# Designment of E-R model based on RDF(S)

Xiangbin Gao[1,2], Dantong Ouyang[1,2], and Yuxin Ye[1,2] ⋆

[1]College of Computer Science and Technology, Jilin University, Changchun 130012, China
[2]Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Changchun 130012, China

**Abstract.** Based on the vast domain resources of RDF(S) on the web and SPARQL's powerful query ability, this article presents a new method of designment of E-R model. The steps for this design are: (1) Formulating SPARQL rules (including resource query rules and schema query rules) by the analysis of RDF(S)'s structure. (2) Parsing the optimal resource obtained through the query sentences. (3) Completing the designment by taking advantages of the translation from RDF(S) model to entity-relationship model in accordance with the content queried. The results indicate that, the designment of E-R model based on RDF(S) could restore user real requirements of great possibilities and help database designer to complete design in a strange area.

**Keywords:** RDF, RDF Schema, E-R model, SPARQL, database design

## 1 Introduction

Generally speaking, database design depends, to a large extent, on the designer's understanding and representing of user requirements[1]. Passed through 60 years development, the main ideas about database construction have improved a lot. Much research has been put forward. Kahn raised a way of obtaining description information in the process of database design[2], which emphasized the designer should gather requirement in the real world and judge the requirement's sufficiency and completeness during the process of design. Blaha, Premerlani came up with the method of Object-Oriented to build relational database[3], it promoted adherence to normal forms and improved integration between databases and applications. Finkelstein, Schkolnick took advantage of the calculation method of heuristic to optimize data dictionary and applied it in relationship database design[4]. Asuman, Birol designed a generalized expert system for database design[5], they optimized the expert system by adding new design approach and modifying older method and formed the human interaction mode. Unfortunately, because of the shock of the web, the traditional methods can not longer fit the needs in some certain mode, so it's confronted with more complicated problems.

Resource Description Framework (RDF)[6] provides a set of data models to support the description of domain resources stored on the web and has strong

⋆ corresponding author: yeyx@jlu.edu.cn (Yuxin Ye).

ability on the knowledge sharing. SPARQL[7] just offers the powerful query capabilities to RDF(S) and could be able to analyze RDF deeply. Based on vast amounts of RDF resources and SPARQL, this article brings a fresh perspective about designment of E-R model. As a result, the designment further strengthens the database design system and satisfies the user in special fields a lot.

## 2 Frame structure

Based on the strong ability on knowledge sharing, we choose RDF as design source. Particularly, we use WordNet[8] and Swoogle[9] to obtain user requirements. In this article, we propose a designment of E-R model based on RDF(S) and it is built around the traditional method. There are three main tasks:

(1) Based on the structure features of RDFS, formulating the relevant query rules of SPARQL;

(2) Parsing the RDF resources through the SPARQL query rules;

(3) Drawing E-R model by transformation from RDF(S) model to E-R model.

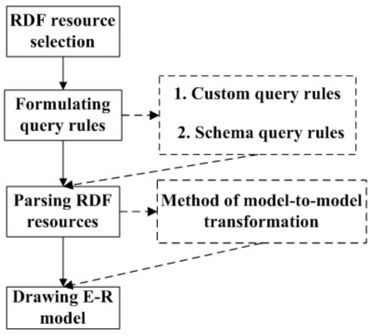


**Fig. 1.** Frame structure

For task (1), formulating query rules by the analysis of RDF(s)'s structure and writing query sentences. Especially, we propose resource query rules which designs by the file's own namespace and schema query rules which designs by RDF Schema's namespace. For task (2), using the method of Width First Traversal to traverse RDF by the sentences obtained from task (1) to get results, such as classes, properties and data types. For task (3), using the rules of transformation between RDF(S) model which describes the structure of RDF(S) and E-R model to draw E-R model through the analysis and arrangement of the results from task (2). The frame structure is shown in Figure 1.

## 3 Formulating query rules

For the designment of E-R model by RDF(S), we parse RDF resources by SPARQL to transform the elements from RDF(S) model to E-R model. As SPARQL mainly uses the rules in WHERE, so the core content of this article is focus on the query rules. To keep things simple, the rules below mentioned keep only the part of WHERE, the parts of PREFIX, SELECT and FROM are all ignored.

## 3.1 Resource query rules

Like RDF, SPARQL also provides the matching pattern of triple. Unlikely, the triple's subject, predicate and object all can be variable. The triples in SPARQL are used to match the triples in RDF and there will be the results if they matches successfully. Because every RDF file has its own namespace and builds the file structure under it, so we can not query all of the information with a set of inherent rules. The designer has to design the query sentences by the specific namespace. In this case, we put forward the resource query rules in table 1.

**Table 1.** Resource query rules

| | Resource query rules | Note |
|---|---|---|
| **Rule1:** | { ?s ?p ?o } | Listing all the triples in the RDF resources. |
| **Rule2:** | { ?s p ?o }<br>{ ?s ?p o }<br>{ s ?p ?o } | The condition of one known item and two unknown items. |
| **Rule3:** | { ?s p o }<br>{ s ?p o }<br>{ s p ?o } | The condition of two known items and one unknown item. |
| **Rule4:** | { ?s p1 ?o}<br>UNION {?s p2 ?o} | Union matching. |
| **Rule5:** | { ?s p1 ?o}<br>OPITIONAL {?s p2 ?o2} | Optional matching. |
| **Rule6:** | { ?s p ?o}<br>FILTER { ?o > cons } | Value restrictions matching. |

In order to have a comprehensive understanding of the file, we can get all the triples by rule 1. As needed, we may often run into the situation that there are one known data item and two unknown or two known data items and one unknown, so we design rule 2 and rule 3. According to the content above, we can make a traverse only by one known data item. Based on the diversity of the semantic, we can add some constraints and deformation to the rules to make the semantic more richer. In rule 4, we use UNION to combine two triples together. As a result, there will be at least one branch matching the triples in RDF. In rule 5, we use OPTIONAL to combine two triples together. Specifically, the second triple will modify the first. In rule 6, we use FILTER to combine two triples together and the second triple will constrain the first. Furthermore, the constraints can be logical expressions of Boolean value.

Specially, there will be the blank nodes in the results. The blank node is a special variable and it only matches the element whose data type is blank.

Otherwise, the blank node doesn't represent the real meaning and it only holds up some space to connect the other two nodes. During the process of query, we will meet the blank nodes inevitably. At this time, designer should ignore the blank node and skip it to search the bilateral nodes.

### 3.2 Schema query rules

About RDF, in addition to the user defined namespace, RDF Schema also provides its own namespace for the layout of the structure of RDF. The rules mentioned above describe the resource rules, but the results are always disorderly and they are even meaningless in extreme cases. On the other hand, resource query is always blind and has low efficiency. Therefore, we put forward the schema query rules shown in table 2. So, we design the query sentence that one known data item connects with one unknown by the predefined vocabulary RDF Schema provides. Thus, as long as the designer understand the special semantics of the predefined vocabularies, he can obtain the unknown data item from the known easily. On the whole, the schema query rules not only improve the readability of RDF but also enhance the regularity of the query. Specially, like resource query rules, we can also add some constraints and deformation to schema query rules to complete more complex query and they are no longer to be described.

**Table 2.** Schema query rules

| | Schema query rules | Note |
|---|---|---|
| **Rule7:** | { ?var rdf:type rdfs:Class } | Class's query. |
| **Rule8:** | { ?var rdfs:subClassOf cons } | Class's inheriting. |
| **Rule9:** | { cons rdf:type ?var} | Class's type. |
| **Rule10:** | { cons rdfs:seeAlso ?var}<br>{ cons rdfs:isDefinedBy ?var } | The relationship between object and subject |
| **Rule11:** | { cons rdfs:lable ?var}<br>{ cons rdfs:comment ?var} | Class's document. |
| **Rule12:** | { ?var rdf:type rdf:Property} | Property's query. |
| **Rule13:** | { ?var rdfs:subPropertyOf cons } | Property's inheriting. |
| **Rule14:** | { cons rdfs:domain ?var} | Property's domain. |
| **Rule15:** | { cons rdfs:range ?var} | Property's range. |
| **Rule16:** | { cons rdf:datatype ?var} | Property's data type. |
| **Rule17:** | { cons rdf:value ?var} | Property's value. |

Given the uncertainties associated with RDF deign, we use resource query rules and schema query rules comprehensively to improve the results' accuracy.

## 4 Parsing RDF resources

Benefiting by the hierarchical structure, we simulate the whole RDF file as the tree model and regard the triples as the nodes of the tree. For the elements we have found without redundancy and missing, this article takes the strategy of top-down and the method of gradually precision to query. Furthermore, Width First Traversal method is also used during the whole process. It is necessary to state that with the query nodes expanding, the semantic relevance between the follow-up nodes and the initial nodes will decrease rapidly. So, in order to pledge the usefulness, the designer should define the layer of query with actual requirement to improve the quality of query results.

## 4.1 The structure of RDF(S) model

In order to achieve the model-to-model transformation, it's necessary for us to analyze the models' structures with their characters to ensure the relation of the specific elements between the two models. As the source, RDF has rich semantic content and can describe the domain knowledge in more detail. We will get rdf-s:class which represents class and rdf:Property which represents property by the generalization of rdfs:Resource. About the RDF(S) model, the most important characteristic is hierarchy, the class can be inherited by rdfs:subClassOf, while the property by rdfs:subPropertyOf. Using the tree structure, rdfs:Resource acts as the root node while rdfs:Class and rdf:Property and their subclasses act as the sub-nodes to expand. Because RDF is a kind of resource to store information, so we save the classes and the attributes after they have been instantiated. For stipulating the class connected with property and the relationship between class and property, RDF provides rdfs:domain and rdfs:range to constrain the property's domain and range respectively. Based on the constraints, we will be able to find the relationship between the class and the property as well as the relationship between the classes. Finally, the RDF network will be presented clearly. The structure is showed in Figure 2.

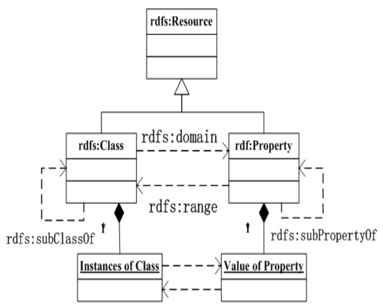


**Fig. 2.** The structure of RDF/RDFS

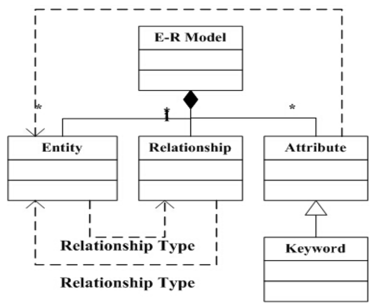


**Fig. 3.** The structure of E-R model

## 4.2 Method of query

**(1) Query against class.** Based on the characteristic of hierarchy, we can use rule 7 and rule 8 to search for all the classes and their sub-classes. The subclass not only has the attributes its parent class has, but also has its own. Correspondingly, the primary key and the foreign key are also inherited. As rule 9 says, rdf:type describes the connection between the instance and its corresponding class. Rule 10 defines the relationship between the subject class and the object class, it connects the classes with the keywords such as rdfs:seeAlso and rdfs:isDefinedBy. Document query provides the designer with a readable description and explains the class's specific definition by rule 11.

When traversing, the classes act as the starting points and the ones searched in the first case stored in queues. Then, putting out the first class and finding its sub-nodes as well as the sibling-nodes until the sub-nodes are all searched. After the traverse of the first class, we put out the second one and do the same process. Likewise, making a traverse for the whole tree. While, because of the namespace user defined, we may not get classes by rule 1 directly sometimes. So the designer should traverse from the result's first record. By the instance and the property mentioned in the first record, we will find the related class.

**(2) Query against property.** Similar to the structure of class, property also has the corresponding hierarchical structure. According to rule 12 and 13, we can respectively find all the properties and their sub-properties. RDF Schema is a description framework centered on property. It utilizes domain class qualified by rdfs:domain and range class qualified by rdfs:range to constraint the property's semantic. On the other hand, it also links up class and property closely. Frequently, using the constraints to query property and class which connects with given nodes is a major way to traverse. As above, we query the domain classes of property through rule 14 and the range classes by rule 15. The rules take the property as predicate and specify the subject's type as well as object's.

Due to the semantic absence of query, some properties only connects with instances rather than classes. In this case, we can only query by instances to find the relationship between property and class.

**(3) Query on datatype.** Since RDFS does not have its own datatype, it uses XSD's (XML Schema Datatype). If and only if the property value is text type, we will get the corresponding data. Specifically, we use rule 16 to query datatype and rule 17 to query the value.

**(4) Query on instance.** The instance of class is described by the corresponding property's value (the range of the property). Since the instance could be obtained by the method of data inquiry, so we won't give specific rules, designer could use the rules of datatype inquiry to accomplish the instance inquiry.

### 4.3 Semantic absence of query

Due to the truth that RDF is an open semantic framework, writers are not under the specific discipline to construct classes, properties and instances. Thus, they could use the namespace they have defined or RDFS predefined. In the reason that there is little or not any RDFS's predefined vocabularies in the RDF files, designer often needs resource query by the requirements. This leads to the situation that some elements are unlikely to be acquired. On the contrary, E-R model makes demands on the form and structure of data with unified and refine requirements. Base on the standard, all the elements are designed strictly to follow the regulation and they are not permitted by default. Therefore, it is inevitable to cause semantics absence when doing the transformation.

## 5 Drawing E-R model

### 5.1 The structure of E-R model

As design result, relational database has the ability of data storage and offers a standard to manage resources. Entity, relationship and attribute together constitute the E-R model whose structure is shown in Figure 3. The relationship defines the relation between entities and it extends the large network structure by connecting entities. An entity has several attributes, while the attributes modify the entity and express the entity's features. Among those, when an attribute or a set of attributes can uniquely determine the entity, we call it primary key. Specially, the relationship between the entities has many types, such as one-to-one (1:1), one-to-many (1:n) and many-to-many (m:n). The types express the correspondence of the entities.

### 5.2 Model-to-model transformation

**(1) Transformation of class.** We transform every class to an entity and take the class name as the entity name. Using the existing knowledge, the designer can select the proper attribute as primary key which can describe the class only such as ID, name, etc. The specific condition should be determined by the actual requirements. Particularly, if the attribute appeared in one class is also the keyword of some other classes, we call it foreign key. While transforming the models, the keywords ( primary key and foreign key ) should be marked.

**(2) Transformation of property.** According to the different semantics, property can be classified into text type (rdfs:Literal) and resource type (URI). When the type is text, we can take the property as attribute which corresponds to its domain class. Also, making the property name as attribute name and property datatype as attribute datatype. When the type is resource, we can take the property as relationship between the entities which correspond to the domain and the range. What's more, we also make the property name as relationship name. Based on the the query about the domains and the range's instances, we can ensure the cardinality of the relationship. Specially, if the entity in one hand has one instance and the other also has one, we call it 1:1; if the entity in one hand has one instance and the other has many, we call it 1:n; if the entity in one hand has many instances and the other also has many, we call it m:n.

**(3) Transformation of datatype.** The database makes a specific request for the datatype of attribute. Therefore, we need to specify the datatype in the model-to-model transformation. Taking the datatype of MySQL as an example, the corresponding relation of several main datatypes is given in Table 3. By comparison, it is found that there are many similarities between XSD datatype and SQL datatype, which also provides the possibility for the conversion.

**Table 3.** The correspondence of data type between XSD and SQL

| XSD | MySQL Database | XSD | MySQL Database |
|---|---|---|---|
| byte | TINYINT | time | TIME |
| int | INT | dateTime | DATETIME |
| long | BIGINT | gDay | DATE |
| short | SMALLINT | Name | VARCHAR(TEXT) |
| double | DOUBLE | string | VARCHAR(TEXT) |
| date | DATE | token | VARCHAR(TEXT) |

**Note:** *VARCHAR stores the variable length strings with the maximum length of 255 characters. If the length is greater than 255 characters, we use TEXT.*

In the actual process, the query and the transformation is a process of the two at the same time, there is no obvious sequence. In combination with the resource query rules and the schema query rules, we parse RDF of the tree structure and facilitate the transformation from RDF(S) model to E-R model. Figure 4 shows the corresponding relationship between RDF(S) model and E-R model.

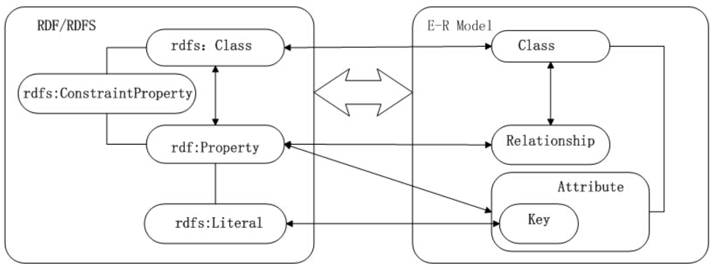
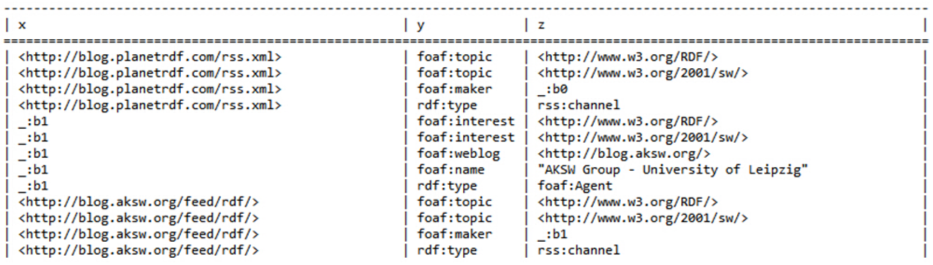


**Fig. 4.** The correspondence between RDF/RDFS model and E-R model

## 6 Example for the Designment based on "blogger"

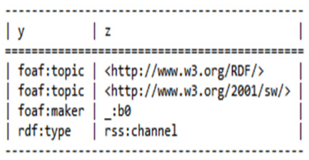
The article chooses the inference engine Jena[?] to assist SPARQL with querying and reasoning. The RDF file and the SPARQL file are used as input and the query results are used as output. Designer selects the content that he wants according to the RDF resources and writes some corresponding query sentences into SPARQL file.

As a typical example, we choose "blogger" whose address is "http:// wiki.creativecommons.org/ Special:ExportRDF/ Blogger" given by IBM to act as source file to design. Specificly, we use resource query rules and schema query rules to parse the file and the layer of query is defined at 5. Based on rule 1, we can list all of the triples which is shown in Figure 5. However, according to rule 7, we can not find any classes, so we traverse from the result's first record.



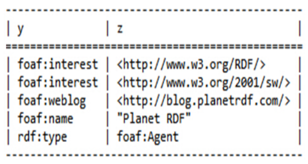
```
--------------------------------------------------------------------------------------------------------------------------------------
| x                                                  | y              | z                                                          |
======================================================================================================================================
| <http://blog.planetrdf.com/rss.xml>                | foaf:topic     | <http://www.w3.org/RDF/>                                   |
| <http://blog.planetrdf.com/rss.xml>                | foaf:topic     | <http://www.w3.org/2001/sw/>                               |
| <http://blog.planetrdf.com/rss.xml>                | foaf:maker     | _:b0                                                       |
| <http://blog.planetrdf.com/rss.xml>                | rdf:type       | rss:channel                                                |
| _:b1                                               | foaf:interest  | <http://www.w3.org/RDF/>                                   |
| _:b1                                               | foaf:interest  | <http://www.w3.org/2001/sw/>                               |
| _:b1                                               | foaf:weblog    | <http://blog.aksw.org/>                                    |
| _:b1                                               | foaf:name      | "AKSW Group - University of Leipzig"                       |
| _:b1                                               | rdf:type       | foaf:Agent                                                 |
| <http://blog.aksw.org/feed/rdf/>                   | foaf:topic     | <http://www.w3.org/RDF/>                                   |
| <http://blog.aksw.org/feed/rdf/>                   | foaf:topic     | <http://www.w3.org/2001/sw/>                               |
| <http://blog.aksw.org/feed/rdf/>                   | foaf:maker     | _:b1                                                       |
| <http://blog.aksw.org/feed/rdf/>                   | rdf:type       | rss:channel                                                |
```

**Fig. 5.** Listing all the triples



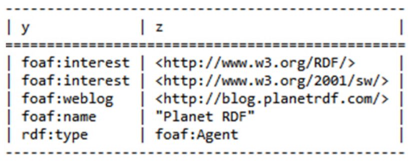
```
--------------------------------------------
| y          | z                            |
============================================
| foaf:topic | <http://www.w3.org/RDF/>     |
| foaf:topic | <http://www.w3.org/2001/sw/> |
| foaf:maker | _:b0                         |
| rdf:type   | rss:channel                  |
--------------------------------------------
```

**Fig. 6.** Querying with the instance



```
--------------------------------------------------
| y             | z                               |
==================================================
| foaf:interest | <http://www.w3.org/RDF/>        |
| foaf:interest | <http://www.w3.org/2001/sw/>    |
| foaf:weblog   | <http://blog.planetrdf.com/>    |
| foaf:name     | "Planet RDF"                    |
| rdf:type      | foaf:Agent                      |
--------------------------------------------------
```

**Fig. 7.** Querying with the blank node



```
--------------------------------------------------
| y             | z                               |
==================================================
| foaf:interest | <http://www.w3.org/RDF/>        |
| foaf:interest | <http://www.w3.org/2001/sw/>    |
| foaf:weblog   | <http://blog.planetrdf.com/>    |
| foaf:name     | "Planet RDF"                    |
| rdf:type      | foaf:Agent                      |
--------------------------------------------------
```

**Fig. 8.** Finding the cardinality

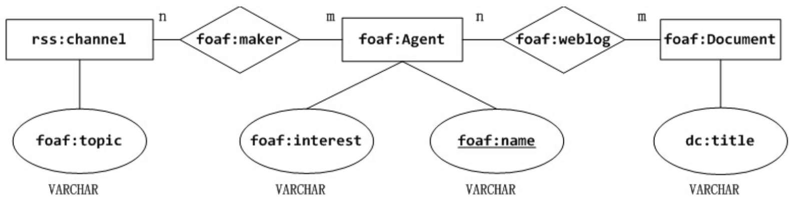


**Fig. 9.** E-R model of the Experiment

As we can see, the first record gives a description to the instance of ”<http://blog.planetrdf.com/rss.xml>”. Therefore, we use rule 9 to query the instances type and the result is ”rss:channel”. Unfortunately, we don’t have found some properties connected with ”rss:channel”. For this reason, we turn to give a query to the instance of ”<http://blog.planetrdf.com/rss.xml>” again. With rule 3, we can get the property (”foaf:topic”) of ”rss:channel”. In addition, its value and data type which is ”string”. Then, we transform it to ”VARCHAR” which is the data type of SQL. What’s more , we can also find the relationship ”foaf:maker” that connects "rss:channel" and a blank node. What is shown in Figure 6. According to the query sentence like

{ < http://blog.planetrdf.com/rss.xml > foaf:maker _:b0.
_:b0 ?y ?z. },

we will get the resource "<http://blog.planetrdf.com/>" connected with "rss:channel" by the blank node. Otherwise, we can also search for "foaf:Agent" which represents the blank node and its properties "foaf:name" and "foaf:interest". By convention, we regard "foaf:name" as primary key and underline it. What is shown in Figure 7. Based on the query about the classes "foaf:maker" connects with, we can ensure the cardinality is "m:n". What is shown in Figure 8.

Repeating the steps, we draw the E-R model about "blogger" in Figure 9.

## 7 Conclusion

Designment of E-R model based on RDF(S) is a new method for database design. It takes advantages of RDF's ability on knowledge sharing and makes use of the resources stored on the web vastly. The results show that the method could restore user real requirements of great possibilities and help database designer to complete design in a strange area.

## References

1. Sugumaran Vijayan, Storey Veda C: The role of domain ontologies in database design: An ontologymanagement and conceptual modeling environment,ACM Transactions on Database Systems,2006,31(3):1064-1094.
2. Kahn Beverly K: A method for describing information required by the database design process, Proceedings of the 1976 ACM SIGMOD international conference on Management of data, 1976:53-64.
3. Blaha Michael R., Premerlani William J.: Relational database design using an object-oriented methodology, Communications of the ACM,1988,31(4):414-427.
4. Finkelstein S.,Schkolnick M: Physical database design for relational database, ACM Transactions on Database Systems,1998,13(1),91-128.
5. Dogac A, Spaccapietra S.: A generalized expert system for database design, Software Engineering, IEEE Transactions on, 1989, 15(4): 479-491.
6. Ebiri Sejla, Goasdoue Francois: Query-Oriented summarization of RDF graphs, Proceedings of the 30th British International Conference on Databases, Edinburgh, United kingdom, July, 2015: 87-91.
7. Meehan Alan, Brennan Rob: SPARQL based mapping management, Proceedings of the 9th International Conference on Semantic Computing, Anaheim CA, United states, February,2015:456-459.
8. Zhang Li, Li Jing-Jiao, Hu Ming-Han: Implementation of Chinese WordNet, Dongbei Daxue Xuebao/Journal of Northeastern University,2003,24(4):327-329.
9. Ding Li, Finin Tim: Swoogle: A search and metadata engine for the semantic web, Proceedings of the13th ACM Conference on Information and Knowledge Management, Washington DC, United state, November, 2004:652-659.
10. Ameen Ayesha: Extracting knowledge from ontology using Jena for semantic web, Proceedings of 2014 International Conference for Convergence of Technology, Pune, India, April, 2014.