New Join Operator Definitions for Sensor Network Databases*

Seungjae Lee, Changhwa Kim, Sangkyung Kim
Department of Computer Science and Engineering
Kangnung National University
Jibyun-dong, Gangnung-si, Gangwon-do
KOREA

Abstract: - Recently, researches on relational database approaches to sensor networks are being tried. There occur, however, some problems in applying directly the traditional relational database concepts into a sensor network in that every database operation is performed only on the real existing data which are tuples in database relations. The reason is because in a sensor network viewpoint situations under which some operations should be performed on non-existing data may occur frequently. For instance, let us assume a sensor network that two different classes of nodes are randomly scattered in the same area. We cannot get join results to know the relationship between two different classes of sensing values because there might be no nodes at a exact same location. For a solution about the above described problem we propose in this paper new join operators. This new join operators can provide more effective data management and standard interfaces to application programs in sensor networks.

Key-Words: - Database, Sensor network, Range join, Balanced range join, Imbalanced range join

1 Introduction

As implementation area of sensor networks is wider, researches on relational database approaches to sensor networks are being tried to process and manage tremendous sensing data which is gathered [1], [2], [3]. However there might be some problems in applying directly the traditional relational database concepts into a sensor network.

In the case of an environment monitoring sensor network, sensor nodes are deployed and gather data in a broad area. Because each sensor node sends data which is sensed at its location to the base station, the base station is limited to get data at the location at which each sensor node is.

Let us assume a sensor network that is consisted of two different classes of sensor nodes, i.e. temperature sensor nodes and humidity sensor nodes, and of which nodes are scattered randomly. In this situation, how can we pose a query to know the relationship between temperature and humidity according to a location? Though a join operation with coordinates as join attributes seems to be a good solution, there is a critical problem. The reason is that the traditional joins only allows exact joining that occurs only when all join attribute values are exactly same. There might be no or few temperature and humidity sensor pairs of which locations are exactly same, therefore, no or few tuples can be participated with the join operation.

For a solution about the above described problems we propose in this paper the new range join operators. It allows joining even though the join attribute values are not same in some range.

Applying a relational database with the range join operators, we can pose a join query more freely. It enables application code simpler and can provide standard application interfaces.

The rest of this paper is structured as follows. In Section 2 we present a brief overview of the related works and Section 3 describes the range join operators in detail. Finally, Section 4 concludes and finalizes this paper.

2 Related Works

In a sensor network, database concepts, especially relational database, are used to manage tremendous data effectively. A sensor network can be regarded as a database called to the sensornet database. A tuple, in a sensornet database, is configured with a sensing value, a

* This research was supported by the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Assessment)(IITA-2006-C1090-0603-0044)

sensing time, a node's location, etc. A set of sensing values gathered from a class of sensor nodes is treated as an attribute and a set of tuples created by a class of sensor nodes is treated as a relation[3].

A sensornet database has some different features from a traditional database and researches to overcome those differences are going on, i.e., in-network processing of sensor data, energy constraint, etc. Current researches related to database operators in a sensornet database can be classified with two categories.

First is researches on aggregation operators reducing energy consumption of a whole network and each sensor node. When a aggregation query, like MIN, MAX, AVERAGE, etc., is executed in sensornet database, it requests sensor data not to all sensor nodes but to a part sensor nodes or reduces the number of requests and approximates a result. It enables to reduce communication energy and extend the lifetime of a whole sensor network [4], [5], [6], [7], [8], [9].

Second, query processing for stream data is another important one. In a traditional database, an operand relation is always static and the result is always static, too. In contrast to this, tuples are created by sensors periodically or not and the relation tends to be dynamic in a sensornet database. This means that a relation in a sensornet database is a non-blocking relation and a query result for such relations is also non-blocking. Researches for operations on non-blocking relations are also proceeding [10], [11], [12], [13].

3 Range Joins

The range joins are identical to the natural join except three aspects. The first is that the domains of each join attribute in the range join must be subsets of real number. The second is that the join attribute set of the range join is allowed to be a subset of the intersection of operand relations and defined by a user. The third is that the range join allows to select tuples of which are combined with different join attribute values form the Cartesian product. The natural join select tuples of which join attribute values are exactly same from the Cartesian product of operand relations. But the range join allows to select tuples of which join attribute values are not exactly same. It can be used to join relations that have tuples of which join attribute values are not exactly same but in an allowable range.

Let us assume that there is a sensor network which consists of temperature sensor nodes and humidity sensor nodes. Each sensor node has only one sensor and is deployed randomly in two dimensional area. From a database point of view, we can assume that there are two relations which are *temp*(*X*, *Y*, *T*) and *hum*(*X*, *Y*, *H*). In this situation, if a join is performed to know the relationship between temperature and humidity according to the locations, the result might be an empty set or few tuples and we cannot achieve the proper result. The reason is that there the probability that a temperature sensor node and a humidity sensor node are at exactly same location is very small. Though we can solve the above problem with the fundamental operations, it might be too lengthy to express.

For a solution about the above problem, we propose new operators called to range join operator. It can be classified into two. One is the balanced range join operator and the other is the imbalanced range join operators. These range join operators select tuples of which join attribute values are similar within a given range from the Cartesian product of relations. Where the domains of each join attribute are subsets of real number, a natural join can be treated as a special case of range join.

Before introducing the range join operators in detail, we define a terminology 'tuple vector' needed to explain it.

Definition.
Where there is a relation R of which schema is A and an attribute set $A' = \{A_1', A_2', \cdots, A_{a'}'\}$ such as $A \supset A'$ and each attribute domain of A' is a subset of real number, a set of which elements are sequences of all possible attribute values $(a_1', a_2', \cdots, a_{a'}')$ is R^n. Therefore, a sequence of t's attribute values which is included within A', $\boldsymbol{t(A')} = (t.a_1', t.a_2', \cdots, t.a_{a'}')$, is **the tuple vector of tuple** t **for the attribute set** A'.

3.1 Balanced Range Join Operator

The balanced range join operator selects tuples of which the difference of join attribute values is under a given value and its symbol is $\bowtie_\rho$ similar to the natural join operator. At first, we discuss the balanced range join for two relations and then over three relations.

i) The balanced range join for two relations

Let us consider the balanced range join for two relations. Where attribute sets of two relation r and s are R and S each and the balanced range join attribute set is $C = \{C_1, C_2, \cdots, C_c\}$ that satisfies $C \subset R$ and $C \subset S$, the balanced range join for r and s is represented as (1). Each attribute domain in C must be a subset of real number and ρ is the range constant that denotes the maximum difference of join attribute values of tuples to be selected from $r \times s$. The schema of the balanced range join is $R \cup S$ like the natural join.

$$r \bowtie_{\rho} s(C) \text{ or } r \bowtie_{\rho} s(C_1, C_2, \cdots, C_c) \tag{1}$$

The process of the balanced range join has three steps. First, select valid tuples from $r \times s$ and call it to the relation v. For a tuple t_r in r and a tuple t_s in s, if t_v is a tuple combined with t_r and t_s satisfying (2), t_v is a valid tuple. This means that a valid tuple is a combined tuple with two tuple that the distance between its vectors for C is under ρ.

$$|t_r(C) - t_s(C)| \le \rho \tag{2}$$

Second, add new attribute set C to v and assign each $C_i (1 \le i \le c)$ with the mean value of $r.C_i$ and $s.C_i$. The balanced range join attribute values of t_v are like (3).

$$t_v.c_i = \frac{t_r.c_i + t_s.c_i}{2}, \ (1 \le i \le c) \tag{3}$$

Finally, remove attribute sets C of r and s from v and finalize the range join operation.

If $\rho = 0$ and $R \cap S = C$, the result of the balanced range join is identical with that of the natural join.

For an example, let us assume that there is a sensor network that consists of six temperature sensors and six humidity sensors in two dimensional area. The data generated from a temperature sensor are the coordinate of the sensor and temperature at the coordinate. All data from all temperature sensor construct the relation *temp*(*X,Y,T*). Similarly, data generated from a humidity sensor are the coordinate of the sensor and humidity at the coordinate. All data from all humidity sensor construct the relation *hum*(*X,Y,H*). *X* and *Y* is the coordinate of the sensor node, *T* is temperature and *H* is humidity. Tbl. 1 is an example for these two relation.

Now, if we perform the natural join to know the relationship *temp* with *hum* according to the location, the result is an empty set and is not a desired result because there is no tuple pair which satisfies *temp.X=hum.X* and *temp.Y=hum.Y*. In this case, the range join can be a good solution if joining one sensor node with nearby different kind of sensor nodes within an admittable range is permitted. In this case, we allow that two nodes of which distance is under 10 are permitted to join. Therefore, the range constant ρ is 10 and the balanced range join equation is as (4)

$$temp \bowtie_{10} hum(X, Y) \tag{4}$$

Tbl. 2 shows the distances between each temperature sensor nodes and humidity sensor node and there are 6 pairs of sensor nodes, grayed values, of which distance is under 10.

The schema of the result relation is (*X, Y, T, H*) and the result of (4) is shown at Tbl. 3. Attribute values of *X* and *Y* is calculated by (3).

ii) The balanced range join for over three relations

Now, let us consider the balanced range join for n relations, i.e. $r_1, r_2, \cdots, r_n$ of which schema is $R_1, R_2, \cdots, R_n$ each. The result of

Table 2. The distances between temperature sensors and humidity sensors.

	TS1	TS2	TS3	TS4	TS5	TS6
HS1	34	20	23	49	68	65
HS2	4	27	30	47	30	39
HS3	43	28	23	5	59	36
HS4	26	4	1	28	54	44
HS5	36	58	60	66	9	41
HS6	41	30	26	5	53	28

Table 1. An example of the *temp* and the *hum* relation.

temp

Node ID	*X*	*Y*	*T*
TS1	62	48	24
TS2	54	70	23
TS3	56	74	25
TS4	78	90	23
TS5	93	34	26
TS6	99	65	22

hum

Node ID	*X*	*Y*	*H*
HS1	34	68	70
HS2	65	45	60
HS3	73	90	77
HS4	56	73	89
HS5	90	25	56
HS6	80	85	86

Table 3. The result of $temp \bowtie_{10} hum(X, Y)$

Node pair	X	Y	T	H
(TS1, HS2)	64	47	24	60
(TS2, HS4)	55	72	23	89
(TS3, HS4)	56	74	25	89
(TS4, HS3)	76	90	23	77
(TS4, HS6)	79	88	23	86
(TS5, HS5)	92	30	26	56

the natural join is always same regardless of the operation order like $(r_1 \bowtie r_2) \bowtie r_3 = r_1 \bowtie (r_2 \bowtie r_3)$ but not in the balanced range join operator. $(r_1 \bowtie_\rho r_2(C)) \bowtie_\rho r_3(C) = r_1 \bowtie_\rho (r_2 \bowtie_\rho r_3(C))(C)$ is not satisfied, generally. A series of the range join operator must not be treated as multiple join operations but as just one operation. To avoid confusion with a series of the natural join operators, ρ is put with the first join operator symbol and is represented as (5). We must be careful that (5) is not the combination of the balanced range join and the natural joins but just one balanced range join.

$$r_1 \bowtie_\rho r_2 \bowtie \cdots \bowtie r_n(C) \text{ or}$$
$$r_1 \bowtie_\rho r_2 \bowtie \cdots \bowtie r_n(C_1, C_2, \cdots, C_n) \qquad (5)$$

The balanced range join for over three relations is similar with that for the relations. Because (5) is just one operation, it selects valid tuples from the full Cartesian product $r_1 \times r_2 \times \cdots \times r_n$ and we denote it to v.

Where t_{r_1}, t_{r_2}, $\cdots$, t_{r_n} are tuples in r_1, r_2, $\cdots$, r_n each and t_v is a combined tuple with t_{r_1}, t_{r_2}, $\cdots$, t_{r_n} in v, t_v is a valid tuple if (6) is satisfied. This means that Euclidean distances between all tuple vectors for the join attribute C, for those tuples of which a valid tuple consist, are under ρ.

$$|t_{r_i}(C) - t_{r_j}(C)| \le \rho, \ \forall i,j (1 \le i,j \le n) \qquad (6)$$

After selecting valid tuples from v, add new attribute set C to v and assign each C_i $(1 \le i \le c)$ with a proper value. We use the average values of the join attributes to be C_i as the balanced range join for two relations. Therefore, the join attribute values of t_v are the averages of attribute values of tuples which combine t_v and are calculated by (7).

$$t_v \cdot c_i = \frac{1}{n} \sum_{j=1}^{n} t_{r_j} \cdot c_i, \ 1 \le i \le c \qquad (7)$$

Where $\rho = 0$ and $R_1 \cap R_2 \cap \cdots R_n = C$ in the balanced range join for multiple relations, it also can be treated as the natural join like $r_1 \bowtie r_2 \bowtie \cdots \bowtie r_n$ and the result is identical with that of the natural join.

For an example, let us consider the balanced range join for three relations. We add the new relation *pres*(*X*, *Y*, *P*) which represents air pressure at each location. An example for *pres* is shown in Tbl. 4.

Now, we want to know the relationship between *temp*, *hum* and *pres* according to the locations using the balanced range join. If we permit to join tuples of sensor nodes within the diameter 10, the balanced range join operation is as (8).

$$temp \bowtie_{10} hum \bowtie pres(X, Y) \qquad (8)$$

Tbl. 5 shows the distances between pressure sensor nodes with temperature and humidity

Table 4. An example of the *pres* relation.

pres

Node ID	*X*	*Y*	*T*
PS1	54	57	1014
PS2	62	43	1020
PS3	58	75	1012
PS4	45	96	1008
PS5	80	83	1016
PS6	95	30	1017

Table 5. The distances pressure sensor nodes with other sensor nodes

(a) The distances temperature sensor nodes with pressure sensor nodes

	TS1	TS2	TS3	TS4	TS5	TS6
PS1	12	13	17	41	45	46
PS2	5	28	32	50	32	43
PS3	27	6	2	25	54	42
PS4	51	28	25	34	78	62
PS5	39	29	26	7	51	26
PS6	38	57	59	62	4	35

(b) The distances humidity sensor nodes with pressure sensor nodes

	PS1	PS2	PS3	PS4	PS5	PS6
HS1	23	38	25	30	48	72
HS2	16	4	31	55	41	34
HS3	38	48	21	29	10	64
HS4	16	31	3	25	26	58
HS5	48	33	59	84	59	7
HS6	38	46	24	37	2	57

Table 6. The result of $temp \bowtie_{10} hum \bowtie pres(X, Y)$

Node pair	X	Y	T	H	T
(TS1, HS2, PS2)	63	45	24	60	1020
(TS2, HS4, PS3)	56	73	54	89	1012
(TS4, HS6, PS5)	79	86	78	86	1016
(TS5, HS5, PS6)	93	30	93	56	1017

sensor nodes. In Tbl. 2 and Tbl. 5, we can find four pairs of node which is in the diameter 10. These are (TS1, HS2, PS2), (TS2, HS4, PS3), (TS4, HS6, PS5) and (TS5, HS5, PS6).

Therefore, the schema of the result relation is (*X*, *Y*, *T*, *H*, *P*) and the result of (8) is shown at Tbl. 6. Attribute values of X and Y is calculated by (7).

3.2 Imbalanced Range Join Operator

The imbalanced range join is divided into the left range join and the right range join. It does not calculate join attribute values of the result relation but assigns the join attribute values of the left or right most relation to those of the result relation.

The processes of the imbalanced range join are like below. First, select valid tuples from the Cartesian product of operand relations. In the case of the left range join operator, it assigns the join attribute values of the left most relation to those of the result relation and is denoted by $⟕$ like (9). The join attribute values of the result relation are those of r in this case.

$$r ⟕_{\rho} s(C) \text{ or } r ⟕_{\rho} s(C_1, C_2, \cdots, C_c) \qquad (9)$$

For right range join, $⟖$ is used for expression as (0) and the join attribute values of the result relation are those of s in this case.

$$r ⟖_{\rho} s(C) \text{ or } r ⟖_{\rho} s(C_1, C_2, \cdots, C_c) \qquad (10)$$

We call the relation r at (9) and s at (10) to the anchor relation of the imbalanced range join because the attribute values of only the relation are unchanged.

In fact, the imbalanced range join operators are additional operators which are expressed by using the fundamental operators and we express (9) and (10) as (11) and (12) where C in R is R_C and C in S is S_C.

$$r ⟕_{\rho} s(C) = \Pi_{(R \cup S) - C, R_C}(\sigma_{d \le \rho}(r \times s)) \qquad (11)$$

$$r ⟖_{\rho} s(C) = \Pi_{(R \cup S) - C, S_C}(\sigma_{d \le \rho}(r \times s)) \qquad (12)$$

$$d = \sqrt{\sum_{i=1}^{c} (r.C_1 - s.C_i)^2}$$

For an example of the left range join, the result of $temp ⟕_{10} hum(X, Y)$ for Tbl. 1 is as Tbl. 7.

In the case of the left range join for n relations, i.e. r_1, r_2, $\cdots$, r_n, symbol $⟕$ appears just once at first as (13) and r_1, the left most relation in the equation, is the anchor relation. In right join operator, $⟖$ appears just once at last as (14) and r_n, the right most relation in the equation, is the anchor relation.

$$\begin{gathered} r_1 ⟕_{\rho} r_2 \bowtie \cdots \bowtie r_n(C) \text{ or} \\ r_1 ⟕_{\rho} r_2 \bowtie \cdots \bowtie r_n(C_1, C_2, \cdots, C_c) \end{gathered} \qquad (13)$$

$$\begin{gathered} r_1 \bowtie_{\rho} r_2 \bowtie \cdots ⟖ r_n(C) \text{ or} \\ r_1 \bowtie_{\rho} r_2 \bowtie \cdots ⟖ r_n(C_1, C_2, \cdots, C_c) \end{gathered} \qquad (14)$$

We can express (13) as (15) by the fundamental operators where R_{1C} is C in r_1

$$\begin{aligned} & r_1 ⟕_{\rho} r_2 \bowtie \cdots \bowtie r_n(C) = \\ & \Pi_{(R_1 \cup R_2 \cup \cdots \cup R_n) - C, R_{1C}} \\ & (\sigma_{(d_2 \le \rho) \wedge (d_3 \le \rho) \wedge \cdots \wedge (d_n \le \rho)}(r_1 \times r_2 \times \cdots \times r_n)) \end{aligned} \qquad (15)$$

$$d_i = \sqrt{\sum_{j=1}^{c} (r_1.C_j - r_i.C_j)^2}, (2 \le i \le n)$$

In the same way, (14) is expressed as (16) where R_{nC} is C in r_n.

$$\begin{aligned} & r_1 \bowtie_{\rho} r_2 \bowtie \cdots ⟖ r_n(C) = \\ & \Pi_{(R_1 \cup R_2 \cup \cdots \cup R_n) - C, R_{nC}} \\ & (\sigma_{(d_1 \le \rho) \wedge (d_2 \le \rho) \wedge \cdots \wedge (d_{n-1} \le \rho)}(r_1 \times r_2 \times \cdots \times r_n)) \end{aligned} \qquad (16)$$

$$d_i = \sqrt{\sum_{j=1}^{c} (r_1.C_j - r_i.C_j)^2}, (1 \le i \le n-1)$$

In contrast to the balanced range join, the

Table 7. The result of $temp ⟕_{10} hum(X, Y)$

Node pair	X	Y	T	H
(TS1, HS2)	62	48	24	60
(TS2, HS4)	54	70	23	89
(TS3, HS4)	56	74	25	89
(TS4, HS3)	78	90	23	77
(TS4, HS6)	78	90	23	86
(TS5, HS5)	63	34	26	56

Table 8. The result of $temp ⟕_{10} hum \bowtie pres(X, Y)$

Node pair	X	Y	T	H	T
(TS1, HS2, PS2)	62	48	24	60	1020
(TS2, HS4, PS3)	54	70	23	89	1012
(TS3, HS4, PS3)	56	74	25	86	1012
(TS4, HS3, PS5)	78	90	23	86	1016
(TS4, HS6, PS5)	78	90	23	86	1016
(TS5, HS5, PS6)	93	34	26	56	1017

imbalanced range join does not use (6) but logical AND of d_i at (15) or (16). This means that if t_v satisfies (17) for all $t_{r_i}(1 \leq i \leq n)$ then t_v is selected as a result tuple where t_v is a tuple in the Cartesian product and combined with t_{r_1}, t_{r_2}, $\cdots$, t_{r_n} in r_1, r_2, $\cdots$, r_n each and $r_k(1 \leq k \leq n)$ is the anchor relation.

$$|t_{r_k}(C) - t_{r_i}(C)| \leq \rho, \ \forall\, i \ (1 \leq i \leq n) \qquad (17)$$

Where $\rho = 0$ and $R_1 \cap R_2 \cap \cdots R_n = C$ in the imbalanced range join, it also can be treated as the natural join like $r_1 \bowtie r_2 \bowtie \cdots \bowtie r_n$ and the result is identical with that of the natural join.

For an example of the left range join for three relations, the result of $temp \ltimes_{10} hum \ltimes pres(X, Y)$ for Tbl. 1 and Tbl. 4 is as Tbl. 8 and is different from Tbl. 6. In this cast, the relation *temp*, the left most relation, is the anchor relation.

4 Conclusion

Sensor nodes are deployed non-continuously in space and its locations are appeared as points. Therefore we can get sensing data at locations where sensor nodes are but cannot at elsewhere.

In this paper, we proposed the new join operators to performing join operation on similar join attribute values. Adapting this operator to a sensor network database, a user can simply pose a query that needs to join relations having similar join attribute values. This also can provide a standard interface to application programs and ensure more robust system.

The range join operators are very useful especially for the sensor networks that consist of several kinds of sensor nodes. It supports more flexible join operation including the natural join by enabling join operation on not only exact same join attribute values but also similar them.

As shown above, applying a database using the proposed join operator to a sensor network, more flexible join operation can be performed and a standard interface can be supplied to application programs from database level.

A query optimization method for a virtual selection operator and a topology adaptive estimate method are important research topics and those will be explored in the future.

References:

[1] Philippe Bonnet, Johannes Gehrke, and Graveen Seshadri, Towards sensor database systems, *In Mobile Data Management*, 2001, pp. 3-14

[2] M. Srivastava, R. Muntz, and M. Potkonjak.: Smart Kindergarten, Sensor-Based Wireless Networks for Smart Developmental Problem-Solving Environments, *In Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking*, 2001

[3] R. Govindan, J. Hellerstein, W. Hong, S. Madden, M. Franklin, and S. Shenker, The Sensor Network as a Database, *Technical Report 02-771, Computer Science Department, University of Southern California*, 2002

[4] J. M. Hellerstein, W. Hong, S. Madden, and K. Stanek, Beyond average: Toward sophisticated sensing with queries, In Information Processing in Sensor Networks: *2nd Intl. Workshop, Springer-Verlag, LNCS 2634*, 2003, pp.63-72.

[5] S. Madden, M.J. Franklin, J. Hellerstein, and W. Hong, Tag: a tiny aggregation service for ad-hoc sensor networks, *In Proc. of OSDI '02*, 2002

[6] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri, Medians and beyond: New aggregation techniques for sensor networks, *In Proc. of Sensys'04*, 2004

[7] Joseph M. Hellerstein, Peter J. Haas, and Helen J. Wang, Online Aggregation, *In Proc. ACM SIGMOD International Conference on Management of Data*, 1997

[8] T. Friedman and D. Towsley, Multicast Session Membership Size Estimation, *In Proc. of IEEE Infocom*, 1999

[9] W. Hou, G. Ozsoyoglu, and B. Taneja, Statistical estimators for relational algebra expressions, *In Proc. Seventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems(PODS)*, 1988, pp.276-287

[10] Johannes Gehrke, Samuel Madden, Query Processing in Sensor Networks, *IEEE CS and IEEE ComSoc*, 2004

[11] Annita N. Wilschut and Peter M. G. Apers, Dataflow query execution in a parallel main-memory environment, *Distributed and Parallel Databases, 1(1)*, 1993, pp.103-128

[12] Peter J. Hass and Joseph M. Hellerstein, Ripple Joins for Online Aggregation, *In Proc. ACM-SIGMOD International Conference on Management of Data*, 1999, pp.287-298

[13] S. Madden and M. Franklin, Fjording The Stream: An Architecture for Queries over Streaming Sensor Data, *In Proceedings of the 18th International Conference on Data Engineering, San Jose, CA*, 2002