Querying Fuzzy Spatiotemporal RDF Data Using R2RML Mappings

Luyi Bai
School of Computer and Communication Engineering Northeastern University (Qinhuangdao)
Qinhuangdao 066004, China
baily@neuq.edu.cn

Jiajia Lu
School of Computer and Communication Engineering Northeastern University (Qinhuangdao)
Qinhuangdao 066004, China
1139280822@qq.com

Shuangdi Wang
School of Computer and Communication Engineering Northeastern University (Qinhuangdao)
Qinhuangdao 066004, China
1227883666@qq.com

***Abstract*—With the development of Internet technology, information management has shown a spurt of growth. In real-life applications, information usually includes spatial, temporal, or spatiotemporal features. Spatiotemporal data has temporal and spatial attributes, and these attributes are often fuzzy. Due to the great significance of fuzzy spatiotemporal data management, how to query fuzzy spatiotemporal data efficiently and effectively has become an important research issue. In that case, this paper formally proposes a new implementation method, which is the query processing of fuzzy spatiotemporal data. In view of the advantages of the most advanced mapping method (R2RML) in data transformation, three algorithms are proposed to transform fuzzy spatiotemporal data from relational database into fuzzy RDF data based on R2RML. On this basis, according to the characteristics of fuzzy RDF data, we give three different fuzzy quantifiers (extreme fuzzy quantifier, range fuzzy quantifier, degree fuzzy quantifier) to represent fuzzy spatiotemporal RDF data. Since SPARQL plays an important role in querying RDF data, it is used for the query of fuzzy spatiotemporal RDF data. In addition, three kinds of fuzzy quantifiers are designed, and the experimental results show the superiority of this method by analyzing experiments in the aspects of the recall and precision.**

Keywords: Fuzzy Spatiotemporal Data, R2RML, RDF, SPARQL

I. ...INTRODUCTION

With the explosive growth of information, various kinds of data have appeared, including a large number of fuzzy spatiotemporal data. The fuzzy spatiotemporal data have temporal properties [2], spatial properties [3], and fuzzy properties. Therefore, how to process and query fuzzy spatiotemporal data is an urgent problem to be solved. In literature [4], Sozer et al. propose fuzzy spatiotemporal semantic research and various types of fuzzy spatiotemporal data on relational databases. Obviously, fuzzy geographic information objects and meteorological information objects can be stored in relational databases.

In practice, people begin to join the field of mapping from relational databases to RDF [5]. RDF (Resource description framework) [11] has been applied as a standard and specification for describing semantic resources. In the semantic Web, RDF can be used to determine the semantics of information resources, from which accurate information can be obtained. RDF has many other great features: simplicity, openness, ease of extension, ease of control, etc., and is increasingly popular with application developers. For these reasons, a number of good ways to map to RDF have been come up with.

Johannesson et al. [6] propose a way to transform a relational data model into a conceptual model by using a mapping technique, which is then continuously repaired by the users to generate the final RDF. To reduce the processes of processing transformations， Stojanovic et al. [7] attempt to use another mapping technique to construct lightweight RDF from a conceptual database model. Johannesson and Stojanovi are among those who have turned relational databases directly into RDF. However, there is only a slight change in the content or semantic meaning of the content. The lightweight ontology structure is very simple and the relationships between concepts are not complicated. In order to construct structures that represent complex relationships between concepts, Hert et al. [8] propose a mapping method for data from relational databases to RDF, and develop a corresponding prototype system D2R mapping based on this method. Although this approach is based on RDF, a specific mapping language still needs to be learned. Hence, there is easy to understand the mapping language in the popular mapping method, Petrini et al. [9] propose a mapping method based on query language, which completes the mapping of relational databases to RDF by querying relational databases using RDF query language to return RDF triples. This method is manual completely, requiring the user to specify not only a mapping table, but also the corresponding relationship of objects or property values to which the table or column is associated. Laborda et al. [10] improve the manual completely method proposed by Petrini et al. [9] and form a semi-automated mapping method. This approach starts by mapping automatically the relational database to expressions that the machine can understand and process. This expression is a non-semantic expression. The method then completes the representation of semantic RDF by constructing the RDF query statement manually and setting the format of the target RDF. This approach is a semi-automatic mapping approach that requires users to participate. For users who do not understand the RDF query language, the mapping task cannot be completed. Even if the user has a basic base of RDF query language, it can be very difficult to build a comprehensive and complete RDF query statement. Therefore, they make a deep study of the expression of RDF, and later they make a specific study of the expression of automated RDF automated fully RDF.

However, these mapping methods are not widely applicable to practice. To solve the problem, R2RML is proposed, and in 2012 it became a standard recommended by W3C for conversion, which is mainly used to transform relational data into RDF data. As the volume of RDF data continues to grow over time, it becomes extremely difficult to make accurate and efficient queries in large volumes of RDF data. In recent years, a better technique for querying RDF has been found finally, using SPARQL to query RDF data. And we've got a number of ways to query RDF data

978-1-7281-6932-3/20/$31.00 ©2020 IEEE

using SPARQL. However, the general SPARQL queries for RDF data can only provide some accurate queries, but in real-life users often need some fuzzy queries [1]. The formal structure of fuzzy SPARQL is given to meet the requirements of fuzziness in query process of most users.

Based on the characteristics of fuzzy spatiotemporal data, this paper uses R2RML to map relational databases to fuzzy RDF data, and extends SPARQL to query fuzzy spatiotemporal RDF data.

The main contributions of this paper are as follows:

- Fuzzy spatiotemporal data model is established.
- We propose three algorithms transforming fuzzy spatiotemporal data from relational database into RDF.
- Three kinds of fuzzy quantifiers are proposed to query fuzzy spatiotemporal RDF data based on SPARQL.

The following research ideas of this paper are as follows: In Section II, R2RML is used to complete the mapping of relational data to RDF, and three algorithms are proposed to complete the triplet mapping. Section III studies mainly the query of SPARQL to RDF data, based on which three fuzzy quantifiers are proposed to complete the fuzzy analysis. We show the experimental results in Section IV.

II.... R2RML FOR TRIPLE MAPPING

R2RML is a specification for mapping relational databases to RDF datasets and became the recommended standard for W3C in 2012. The working mechanism of R2RML consists of three elements: database input, R2RML mapping file, and R2RML processor.

There are three cases for entering a database. The first is a basic table of a relational database; the second is a basic view; and the third is a valid SQL query. They are collectively referred to as logical tables. Each logical table is mapped to an RDF triple by R2RML, and each row of data in the logical table can be converted into multiple RDF triples. The R2RML mapping file is a physical representation of the R2RML mapping rules, and the R2RML mapping itself is an RDF graph written in Turtle syntax in the R2RML vocabulary, which is called the R2RML mapping. The mapping language implementation of the R2RML-based RDB2RDF tool is named R2RML processor.

A. Triple mapping algorithm (DoTM)

The transformation of the relational database into RDF algorithm based on R2RML mainly includes three algorithms: The triple mapping algorithm (DoTM), the predicate object mapping algorithm (DoPOM), and the reference object algorithm (DoROM). The triple mapping algorithm is the core algorithm, which can implement most of the functions required for mapping. It can also call the predicate object mapping algorithm and the reference object algorithm to process the details of the triple mapping further. At the same time, they store and merge the resulting results into an ArrayList, and output the ArrayList to the RDF data set finally.

(1) It must have only one rr:logicalTable attribute. Its value is a logical table that specifies the result of the SQL query to be mapped to the triple.

(2) It must have only one subject mapping that specifies how to generate a subject for each row of the logical table. It can be specified in two ways:

Use the rr:subjectMap attribute, and its value must be a subject mapping;

Use the constant shortcut property rr:subject.

(3) It may have zero or more rr:predicateObjectMap attributes, and its value must be a predicate object mapping. By referencing the predicate object mapping algorithm (DoPOM), the reference object algorithm (DoROM) specifies the predicate mapping and object mapping. Together with the subject generated by the subject mapping, they can form one or more RDF triples for each row. Then, merge the RDF data into an ArrayList.

The following is the triple mapping algorithm 1:

Algorithm 1 (**DoTM**)

Input: Database connection, spatiotemporal relationship data
Output: ArrayList array

```
1.  Load the property c file,open the connection relational database
2.  Load the triple mapping file
3.  array←∅
4.  rr:logicalTable← logicalTable
5.  rows←executeQuery
6.  For each row ∈ rows
7.     rr：subjectMap←subject(row，subjectMap)
8.     rr：subject←subject(row, subjectMap)
9.     Array.add(rr：subject)
10.    rr：predicateObjectMaps←predicate and object
11.       For each rr: predicateObjectMap∈ rr：predicateObjectMaps
12.          array←array∪DoPOM
13.             For each RefObjectMap∈ rr：predicateObjectMap
14.                array←array∪DoROM
15.       End for
16.  End for
17.  Return array
```

The triple mapping algorithm first establishes a connection to the MySQL database, stores fuzzy spatiotemporal data according to the spatiotemporal data model, and then loads the property file. Open the relational database to which it is connected, and load the configured R2RML mapping file (lines 1-2). An ArrayList container is created and initialized to save the resulting RDF triples (line 3) each time. Next, get the rr:logicalTable attribute (generally the name of the table) from the logicalTable (line 4). Each row of the logical table is traversed, and subject mapping is performed before traversal of each row, mainly through rr:subjectMap and rr:subject to obtain the subject attribute value, which is the primary key value of the current row of the logical table generally (lines 5-8). Store the value in the ArrayList container (line 9). Then the predicate object mapping is performed and the DoPOM algorithm is called (shown in b below) to get the predicate mapping value and object mapping value stored in the ArrayList container (lines 10-12). If there is a reference object mapping, the DoROM algorithm (shown in c below) is called and the obtained object properties are put into the container. Finally, the RDF data set (lines 13-17) in the ArrayList container is uniformly returned.

B. Predicate object mapping algorithm (DoPOM)

The predicate object mapping algorithm mainly implements the mapping processing of the triplet predicate and the object. A predicate object mapping is a function that creates one or more predicate objects for each row of a

logical table. It is used in conjunction with subject mapping to generate RDF triples in a triplet graph. A predicate object mapping is represented by a resource that references the following other resources.

There are one or more predicate mappings. They can be specified in one of two ways: use the rr:predicateMap property, and the value of this property must be a predicate mapping; use the constant shortcut property rr:predicate.

There are one or more object maps or reference object maps. They can be specified in one of two ways: use the rr:objectMap property, and the value of this property must be an object mapping or a reference object map; use the constant shortcut property rr:object.

The following is the predicate object mapping algorithm 2:

Algorithm 2 (**DoPOM)**

Input: Database connection, spatiotemporal relationship data
Output: ArrayList array

```
1.    rows←executeQuery
2.    array←∅
3.    For each row ∈ rows
4.      For each rr：predicateObjectMap∈ rr：predicateObjectMaps
5.        For each predicate∈ rr：predicateObjectMap
6.          rr：predicateMap←predicate(row，predicate)
7.          rr：predicate←predicate(row，predicate)
8.          array.add(rr：predicate)
9.        End for
10.       For each object∈ rr：predicateObjectMap
11.         rr:objectMap←object(row，object)
12.          rr：object←object(row，object)
13.          array.add(rr：object)
14.         For each RefObjectMap∈ rr：predicateObjectMap
15.               array←array∪DoROM
16.         End for
17.       End for
18.     End for
19.   End for
20.   Return array
```

The predicate object algorithm first loads the properties configuration file, opens the connection database, then loads the triplet mapping file, and opens queries by line (line 1). Create an ArrayList container and initialize it (line 2). Query by line, multiple predicates and objects are obtained in each line, and predicate mapping is first performed for each set of predicate objects (lines 3-5). There are two main ways to get the predicate attribute value (line 7): rr:predicateMap (line 6) and rr:predicate. Store the resulting predicate property values in the ArrayList container (line 8). Then the corresponding object is mapped to the object, and the object attribute value is obtained by rr:objectMap and rr:object (lines 10-12). Store the resulting object property value in the ArrayList container (line 13). If there is a reference object map, the DoROM algorithm (lines 14-19) is continued. The obtained object property values are placed in the container, and the RDF dataset (line 20) in the ArrayList container is uniformly returned finally.

C. Reference Object Algorithm (DoROM)

The reference object mapping algorithm mainly implements the processing of the referenced object connection. The reference object mapping converts the logical table according to whether the reference object mapping is included in the triple map, and generates a corresponding RDF triple. The reference object mapping allows the subject of another triple mapping to be used as the subject generated by the predicate object mapping. Since both triple maps can be based on different logical tables, they may require a join between logical tables.

The reference object mapping is represented by the following resources:

There is only one rr:parentTriplesMap attribute, and its value must be a triple map, called the parent triple mapping of the reference object mapping.

There may be one or more rr:joinCondition attributes, whose value must be a join condition. A join condition is represented by two attributes, and each of them has a value:

rr:child its value is called the subcolumn of the join condition and must be the column name that exists in the logical table that contains the triple mapping of the reference object mapping.

rr:parent its value is called the parent column of the join condition, and must be the name of the column that exists in the logical table of the parent triplet mapping that references the object mapping.

The following is the reference object algorithm 3:

Algorithm 3 (**DoROM**)

Input: Database connection, spatiotemporal relationship data
Output: ArrayList array

```
1.    array←∅
2.    rr:objectMap←object(row，object)
3.    rr：object←object(row，object)
4.    rows←executeQuery
5.    For each row ∈ rows
6.      rr:RefObjectMap←RefObject(row，RefObject)
7.      rr:parentTriplesMap←-<TriplesMap1>
8.      JoinQuery(TriplesMap1，TriplesMap2)
9.      rr:joinCondition←( rr：child，rr：parent)
10.     rr：child←-<TriplesMap2>
11.     rr：parent←-<TriplesMap1>
12.     array.add(rr：RefObject)
13.   End for
14.   Return array
```

The reference object algorithm is required to load the property configuration file, open the connection to the relational database, load the triplet mapping file, set up an ArrayList container, and initialize an ArrayList container (line 1). Object mapping is performed for each set of reference objects. Object property values are obtained mainly in two ways (lines 2-3): rr:objectMap and rr:object. Perform the query operations line by line (line 4), map objects to each set of objects, and reference the parent triplet mapping of the object mapping to logical table 1 (lines 5-7). Join the two logical tables. Firstly, rr:parentTriplesMap attribute is used to represent the first logical table as the connected logical table. Then, the connection conditions rr:child and rr:parent are used to ensure that the two logical tables can be joined (lines 8-9). The rr:child and rr:parent properties represent the fields corresponding to the two logical tables, rr:child for the connected property field (line 10), and rr:parent for the connected property field (line 11). Finally, the value is added to the container, returning the RDF dataset (lines 12-14).

III. …SPARQL QUERY

With the development of semantic Web, more and more fuzzy spatiotemporal RDF data are published on the Web,

and the query of fuzzy spatiotemporal RDF data also attracts people's attention. Since the W3C organization recommended the use of SPARQL to query RDF data, various methods have emerged for querying fuzzy spatiotemporal RDF data with SPARQL. Therefore, the paper from two aspects carries on the research explanation in the following.

A. Fuzzy SPARQL structure quantifier

The fuzzy SPARQL structure quantifier is a representation of the fuzzy of an exact number in a SPARQL query. Assuming an exact number X, a fuzzy SPARQL structure quantifier can be expressed as "about X", "at least X", "at most X", " Close to X", "very close to X" and other forms, such as "about 1000", "at least 1000", "at most 1000", " Close to 1000", "very close to 1000" and so on. These quantifiers correspond with static attribute intervals one-to-one, and each interval value has an ordered characteristic. Therefore, combining the ordered linguistic value characteristics with the fuzzy set theory, and constructing 13 ordered linguistic value subdomains using the ladder membership function, the fuzzy value of each subdomain can be expressed as a quaternion form to describe , such as (a_i, b_i, α_i, β_i), where a_i represents the lower bound of the subdomain membership interval, b_i represents the upper bound of the subdomain membership interval, α_i represents the fuzzy adjustment distance of the subdomain and the previous adjacent subdomain, and β_i represents the blur adjustment distance of the subdomain and the next adjacent subdomain. A description of the 13 ordered language values for typhoon data is given below. See Table I below.

According to the ordered language value subdomain table given in this paper, the standard SPARQL query can be fuzzy extended. Firstly, we allow fuzzy terminology and linguistic value subdomain identification to represent typhoon speed and pressure in the clause of the filter. Secondly, in order to ensure that the user can clearly express the degree of ambiguity of the query, it is also necessary to add "with S_i" after the filter. Finally, according to the numerical interval in the table corresponding to S_i, we can specify the scope of the fuzzy quantifier query.

For example, query "data information with very strong typhoon data pressure", we can express it as "filter (? pressure = "very strong") with S_{10}". For the fuzzy SPARQL statement of the above typhoon data, the system will automatically extract the fuzzy value existing in the request according to the users' request to query the fuzzy request, and map to the corresponding subdomain in Table I according to this value. Then determine the membership degree. At last, a complete SPARQL statement representing fuzziness is generated.

The complete fuzzy SPARQL query statement has the following structure:

SELECT ?object

WHERE {?object hasproperty ?property.

FILTER (?property=FuzzyTerm) with S_i }；

In the form above, FuzzyTerm in FILTER indicates the value of the fuzzy term input by the users. S_i is a global variable owned by the filter condition and corresponds to the subdomain interval to be defined. The system will automatically map to the subdomain, according to FuzzyTerm. In the table I, the subdomain label corresponding to the fuzzy term value is brought back to FILTER to generate the query filter condition.

B. Query processing of fuzzy quantifiers

The query value intervals corresponding to general quantifiers are static. According to the characteristics of linguistic variable values, we can divide fuzzy quantifiers into two categories: the range of expression and the degree of expression. Quantifiers that express a range often have characteristics of numerical ranges, such as quantifiers with vague words such as "about" as a prefix. Quantifiers that express degree are often quantifiers with prefixes such as "very" or "extreme". We will give a processing method for the features of these two types of fuzzy quantifiers in the query conditions.

1) Conversion of range fuzzy quantifiers

Fuzzy quantifiers that represent ranges: Usually the words "about", "close", "not more than", "at least", etc. are added before the exact words, and the exact words are blurred. If there is a condition about the fuzzy quantifier in the users' query statement, we call this query a fuzzy query that represents the range. The representation of the fuzzy term of the range is generally defined by the convex fuzzy set membership function. When the query processing is performed, it can be divided into two different cases according to the fuzzy query condition:

TABLE I. ORDERED LANGUAGE VALUE SUBDOMAIN TABLE

Subdomain	Fuzzy Term	Fuzzy Value
S12	absolute fast，absolute strong	（1.0，1.0，0.0，0.0）
S11	extremely fast，extremely strong	（0.9，1.0，0.0，0.5）
S10	very fast，very strong	（0.81，0.95，0.05，0.04）
S9	fast，strong	（0.71，0.85，0.04，0.05）
S8	a little fast，a little strong	（0.62，0.76，0.05，0.04）
S7	slightly fast，slightly strong	（0.52，0.66，0.04，0.05）
S6	medium speed，medium strength	（0.43，0.57，0.05，0.04）
S5	slightly slow，slightly weak	（0.33，0.47，0.04，0.05）
S4	a little slow，a little weak	（0.24，0.38，0.05，0.04）
S3	slow，weak	（0.14，0.28，0.04，0.05）
S2	very slow，very weak	（0.05，0.19，0.05，0.04）
S1	extremely slow，extremely weak	（0.0，0.9，0.04，0.0）
S0	absolute slow，absolute weak	（0.0，0.0，0.0，0.0）

(1) Extreme value processing: When the upper limit ("at most *Y*") or the lower limit ("at least *Y*") appears in the fuzzy quantifier, it can mean that "at most *Y*" is ≤ *Y*, meaning "at least *Y*" is ≥ *Y*.

(2) Range processing: When the range constraint ("about *Y*" or "close to *Y*") appears in the fuzzy quantifier, the fuzzified membership interval mapping can be performed by Table I. Then the membership function is selected to determine the value interval, and the fuzzy query condition is transformed into an exact value interval for query finally.

Taking "about *Y*" as an example, the extension method of the intermediate membership function of Cauchy distribution is used to formulae 1. The extremum of the right half of the α-intercept is obtained according to the left edge value of the membership degree of the mapping interval, and then the right value is determined. Using the symmetry of the convex fuzzy set determines the range of values of the left part, and obtains the range of values of the linguistic variables finally.

$$\mu_{\text{about Y}}(x)=\frac{1}{1+(\frac{x+y}{\beta})^2} \tag{1}$$

In this formula 1, y is an exact number, and β is the adjustment value of the curve width. The range of the query precision can be adjusted according to β.

2) Conversion of degree fuzzy quantifier

In some natural languages, the words "very, extremely, slightly" are prefixes for the quantifiers, which adjust the meaning of the quantifier and turn the original quantifier into a new quantifier. The fuzzy quantifier that expresses the degree: When mood operators are used as prefixes to modify fuzzy quantifiers, fuzzy quantifiers expressing degree are formed. For example, if the linguistic variable x is "pressure", we can add a set of mood operators in front to represent fuzzy queries representing degree. The set of mood operators can be expressed as W(x)= {absolutely strong, very strong, very Strong, strong, weak, very weak, extremely weak, absolutely weak}. This mood operator is an ordered language value. When performing the query, each mood operator can be mapped to the subdomain corresponding to Table I to determine the interval of membership degree. This interval is [0, 1]. Finally, according to the membership degree interval, the function of the membership degree is called to complete the query of the fuzzy quantifier representing the degree. The conversion process is as follows:

(1) The fuzzy quantifier representing the degree is extracted according to the users' query conditions, and then the interval of fuzzy value is determined according to table I. For example, we want to query data with very strong typhoon data pressure. First, we will extract the fuzzy quantifier "very strong" in the query, and then correspond to the membership interval label S10 in the table I. The corresponding interval is [0.81, 0.95], so we get an equation [very Strong]= S_{10}={x， $\mu_A(x)|\ \mu_A(x)\in\ \alpha$， $\alpha\in$[0.81,0.95]}.

(2) Call the trapezoidal membership function (2) to obtain the value range of the query field. Assume that the linguistic variable in the query condition is "workload" and its value interval is $n\in$[2, 30]. To query the "workload is very heavy", the trapezoidal membership function formula (2) is called. Determine the membership interval of the threshold a, and find the value of "very heavy workload" is $22 \le n \le 29$.

$$\mu_A(x)=\begin{cases}0, & x\le a\\ \dfrac{x-a}{b-a}, & a<x\le b\\ 1, & x>b\end{cases} \tag{2}$$

(3) Generate standard SPARQL queries. After the fuzzy condition processing, some fuzzy terms are converted into accurate query conditions, and then the standard SPARQL query is obtained to draw experimental results. Remove fuzzy conversion rules.

According to the transformation of fuzzy quantifiers proposed above, we can summarize three rules for transforming fuzzy quantifiers into exact words, which are fuzzy quantifier conversion rules for extreme values, fuzzy quantifier conversion rules for ranges, and fuzzy quantifier conversion rules for degree, respectively. After the transformation using the three rules, the standard SPARQL query is executed.

The specific defuzzification conversion rules are shown in Table II.

IV. ... SPARQL QUERY EXPERIMENT

In this section, we conduct mainly SPARQL query experiments. According to different fuzzy quantifiers, three kinds of fuzzy queries are executed. Different parameter settings lead to different experimental results. To verify fully the feasibility of SPARQL fuzzy query, three kinds of queries are analyzed on recall and precision. In the process of obtaining recall and precision, two formulas are used to calculate its recall and precision. F-Score is used to balance recall and precision. The formula is as follows:

TABLE II. CONVERSION RULES

Type	Fuzzy Query Condition	Language Indicator	Exact Query Condition
fuzzy extreme values	?x = "at least *Y*"	/	filter(?x ≥*Y*)
	?x = "at most *Y*"	/	filter(?x ≤*Y*)
fuzzy range	?x = "about *Y*"	S10-S11	filter(?x ≥ a && ?x ≤ b)
	?x = "near *Y*"	S6-S11	filter(?x ≥ a && ?x ≤ b)
	?x = "a little near *Y*"	S8-S11	filter(?x ≥ a && ?x ≤ b)
fuzzy degree	?x = "extremely strong"	S11	/
		?x= "OrderedLanguage "	Table I

$$\text{Precision}=\frac{TP}{TP+FP} \quad (3)$$

$$\text{Recall}=\frac{TP}{TP+FN} \quad (4)$$

$$\text{F-Score}=\frac{2*\text{Precision}*\text{Recall}}{(\text{Precision}+\text{Recall})} \quad (5)$$

Equation 3 represents the precision of the query results, where Precision represents the ratio of the number of triples in the query to the total number of tuples retrieved in the dataset.

Equation 4 represents the recall of the query result, where Recall represents the ratio of the number of related tuples in the query result to the total number of related tuples in the retrieved data set, which is usually used to determine the quality of the query result.

Equation 5 represents the average of harmonic recall and precision.

A. Extreme value fuzzy quantifier query

This experiment carries mainly out the feasibility verification of the fuzzy quantifier query of extreme values. In the experiment, we query the recall and precision under the condition of pressure "at most *Y*".

Then we can extract the fuzzy quantifier which is "at most" according to the query conditions, giving its membership function parameter μ five values of 0.5, 0.6, 0.7, 0.8, 0.9. For the exact word *Y*, 900hpa, 920hpa, 940hpa, 960hpa, 980hpa, and 1000hpa are set to conduct effective query. Afterward, we perform 30 sets of experiments on the extreme fuzzy quantifiers to verify the feasibility of the query. The following two tables III and IV represent the recall and precision under different membership parameters. Table III is obtained by Equation 3 and table IV is obtained by Equation 4 as follows.

Through the table data, we can clearly see the recall, precision and F-Score of the extreme value fuzzy quantifiers under different parameters by drawing the line graph. As shown in Fig 1, Fig 2, Fig 3 below.

In the query experiment of extreme value fuzzy quantifiers, we can clearly see through the line graph that the recall is decreasing and the precision is increasing with the increase of the membership parameter under different pressures. When the membership degree parameter is equal to 0.5, the recall can reach 90%. When the membership degree parameter is equal to 0.9, the precision can be as close as 90%. Therefore, it can be explained that the query about the extreme value fuzzy quantifier is feasible. When the pressure is 900, 920, 940, 960, 980 and 1000 respectively, the average value according to the formula 5 F-Score is approximate: 0.77, 0.78, 0.76, 0.76, 0.77, respectively.

TABLE III. PRECISION OF EXTREME VALUE FUZZY QUANTIFIER

	Precision					
P(hpa)	900	920	940	960	980	1000
$\mu = 0.5$	0.9	0.88	0.9	0.91	0.89	0.87
$\mu = 0.6$	0.85	0.82	0.85	0.84	0.82	0.83
$\mu = 0.7$	0.82	0.78	0.79	0.8	0.82	0.81
$\mu = 0.8$	0.77	0.73	0.75	0.76	0.76	0.78
$\mu = 0.9$	0.7	0.68	0.69	0.72	0.7	0.69

TABLE IV. RECALL OF EXTREME VALUE FUZZY QUANTIFIER

	Recall					
P(hpa)	900	920	940	960	980	1000
$\mu = 0.5$	0.68	0.69	0.65	0.67	0.68	0.7
$\mu = 0.6$	0.72	0.73	0.71	0.7	0.69	0.68
$\mu = 0.7$	0.75	0.74	0.72	0.71	0.75	0.73
$\mu = 0.8$	0.81	0.84	0.82	0.8	0.79	0.78
$\mu = 0.9$	0.87	0.89	0.86	0.85	0.84	0.86

B. Range fuzzy quantifier query

This experiment carries mainly out the feasibility verification of the range fuzzy quantifier query. In the experiment, we query the recall and precision under the condition of pressure "about *Y*". For range fuzzy quantifiers, this paper uses 30 sets of parameters and conducts the same experiments as above to verify the feasibility of the query. The following two tables V and VI represent the recall and precision under different membership parameters.

Table V is obtained by Equation 3 and table VI is obtained by Equation 4 as shown below.

Through the table data, we can clearly see the recall, precision and F-Score of the range fuzzy quantifiers under different parameters by drawing the line graph. As shown in Fig 4, Fig 5, Fig 6 below.

In the query experiment of the range fuzzy quantifiers, we can clearly see through the line graph that the recall is

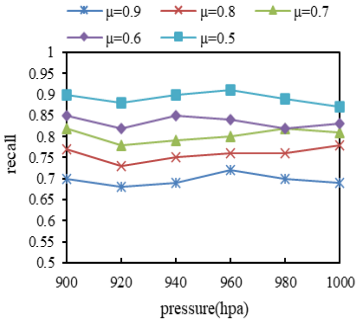

Fig. 1. Recall of extreme value fuzzy quantifier

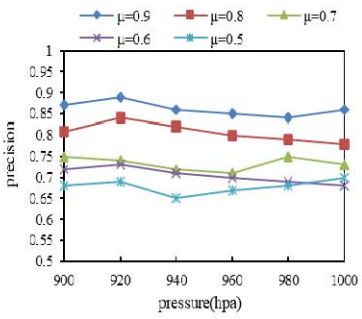

Fig. 2. Precision of extreme value fuzzy quantifier

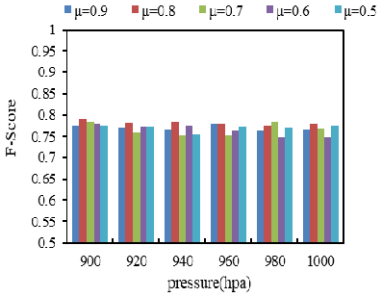

Fig. 3. F-Score of extreme value fuzzy quantifier

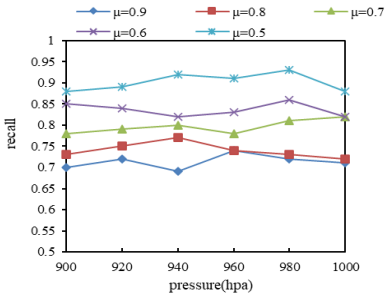

Fig. 4. Recall of range fuzzy quantifier

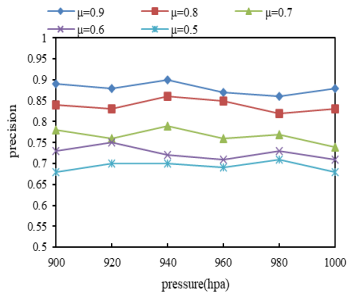

Fig. 5. Precision of range fuzzy quantifier

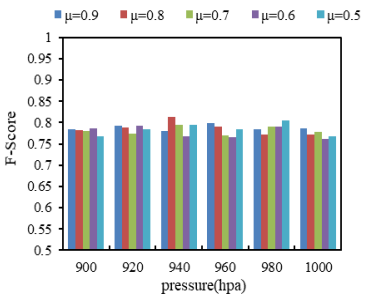

Fig. 6. F-Score of range fuzzy quantifier

decreasing and the precision is increasing with the increase of the membership parameter under different pressures.

When the membership degree parameter is equal to 0.5, the recall can reach 90%. When the membership degree parameter is equal to 0.9, the precision can be as close as 90%. Therefore, it can be explained that the query about the range fuzzy quantifier is feasible. When the pressure is 900, 920, 940, 960, 980 and 1000 respectively, the average value according to the formula 5 F-Score is approximate: 0.78, 0.78, 0.78, 0.77, 0.77.

C. *Degree fuzzy quantifier query*

This experiment carries mainly out the feasibility verification of the degree fuzzy quantifier query. In the experiment, we query the precision and recall under the "extremely strong" pressure. Meanwhile, we carry out the experiments under the five conditions of membership parameter μ= {0.5, 0.6, 0.7, 0.8, 0.9} for the degree fuzzy quantifiers. As shown in Fig 7 and Fig 8.

Through the histograms, we can clearly see that the recall is increasing and the precision is decreasing with the increase of the membership parameter.

In the experiment on degree fuzzy quantifiers, when the membership parameters are 0.5, 0.6, 0.7, 0.8, 0.9, the recall for the strong typhoon pressure is 0.72, 0.76, 0.83, 0.86, 0.9, respectively, and the precision is 0.9, 0.84, 0.78, 0.73, 0.68 respectively. The average value of F-Score calculated by Equation 5 is 0.79.

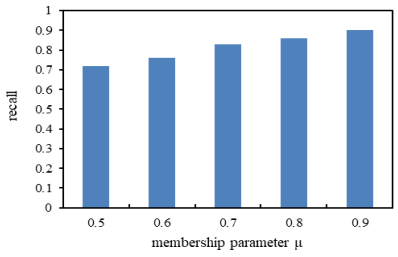

Fig. 7. Recall of degree fuzzy quantifier

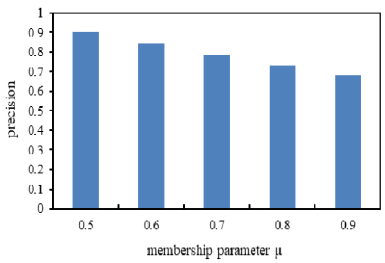

Fig. 8. Precision of degree fuzzy quantifier

TABLE V. PRECISION OF RANGE FUZZY QUANTIFIER

	Precision					
P(hpa)	900	920	940	960	980	1000
$\mu = 0.5$	0.89	0.88	0.9	0.87	0.86	0.88
$\mu = 0.6$	0.84	0.83	0.86	0.85	0.82	0.83
$\mu = 0.7$	0.78	0.76	0.79	0.76	0.77	0.74
$\mu = 0.8$	0.73	0.75	0.72	0.71	0.73	0.71
$\mu = 0.9$	0.68	0.7	0.7	0.69	0.71	0.68

TABLE VI. RECALL OF RANGE FUZZY QUANTIFIER

	Recall					
P(hpa)	900	920	940	960	980	1000
$\mu = 0.5$	0.7	0.72	0.69	0.74	0.72	0.71
$\mu = 0.6$	0.73	0.75	0.77	0.74	0.73	0.72
$\mu = 0.7$	0.78	0.79	0.8	0.78	0.81	0.82
$\mu = 0.8$	0.85	0.84	0.82	0.83	0.86	0.82
$\mu = 0.9$	0.88	0.89	0.92	0.91	0.93	0.88

D. *Comparison of fuzzy quantifier query experiments*

In this section, by comparing the keyword-based RDF data query with the method proposed in this paper, the keyword-based RDF data query can be regarded as a directed graph with the subject and object of the RDF triple as nodes and the predicate as the edge. In this experiment, different fuzzy quantifiers are selected for six comparative experiments, such as the extreme value fuzzy quantifiers "at least" and "at most", the range fuzzy quantifiers "about" and "close", and the degree fuzzy quantifiers "very" and "extreme". This section of the experiment is carried out on the actual typhoon data set to effectively analyze the recall and precision of the two methods. The specific content of the experiment is shown in Fig 9 and Fig 10.

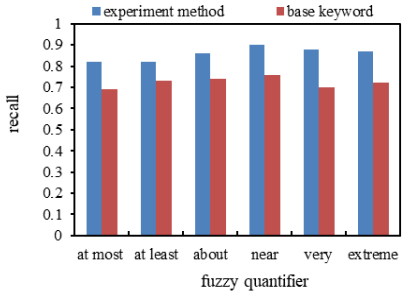

Fig. 9. Recall of comparison results

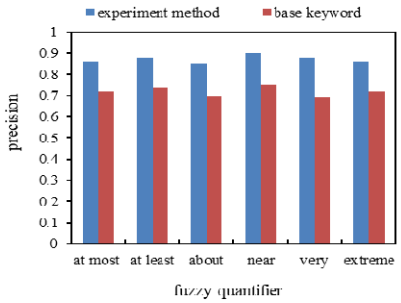

Fig. 10. Precision of comparison results

As shown in Figure 9, the recall of the query of three different fuzzy quantifiers is 0.85, 0.82, 0.86, 0.9, 0.88, and 0.87, respectively. The recall of the keyword-based RDF data query is 0.69, 0.73, 0.74, 0.76, 0.7, and 0.72, respectively. As shown in Figure 10, the precision of the query of the three different fuzzy quantifiers is 0.86, 0.88, 0.85, 0.9, 0.88, and 0.86, respectively. The precision of the keyword-based RDF data query is 0.72, 0.74, 0.7, 0.75, 0.69, and 0.72, respectively. The average value of the F-Score proposed in this paper is 0.86, while the keyword-based RDF data query is 0.72. Therefore, the method proposed in this paper is much better than the keyword-based RDF data query. It is proved that the keyword-based RDF data query performance is not very good, so this paper has a good performance in the fuzzy quantifier query.

CONCLUSION

In this paper, a fuzzy spatiotemporal data model is established. Based on this model, fuzzy spatiotemporal data is stored in a relational database. Then, R2RML is used to transform fuzzy spatiotemporal data from relational database into RDF. Afterward, three efficient algorithms are proposed for triplet mapping. Meanwhile, three different fuzzy quantifiers are proposed, which can effectively query fuzzy spatiotemporal RDF data based on SPARQL. Finally, we verify the recall and precision of the query experiment, and the experimental results show the feasibility of our proposed method.

In the near future, we plan to continue our work by investigating two issues not addressed in this paper. On the one hand, we consider improving the better fuzzy spatiotemporal data model by analyzing some common features among fuzzy spatiotemporal data in a more detailed way, so as to make its application scope wider. On the other hand, the queries are efficient when the dataset is relatively small, but less efficient when the dataset is extremely large. Therefore, we plan to improve the core algorithm of the transformation to ensure the stable relatively time efficiency.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (61402087), the Natural Science Foundation of Hebei Province (F2019501030), the Natural Science Foundation of Liaoning Province (2019-MS-130), and the Fundamental Research Funds for the Central Universities (N2023019).

REFERENCES

[1] H. Aidan, M. Marc, et al., “Towards fuzzy query-relaxation for RDF,” In Proceedings of 9th Extended Semantic Web Conference, Heidelberg,Berlin, pp. 687-702, 2012.

[2] N. Obeid, “A formalism for representing and reasoning with temporal information, event and change,” Applied Intelligence, vol 23,pp. 109-119, 2005.

[3] L. Hu, W. S. Ku, S. Bakiras, et al., “Spatial query integrity with voronoi neighbors,” IEEE Transactions on Knowledge and Data Engineering, vol. 25, pp. 863-876, 2013.

[4] A. Sözer, A. Yazici, and H. Oğuztüzün, “Indexing fuzzy spatiotemporal data for efficient querying: A meteorological application,” IEEE Transactions on Fuzzy Systems, vol. 23, pp. 1399-1413, 2015.

[5] S. S. Sahoo, W. Halb, S. Hellmann, et al., “A survey of current approaches for mapping of relational databases to RDF,” W3C RDB2RDF Incubator Group Report, vol. 1, pp. 113-130, 2009.

[6] P. Johannesson, “A method for transforming relational schemas into conceptual schemas,” In Proceedings of 1994 IEEE 10th International Conference on Data Engineering, pp. 115-122, 1994.

[7] L. Stojanovic, N. Stojanovic, and R. Volz, “Migrating Data-intensive WebSites into the Semantic Web,” In Proceedings of the l7th ACM Symposium on Applied Computing (SAC), pp. 1100-1107, 2002.

[8] M. Hert, G. Reif, and H. C. Gall, “A comparison of RDB-to-RDF mapping languages,” In Proceedings of the 7th International Conference on Semantic Systems, pp. 25-32, 2011.

[9] J. Petrini and T. Risch, “Processing queries over RDF views of wrapped relational databases,” In Proceedings of first International Workshop on Wrapper Techniques for Legacy Systems, pp. 16-29, 2004.

[10] C. P. Laborda and S. Conrad, “Bringing relational data into the semantic web using sparql and relational.owl,” In Proceedings of the 22nd International Conference on Data Engineering Workshops, pp. 55-62, 2006.

[11] M. Frank and M. Eric, “RDF primer,” W3C Recommendation 10 February 2004. Available: http://www.w3.org/TR/2004/REC-rdf-primer-20040210.

[12] L. Y. Bai, L. Yan, and Z. M. Ma, “Determining topological relationship of fuzzy spatiotemporal data integrated with XML twig pattern,” Applied Intelligence, vol. 39, pp. 75-100, 2013.

[13] M. Baharam, K. G. Peay, and L. Tedersoo, “Local□scale biogeography and spatiotemporal variability in communities of mycorrhizal fungi,” New Phytologist, vol. 205, pp. 1454-1463, 2015.

[14] T. Emrich, H. P. Kriegel, N. Mamoulis, et al., “Indexing uncertain spatio-temporal data,” In Proceedings of the 21st ACM international conference on Information and knowledge management, pp. 395-404, 2012.

[15] P. J. Williams, M. B. Hooten, J. N. Womble, et al., “An integrated data model to estimate spatiotemporal occupancy, abundance, and colonization dynamics,” Ecological Society of America, vol. 98, pp. 328-336, 2017.

[16] J. S. Son, J. D. Kim, and D. K. Baik, “A storage-independent model for SPARQL-to-SQL translation algorithms,” In Proceedings of Applied Mechanics and Materials, vol. 764, pp. 796-800, 2015.

[17] M. Arenas, A. Bertails, and E. Prud’hommeaux, et al., “A direct mapping of relational data to RDF,” W3C recommendation, vol. 27, pp. 1-11, 2012.

[18] T. Berners-Lee, “Relational databases on the semantic web,” W3C Design Issues, 2001.

[19] D. J. Dubois, H. Prade, and R. R. Yager, “Readings in fuzzy sets for intelligent systems,” California: Morgan Kaufmann, 2014.