGeometry B) : Boolean	Returns whether a specific time or a specific time interval A contains a spatio-temporal object of a specific time or a specific time interval B
stLeftContains(stGeometry A, stGeometry B) : Boolean	Returns whether a spatio-temporal object of a specific time period B is included from the start time of a specific time or specific time interval A
stRightContains(stGeometry A, stGeometry B) : Boolean	Returns whether a spatio-temporal object is included in a specific time or a specific time interval A until a specific time or the end time of a specific time interval B
stDisjoint(stGeometry A, stGeometry B) : Boolean	Returns whether a specific time or a specific time segment A does not meet a spatio-temporal object at a specific time or a specific time segment B
stLeftDisjoint(stGeometry A, stGeometry B) : Boolean	Returns whether a spatio-temporal object of a specific time or interval B does not meet before a specific time or a specific time interval A
stRightDisjoint(stGeometry A, stGeometry B) : Boolean	Returns whether a spatio-temporal object of a specific time or interval B does not meet after a specific time or a specific time period A
stTouches(stGeometry A, stGeometry B) : Boolean	Returns whether the boundary of a space-time object of a specific time or a specific time interval B meets a specific time or a specific time interval A
stLeftTouches(stGeometry A, stGeometry B) : Boolean	Returns whether the start time of a specific time or specific time interval A and the time-to-space object of a specific time or end time of a specific time interval B meet
stRightTouches(stGeometry A, stGeometry B) : Boolean	Returns whether the end time of a specific time or specific time interval A and the time-to-space object of a specific time or start time of a specific time interval B meet
stOverlaps(stGeometry A, stGeometry B) : Boolean	Returns whether the spatio-temporal objects overlap at a specific time or a specific time segment A at a specific time or a specific time segment B
stLeftOverlaps(stGeometry A, stGeometry B) : Boolean	Returns whether the spatio-temporal objects overlap at a specific time or at a specific time interval B at the start time of a specific time or specific time interval A
stRightOverlaps(stGeometry A, stGeometry B) : Boolean	Returns whether the spatio-temporal objects overlap at a specific time or at a specific time interval B at the end time of a specific time or a specific time interval A

As shown in Table 7, the time stamp/interval time-space relational operator receives two spatio-temporal objects A and B as input values, performs time-space operations, and returns true or false. For example, the stContains operator, stLeftContains, and stRightContains operators, provide the stLeftContains and stRightContains operators, dividing a specific space-time object before and after a stub, to distinguish and operate spatio-temporal objects.

The Spatio-temporal Analysis Operator Processing Module extends the standard specification of OGC's "Simple Features Specification for SQL" to provide space-time analysis operators to support space-time operations in sensor nodes. This operator is divided into a time stamp operator and an interval operator. The return value of the operation is a space-time data type.

Fig. 6 shows a representative example of space-time relational operators.

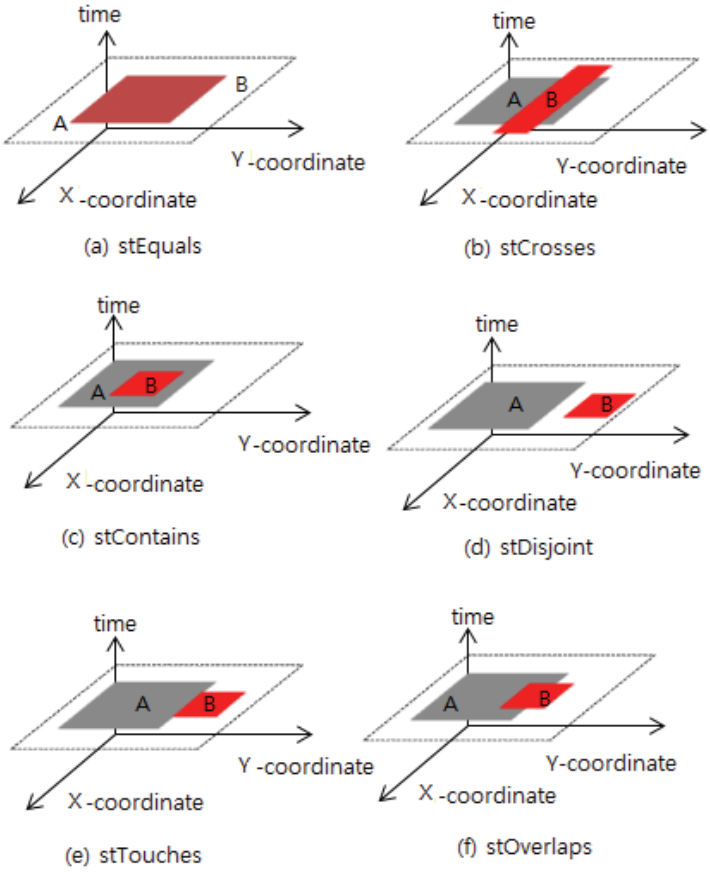

Fig. 6. Example of spatio-temporal relation operator.

As shown in Table 8, the time stamp/interval spatio-temporal analysis operator receives two spatio-temporal objects A and B as input values, performs space-time operation, and returns the spatio-temporal data type. For example, stIntersection of spatio-temporal analysis operators receives two spatio-temporal objects and returns spatio-temporal objects that intersect them as spatio-temporal data types.

Table 8. Time stamp/interval spatio-temporal analysis operator

Operator	Explain
stUnion(stGeometry A, stGeometry B) : stGeometry	Returns the union of spatio-temporal objects at a specific time or a specific time interval A and a specific time or specific time interval B
stDifference(stGeometry A, stGeometry B) : stGeometry	Returns the difference set of space-time objects at a specific time or a specific time interval A and a specific time or specific time interval B
stIntersection(stGeometry A, stGeometry B) : stGeometry	Returns the intersection of space-time objects at a specific time or a specific time interval A and a specific time or a specific time interval B
stBuffer(stGeometry A, Double L) : stGeometry	Returns a spatio-temporal object whose boundary of space-time objects in a specific time or specific time interval A has changed by the length L
stDistance(stGeometry A, stGeometry B) : Double	Returns the distance of a space-time object at a specific time or a specific time interval A and a specific time or a specific time interval B

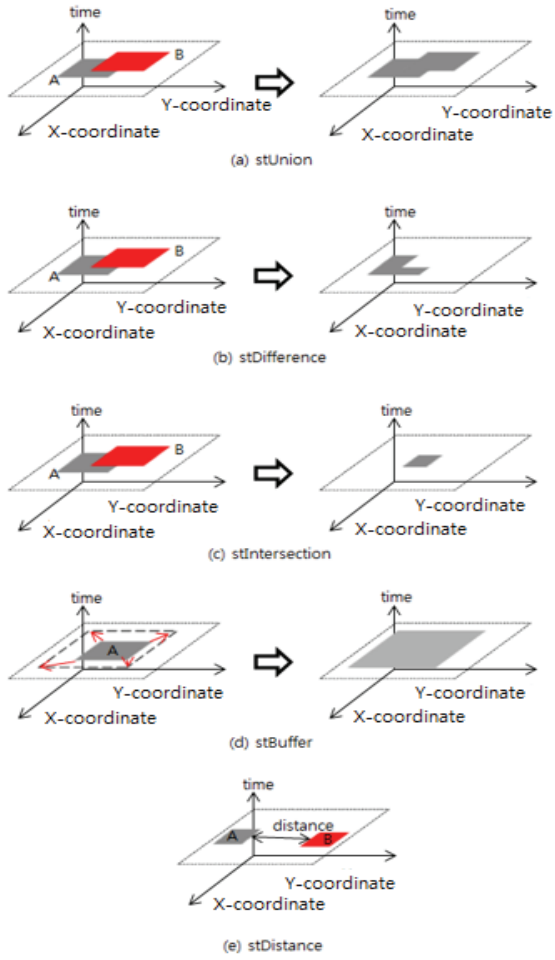

Fig. 7. Example of spatio-temporal analysis operator.

Fig. 7 shows a representative example of the space-time analysis operator.

The space-time trajectory operator processing module provides a space-time trajectory operator to support the space-time operation processing in the sensor node. The return value for this operator is true or false.

Table 9 shows the types of space-time trajectory operators.

Table 9. Spatio-temporal trajectory operator

Operator	Explain
stEnters(stGeometry A, stGeometry B) : Boolean	Returns whether the spatio-temporal object of a specific time or specific time interval B enters a specific time or a specific time interval A from the outside to the inside
stInsides(stGeometry A, stGeometry B) : Boolean	Returns whether a spatio-temporal object exists within a specific time or a specific time interval B within a specific time or a specific time interval A
stLeaves(stGeometry A, stGeometry B) : Boolean	Returns whether the spatio-temporal object of a specific time or specific time interval B goes out from inside to outside at a specific time or specific time interval A
stMeets(stGeometry A, stGeometry B) : Boolean	Returns whether a specific time or a specific time interval A can be reached only at the boundary of a space-time object at a specific time or a specific time interval B
stPasses(stGeomery A, stGeometry B) : Boolean	Returns whether the spatio-temporal object of a specific time or specific time interval B enters a specific time or a specific time interval A from the outside to the outside again and returns to the outside

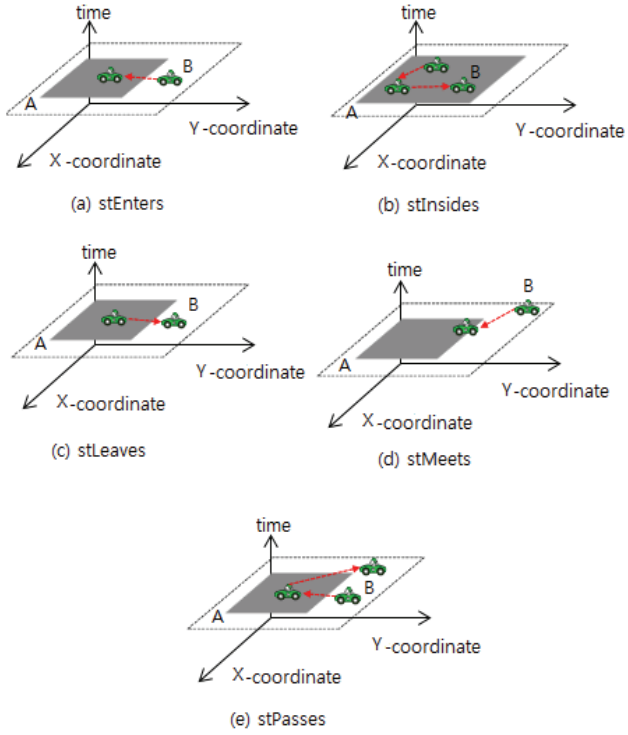

Fig. 8. Example of spatio-temporal trajectory operator.

As shown in Table 9, the space-time trajectory operator receives two space-time objects A and B as input values, performs time-space operations, and returns true or false. For example, the stEnter of the space-time trajectory operator receives two space-time objects and returns true or false as to whether a specific space-time object enters a specific space-time space.

Fig. 8 shows a representative example of space-time square operators.

4. Experiments and Results

In this paper, we implemented TinyOS 1.1.15 sensor operating system and TinyDB 1.1.3 sensor database system to implement STTinyDB. We used nesC 1.2.8 and g++ 3.4.3 provided by TinyOS. The management tool used Java 1.4 on Windows XP.

A hypothetical scenario for ecosystem monitoring services is to collect sensors and temperature and time and space information of migratory birds with a unique number of migratory birds by attaching sensors to 100 migratory birds to prevent the spread of avian influenza by migratory birds.

Table 10 shows the query used in the hypothetical scenario.

Table 10. Example of query

Query type	Example
Query 1	SELECT nodeid, temp, stLoc FROM sensors WHERE stEnters(stMultiPolygon(stPolygon(tPeriod(2015/05/01 00:00:00, 2015/05/01 23:59:59), 0 600, 0 900, 400 900, 400 600, 0 600), stPolygon(tPeriod(2015/05/01 00:00:00, 2015/05/01 23:59:59), 50 0, 500 300, 900 300, 900 0, 500 0)), stLoc) AND temp > 40 SAMPLE PERIOD 1024
Query 2	SELECT nodeid, temp, stLoc FROM sensors WHERE stLeaves(stMultiPolygon(stPolygon(tPeriod(2015/05/01 00:00:00, 2015/05/01 23:59:59), 0 600, 0 900, 400 900, 400 600, 0 600), stPolygon(tPeriod(2015/05/01 00:00:00, 2015/05/01 23:59:59), 50 0, 500 300, 900 300, 900 0, 500 0)), stLoc) AND temp > 40 SAMPLE PERIOD 1024

As shown in Table 10, Q1 returns body temperature and multi-dimensional time-space information of migratory birds that have body temperature exceeding 40 ° C in migratory birds entering Nakdong habitat or Yalu habitat for one day. Qi 2 returns body temperature and multi-dimensional space-time information of migratory birds that have a body temperature exceeding 40 ° C during the day, out of the Nakdong River habitat or Yalu river habitat.

Fig. 9 shows the result screen for query 1.

Fig. 9 shows the body temperature and time-space information of migratory birds satisfying Question 1. Among the unique numbers of migratory birds, 78, 49, and 12 are the Nakdong River habitat (stPolygon(tPeriod(2015/05/01 00:00:00, 2015/05/01 23:59:59), 0 600, 0 900, 400 900, 400 600, 0 600) or the Ampholyte habitat (stPolygon(tPeriod(2015/05/01 00:00:00, 2015/05/01 23:59:59), 50 0, 500 300, 900 300, 500 0)) among the migratory birds.

Fig. 10 shows the result screen for Query 2.

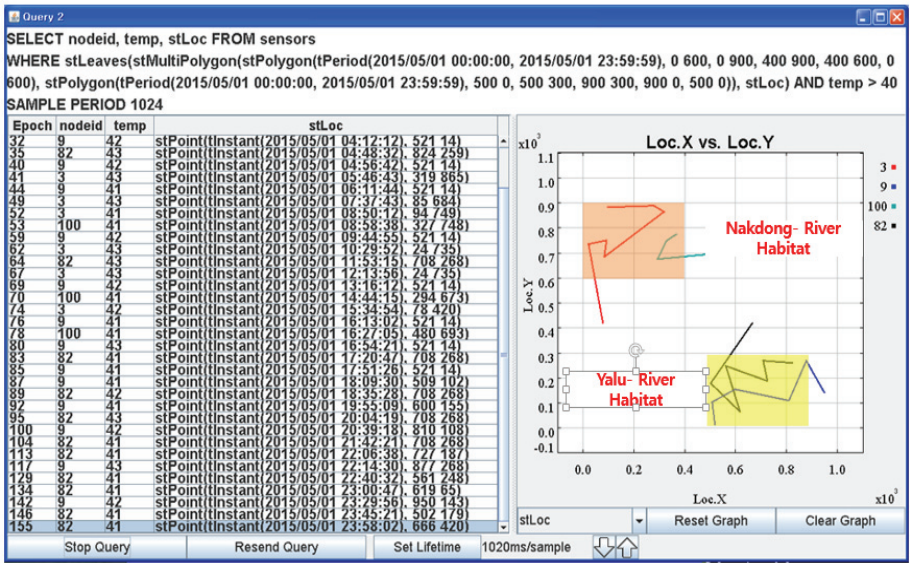

Fig. 9. Screen of Query 1.

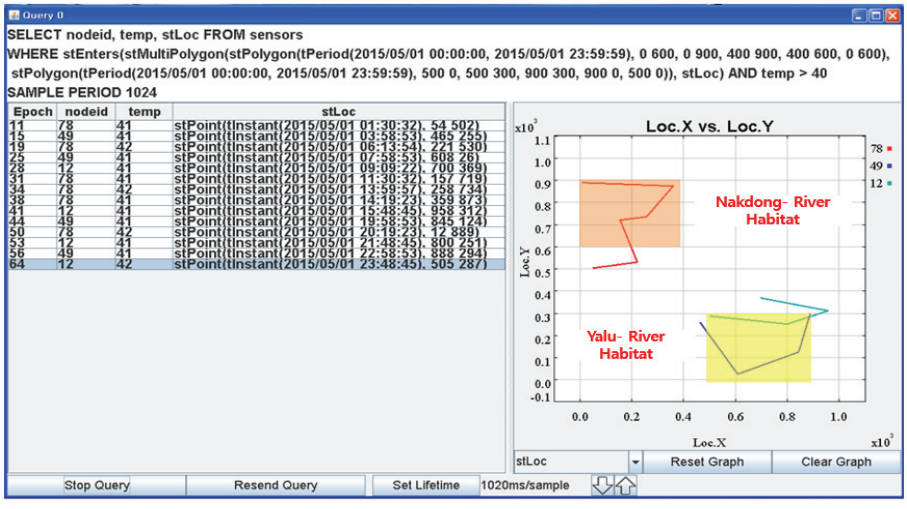

Fig. 10. Screen of Query 2.

Fig. 10 shows the body temperature and time-space information of migratory birds satisfying Question 2. Among the unique numbers of migratory birds, 3, 9, 100, and 82 are the stigma of the Nakdong River (tPeriod(2015/05/01 00:00:00, 2015/05/01 23:59:59), 0 600, 0 900, 400 900, 400 600, 0 600)) or the Yalu River habitat (stPolygon(tPeriod (2015/05/01 00:00:00, 2015/05/01 23:59:59), 50 0, 500 300, 900 300, 900 0, 500 0)) among the migratory birds outside the body temperature of 40°C can be seen.

By applying STTinyDB developed in this paper to ecosystem monitoring scenarios, it is shown that this system can be useful for applications requiring multi - dimensional sensor data query processing in sensor networks.

In order to evaluate the performance of STTinyDB developed in this paper, we compared performance by setting the number of sensor nodes, number of queries, data generation period, and filtering conditions based on the implementation environment described in the system implementation. Table 11 shows the parameters set for performance evaluation.

Table 11. Parameter for performance evaluation

Parameter	Value
Number of sensor node	100, 200, 400, 600
Number of query	20, 40, 60, 80
Period of data generation	128 ms
Filtering condition	Period of filtering 1024 ms
Query	
TinyDB	SELECT nodeid, time, x, y FROM sensors WHERE time >= "2015/04/01 09:00:00" AND time <= "2015/04/01 12:00:00" AND x >= 0 AND x <= 400 AND y >= 0 AND y <= 400
Spatial TinyDB	SELECT nodeid, time, loc FROM sensors WHERE time >= "2015/04/01 09:00:00" AND time <= "2015/04/01 12:00:00" AND contains(polygon(0 0, 0 400, 400 400, 400 0, 0 0), loc)
STTinyDB	SELECT nodeid, stLoc FROM sensors WHERE stContains(stPolygon(tPeriod(2015/04/01 09:00:00, 2015/04/01 12:00:00), 0 0, 0 400, 400 400, 400 0, 0 0), stLoc)

In this paper, comparison experiments of existing system and execution time and memory usage were conducted. First, when filtering is not applied, proven that performance is superior because spatio-temporal data type, space-time operator, memory sharing technology was applied. Then, when filtering was applied, it proved that the query processing performance is excellent while maintaining the allowable error range of the query processing result.

Performance evaluation was performed by querying each sensor node for one multi-dimensional spatio-temporal query, using data filtering and no filtering, and comparing the average time returned. The number of sensor nodes is increased from 100 to 600, and the performance improvement due to the increase of sensor nodes is examined. Fig. 11 shows the result of measuring the execution time according to the number of sensor nodes.

As shown in Fig. 11(a), the proposed STTinyDB improves performance by 14% on average and 7% on average compared to spatial TinyDB when filtering is not applied. As shown in Fig. 11(b), STTinyDB improves performance by 28% on average and 14% on average compared to spatial TinyDB when filtering. The reason for this performance evaluation result is that the multi-dimensional space type, operator, and filtering processing function of STTinyDB process the multi-dimensional space-time queries more quickly.

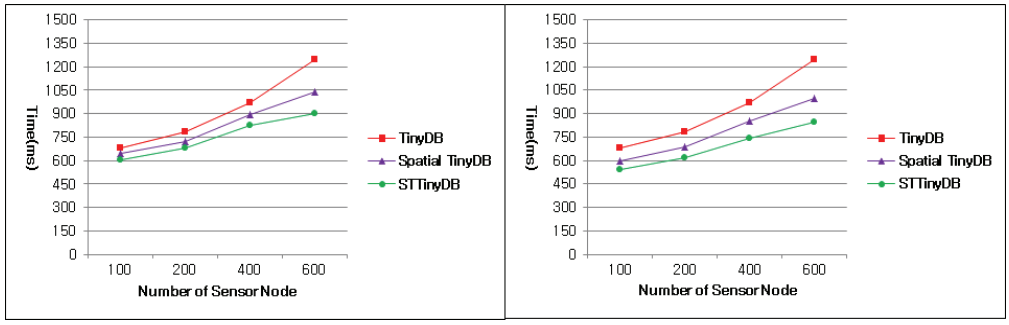

Fig. 11. Execution time by sensor node number. (a) Used filtering and (b) non-used filtering.

In other words, TinyDB treats the time, x, and y attributes in the where clause, while spatial TinyDB treats the time attribute separately in the where clause, whereas STTinyDB treats it as one operator at a time fast. The memory usage evaluation is based on how much memory is shared among multiple queries when one multi-dimensional spatio-temporal query is processed by one sensor node, whether data filtering is used or not. Respectively. As the number of multi-dimensional space-time queries increases from 20 to 80, the efficiency of memory usage due to the increase of multi-dimensional space-time queries is examined. Fig. 12 shows the result of measuring the memory usage according to the number of queries.

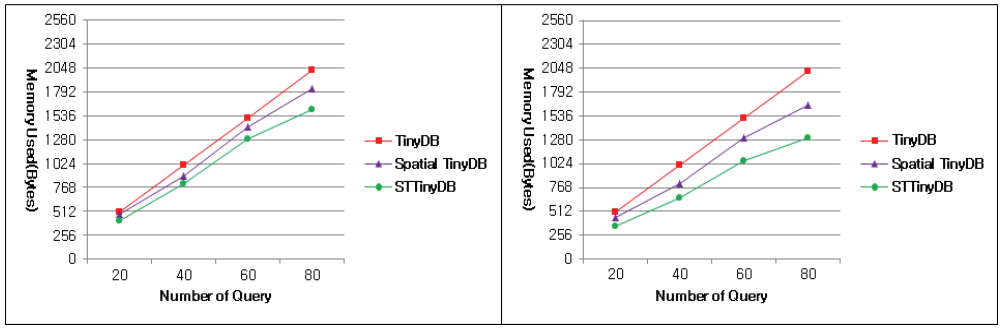

Fig. 12. Memory use rate by query number. (a) Used filtering and (b) non-used filtering.

As shown in Fig. 12(a), when the filtering is not applied, the STTinyDB shows that the average memory usage is reduced by 11% compared to TinyDB and 5% on average compared to spatial TinyDB. And, as shown in Fig. 12(b), STTinyDB shows an average of 24% less memory usage and 10% smaller memory usage than spatial TinyDB when filtering. The reason for this performance evaluation result is that because the memory sharing function of STTinyDB, several queries share memory.

5. Conclusion

Recently, with the development of IoT technology, various sensor data collection technology and wireless communication technology have been rapidly developed, so that interest and research on sensor network related technology are increasing.

A variety of spatial query processing systems have been studied for efficient query processing of two-dimensional spatial sensor data in sensor networks. However, existing spatial query processing systems do not support multi-dimensional sensor data because they do not support data types and operators for multi-dimensional sensor data processing.

Therefore, in order to solve this problem, we developed a multi-dimensional space-time query processing system by extending "Simple Features Specification for SQL" among OGC standard specification. The system provides multi-dimensional data types and space-time relation/analysis operators, and also provides trajectory operators to handle the trajectory of moving sensors. In addition, the memory sharing function is provided to reduce the system load caused by the data stream by sharing necessary data among various multi-dimensional spatio-temporal queries used in the present system. In case of inputting a lot of input data streams that cannot be processed, it provides a function to reduce the amount of input data stream as much as possible and to prevent overload from occurring.

Finally, by applying STTinyDB developed in this paper to ecosystem monitoring scenarios, it is shown that this system can be useful for many applications requiring multi-dimensional spatio-temporal query processing in sensor networks. STTinyDB is faster than STTinyDB, TinyDB and spatial TinyDB because of its multi-dimensional space-time operator and filtering processing capability. It is faster than other two systems to process queries and was used effectively. Overall, the STTinyDB proved to be superior to the other two systems.

In this paper, we did not consider multi-dimensional index method for searching multi-dimensional sensor data with time. Therefore, it is necessary to study the application of multi-dimensional indices for fast query processing of multi-dimensional sensor data.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2017R1A2B4011243).

References

[1] A. Raza, "Working with spatio-temporal data type," in *Proceeding of the 22nd Congress of the International Society for Photogrammetry and Remote Sensing*, Melbourne, Australia, 2012, pp. 5-10.

[2] A. D'Ulizia, F. Ferri, and P. Grifoni, "Moving GeoPQL: a pictorial language towards spatio-temporal queries," *GeoInformatica*, vol. 16, no. 2, pp. 357-389, 2012.

[3] International Organization for Standardization, *Data Elements and Interchange Formats–Information Interchange–Representation of Dates and Times*, ISO 8601:2004, 2004.

[4] S. H. Kim, D. H. Kim, and H. D. Park, "A tracking service of animal situation using RFID, GPS, and sensor," *Journal of the Institute of Internet, Broadcasting and Communication*, vol. 9, no. 5, pp. 79-84, 2009.

[5] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: an acquisitional query processing system for sensor networks," *ACM Transactions on Database Systems*, vol. 30, no. 1, pp. 122-173, 2005.

[6] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, et al, "TinyOS: an operating system for wireless sensor networks," in *Ambient Intelligence*. New York, NY: Springer, 2005, pp. 115-148.

[7] Open Geospatio Consortium, *OpenGIS Implementation Specification for Geographic Information - Simple Feature Access - Part 1: Common Architecture (version 1.2.1)*, 2011.

[8] Open Geospatio Consortium, *OpenGIS Implementation Standard for Geographic Information -Simple Feature Access - Part 2: SQL Option (version 1.2.1)*, 2011.

[9] P. D. Felice, M. Ianni, and L. Pomante, "A spatial extension of TinyDB for wireless sensor networks," in *Proceeding of IEEE Symposium on Computers and Communications*, Marrakech, Morocco, 2008, pp. 1076-1082.

[10] D. O. Kim, L. Liu, I. S. Shin, J. J. Kim, and K. J. Han, "Spatial TinyDB: a spatial sensor database system for the USN environment," *International Journal of Distributed Sensor Networks*, vol. 9, no. 8, article no. 512368, 2013.

[11] N. M. Laxaman, M. D. J. S. Goonathillake, and K. D. Zoysa, "Tikiridb: shared wireless sensor network database for multi-user data access," in *Proceeding of Conference on Computer Society of Sri Lanka*, Colombo, Sri Lanka, 2010.

Jeong-Joon Kim http://orcid.org/0000-0002-0125-1907

He received his B.S. and M.S. in Computer Science at Konkuk University in 2003 and 2005, respectively. In 2010, he received his Ph.D. in at Konkuk University. He is currently a professor at the department of Computer Science at Korea Polytechnic University. His research interests include Database Systems, Big Data, Semantic Web, Geographic Information Systems (GIS) and Ubiquitous Sensor Network (USN), etc.