Atmospheric Lidar Data Storage Model Based on Ontology

Hao Chen,1 Dengxin Hua,2 Yikun Zhang,1 and Qing Yan2

1School of Computer Science & Engineering, Xi’an University of Technology, Xi’an 710048, China
2School of Mechanical & Instrumental Engineering, Xi’an University of Technology, Xi’an 710048, China

Correspondence should be addressed to Hao Chen; haozi2638@163.com

Received 19 September 2016; Accepted 26 January 2017; Published 6 March 2017

Hindawi
Scientific Programming
Volume 2017, Article ID 3607068, 8 pages
https://doi.org/10.1155/2017/3607068

Research Article

1. Introduction

Air pollutions, such as Acid rain, haze, and light chemical smoke, have become increasingly prominent in China, which has been considered as a serious threat to public health and environmental safety. In order to better understand the pollutions, many tools have been developed to monitor the environment. Lidar is an active instrument for vertical structure of atmospheric profiling with high accuracy and high spatial and temporal resolution. Recently, a number of regional atmospheric monitoring networks have been built based on lidar [1–4]. Analysis of lidar data from every monitored site is the key to investigating the regional air quality and temporal and spatial distribution of pollutants and to forecasting the emergency incidents, such as dust storms, forest fires, and volcanic eruptions [5–7].

Lidar observations have been carried out by many independent institutes, which leads to inconsistency of standards in terms of the protocol of data acquisition and data storage. Current lidar data structures which vary degrees of heterogeneity in structure and semanteme are hard to be shared without uniform data standard. The existing differences include the meaning of various concepts and the units of collected information. With the abilities of expressing and reasoning in semantic, ontology which has been widely used in data sharing and integration can describe the inner relationships between various parameters’ concepts or terms and reveal the hidden relationships by logical reasoning [8]. The ontology storage model of lidar data could be the prerequisite for exploring its huge quantity [9].

At present, there are four storage methods for ontology [10]: memory-based, plain text, specific management tools [11], and database [12, 13]. Methods of memory-based, plain text, and specific management have limitations in efficiency, analysis management, universality, and extensibility. The database storage model can further divide into three methods: relational database method, Native XML method, and object-oriented database method [14]. Among them, relational database method is the most popular and matured method in ontology storage. The limitation of this method is that database's pattern does not match ontology and has poor extensibility and semantic support. The Native XML method has better expansibility but lower efficiency. Object-oriented database method has higher semantic integrity and query efficiency but lacks the ability of storing the complex relationship between ontology’s concepts [15].

The lidar data have complex structures [16, 17] and mass storage [18] because the lidar data contained many atmospheric parameters with strong correlations. The above-mentioned storage methods have not been applied successfully to atmospheric lidar data storage. Based on the study of the relationship between lidar data and ontology structure,
we proposed and developed a mapping mechanism for lidar data ontology and object-oriented storage model.

In our previous work, the representation method and mapping mechanism of OODM of atmospheric lidar data were proposed and show that the OODM for lidar data is effective [19]. In present work, the details of database state of lidar data's OODM, the mapping mechanism of lidar data ontology to OODM, the data storage mode, and structure based on the OODM and the detailed process of the experiment are given.

2. Related Technologies

Ontology consisting of three elements, individual (instance), class (concept), and attribute [20], is a description of the concepts and relationships. The concept is the main part of the ontology, and the instance is the embodiment of the concept. Both the concept and the instance are the entity elements of the ontology, while the attributes are the descriptive elements.

Object-oriented data model (OODM) composed of object, class, and inheritance concept has been widely used in data modeling due to its flexibility in expressing the objects. Class describes a set of common characteristics of objects, and it is considered as a collection of objects with the same characteristics, where each object has the same properties and operations. Class is also associated with each other by inheritance, overwrite, and relationship [21].

The structure of ontology is similar to object-oriented data model in many respects. Concepts and attributes of ontology can be described and expressed with the classes and its attributes. Instances can be saved as object. Axiom and function of ontology can be simulated by the methods of OODM. Object-oriented method considers everything as a set of various objects. The complex objects can be decomposed into a combination of simple objects. Similarly, any complex knowledge of the ontology can be decomposed into a combination of basic concepts.

3. Lidar Data Ontology

The definitions of the lidar data’s ontology and the object-oriented model are given as follows.

Definition 1. A lidar data ontology O consists of five elements \{C, R, A, I, X\}, where C is a set of lidar data’s concepts; R is a set of relationships; A is a collection of attribute sets for each concept and relationship; I is a set of lidar data’s instances and X is a set of axioms [22].

Each concept \(c_i\), where \(c_i \in C\), represents a set of the same kind of objects and can be described by the same set of attributes, denoted by \(A^C(c_i)\). Each relationship \(r(c_p, c_q)\), then \(r_i \in R\), represents a binary association between concepts \(c_p\) and \(c_q\), and the instances of such a relationship are pairs of \((c_p, c_q)\). The attributes of \(r_i\) are denoted by \(A^R(r_i)\). Each axiom in \(X\) is either a constraint on the attribute values of concept and relationship or a constraint on the relationships between concepts objects. Each constraint can be expressed as a prolog (rule).

Definition 2. An object-oriented data model (OODM) of lidar data is a finite set of class declarations. Each class describes a series of atmospheric parameter objects with common attributes and methods. The OODM is a tuple \(S = \{Cs, As, Ds\}\), where \(Cs\) is a finite set of classes; \(As\) is a finite set of attributes; \(Ds\) is a finite set of class declarations. \(Ds\) contains exactly one declaration. The definition of declaration is defined as follows:

Class C is-a \(C_1, \ldots, C_k\) type-is \(T\),

where \(T\) denotes a type expression which was built based on the following syntax:

\[
T \rightarrow C |
\]

\[
\text{Union} T_1, \ldots, T_k \text{ End} |
\]

\[
\text{Set of } T |
\]

\[
\text{Record } A_1 : D_1, \ldots, A_k : D_k \text{ End}
\]

The part of is-a is optional, which represents class inheritance, for example, specified inclusion within the sets of instances of the involved classes, while the part of type-is which can be used to assign objects of the instances of C by the type expression \(T\) restricts the structures of the classes. Union, \ldots, End represents summary between an abstract class and a series of object classes.

Definition 3. The database instance \(J\) is a database state of lidar data’s OODM. The instance \(J\) is defined by the notions of object identifier and value. Given finite atomic domains \(D_1, D_2, \ldots, D_n, n \geq 1\), then \(D = D_1 \cup D_2 \cup D_3 \cup \cdots \cup D_n\), \(D_i \cap D_j = \emptyset, i \neq j\). \(V\) is atomic value.

Given an infinite set \(ID\), set \(V\) of value, and each \(id \in ID\) then \((id, v)\) is an object. Let \(o = (id, v)\), then \(identity(o) = id\), \(value(o) = v\). For \(O = \{a_1, a_2, \ldots, a_k\}\), then \(O = ID \times V\).

\[[[a_1 : v_1, a_2 : v_2, \ldots, a_k : v_k]] \in V, a_i \in A, v_i \in V, 1 \leq i \leq k\]

A database instance \(J\) is constituted by \(O', \pi', \rho'\), where \(O'\) is a finite set of object identifiers, \(\pi'\) is a mapping assigned to each class in \(C_j\) to a subset of \(O'\), denoted by \(C \in Cs, \pi'(C) \subseteq O'\), and \(\rho'\) is a mapping assigned a value in \(V\) to each object in \(O'\), denoted by \(OIDD \in OID, \rho'(OIDD) \subseteq V\).

4. The Mapping Mechanism of Lidar Data Ontology to OODM

As the definitions given in Section 3, ontology is constituted of structure and instance, while an object-oriented database is constituted of OODM and database instance. They share some commons in structure; for example, both of them express the concepts of the real world based on class concepts and relationships; both of them have also class and the correspondence properties.

The metamodel elements of ontology can be transformed into the metamodel elements of the OODM in structural
and instance. In order to fully support the object-oriented features, ontology concepts of lidar data have to be abstracted to core class, and ontology relationships would be abstracted to relationship class. The core class stores simple and fixed concepts with few correlation attributes. The relationship class stores relationships between different concepts and the relationship attributes of ontology [23].

Let $S = \{C, As, Ds\}$, which is the corresponding OODM of lidar data’s ontology $O_s$, and $\phi$ is the mapping rule of structure transformation; thus $S = \phi(Os) = \{Cs, As, Ds\}$. The mapping rules are described as follows:

(a) For each $c_i \in C$ in $O_s$, it can be mapped to the class $C_o$ of the OODM, $C_o \in Cs, \phi(c_i) \in Cs$.

(b) For each $r_i \in R$ in $O_s$, it can be mapped to the class $R_o$ of the OODM, $R_o \in Cs, \phi(r_i) \in Cs$.

(c) For each attribute set $A^C(c_i)$ of $c_i$ in $O_s$, $c_i \in C$, it can be mapped to the attribute $A_i$ of class C in the OODM, $\phi(A^C(c_i)) = A_i \in As$.

(d) For each attribute set $A^C(r_i)$ of $r_i$ in $O_s$, $r_i \in R$, it can be mapped to the attribute $A_i$ of class C in the OODM, $\phi(A^C(r_i)) = A_i \in As$.

(e) For each expression axiom of $c_i$ in $O_s$, $c_i \in C$, it can be mapped to the class C's declaration of the OODM: Class C is-a $C_1, \ldots, C_k$ or Class C type-is $T$.

(f) For each data type definition of $c_i$ in $O_s$, $c_i \in C$, it can be mapped to the class C's declaration of the OODM: Class C type-is $T$ End.

(g) For each data attribute axiom of $c_i$ in $O_s$, $c_i \in C$, it can be mapped to the class C’s declaration of the OODM: Class C type-is $T$ End.

(h) For each data attribute axiom of $c_i$ in $O_s$, $c_i \in C$, it can be mapped to the class C's declaration of the OODM: Class C is-a $C_1, \ldots, C_k$ type-is $Record A : Set of T$ End.

We supposed that the instance of the ontology $O_s$ and the data instance of the corresponding OODM are derived as follows:

(a) Each individual object identifier and class identifier in $O_s$ can be mapped to the object and class's instance of the OODM.

(b) Each individual data type identifier and object type identifier in $O_s$ can be mapped to the data type attribute and object type attribute.

(c) The individual axiom Individual(o type(C)) of $O_s$ can be mapped to the class declaration of the OODM's instance.

(d) The individual axiom of $O_s$: Individual(o, type(C)), value(R1, c1), ..., value(Rn, an), value(U1, v1), ..., value(Un, vn) can be mapped to the class declaration of the OODM's instance $o : [A_1 : V_1, \ldots, A_n : V_n]$.

The mapping algorithm from the lidar data ontology to the OODM is given by OWL DL syntax in Tables 1 and 2.

### Table 1: Identifier mapping algorithm.

<table>
<thead>
<tr>
<th>Syntax of ontology $O_s$</th>
<th>Mapping method</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_i \in C$</td>
<td>Add $\phi(c_i)$ to $Cs$</td>
</tr>
<tr>
<td>$r_i \in R$</td>
<td>Add $\phi(r_i)$ to $Cs$</td>
</tr>
<tr>
<td>$c_i \in A, A^C(c_i) \in A$</td>
<td>Add $\phi(A^C(c_i))$ to $As$</td>
</tr>
<tr>
<td>$r_i \in R, A^C(r_i) \in A$</td>
<td>Add $\phi(A^C(r_i))$ to $As$</td>
</tr>
<tr>
<td>$C \subseteq C_1 \cup C_2 \cdots \cup C_k$</td>
<td>Create(Class C is-a $C_1, \ldots, C_k$) add to $Ds$</td>
</tr>
<tr>
<td>$S = {C_s, A_s, D_s}$</td>
<td>Create(Class $C$ type-is $T$) add to $Ds$</td>
</tr>
<tr>
<td>$U \subseteq C, U \subseteq U_i \text{ or } 1U \subseteq C_i$</td>
<td>$Record A_1 : D_1, \ldots, A_k : D_k$ add to $Ds$</td>
</tr>
<tr>
<td>$T \subseteq \forall R_i \ . C$</td>
<td>$Record A : Set of T$ add to $Ds$</td>
</tr>
<tr>
<td>$R \subseteq C, R \subseteq U_i \text{ or } 1R \subseteq C_i$</td>
<td>$Record A : Set of T$ add to $Ds$</td>
</tr>
</tbody>
</table>

### Table 2: Instance mapping algorithm.

<table>
<thead>
<tr>
<th>Syntax of ontology $O_s$</th>
<th>Mapping method</th>
</tr>
</thead>
<tbody>
<tr>
<td>$o : C_1$</td>
<td>Create $o : type(C)$</td>
</tr>
<tr>
<td>$(o, o_1) : R$, $(o, v) : U_i$, $R_i, U_i \in C, o_1 \in O_s$</td>
<td>$Record o = (o_1 : v_1, a_2 : v_2, \ldots, a_k : v_k), a_i \in A$, $v_i \in V$</td>
</tr>
<tr>
<td>$o \in O_s$, $o_1 = o_2 = \cdots = o_k$, $o_i \neq o_j$</td>
<td>Create $o$ add instance(S)</td>
</tr>
</tbody>
</table>

### 5. The Storage Mode Based on the OODM

Currently, the object-oriented database management system has implemented most of the features of the OODM; however, it lacks the mathematical theory and is relatively inefficient compared to the relational database system with a rigorous mathematical foundation, which was based on the relational algebra theory and set theory [24]. To further improve query efficiency of lidar data ontology, we proposed a storage model based on the relational database.

The data table is built based on the vocabulary of the OWL and Definition 3 given in Section 3. The $Table$ Class contains the information of ontology about the OODM, including types, class names, and class identifiers. The table Property contains the properties of class or of relationship class in ontology, including the identifiers, types, and characteristics; the table PropertyOf describes the affiliation between the property and the class; the table SubSupClass stores the structure information between two classes; the table InstanceValue describes the property values of the instance in the form of a triple; the table InstanceOf describes the affiliations of the class and the instance; the table Restriction describes the constraints of the properties.

The mapping rule from the OODM to a relational database is defined as follows: for the class of the OODM, the class identifier, name, and type are created and stored in table...
Class; the variables of class are stored in the table Property, and property identifiers are then created. The properties and characters of the variables in class correspond to the Type and Character in Table Property. The table PropertyOf stores the identifiers of classes and of properties. The table SubSupClass stores the identifiers of the parent and child classes expressed the inheritance relationship of the two classes. The table InstanceValue stores the identifiers of classes and of objects. The property values and the special constraints are stored in the table InstanceValue and the property values are stored in the table Value. The table InstanceOf stores the identifiers of classes and of objects. The property values and the special constraints are stored in the table Restriction and correlated with each other in the identifiers of classes and of objects, as shown in Figure 1.

6. Application Example and Experimental

The atmospheric lidar data ontology storage model was examined using the experimental data collected in Laser Radar Center of Remote Sensing of Atmosphere of Xi’an University of Technology.

According to Definition 1, we can get a lidar data ontology $O_{\text{lidar}} = \{C_{\text{lidar}}, R_{\text{lidar}}, A_{\text{lidar}}, X_{\text{lidar}}\}$, where $A_{\text{lidar}} = A^C_{\text{lidar}} \cup A^R_{\text{lidar}}$. The parts of $O_{\text{lidar}}$ are defined as follows:

$C_{\text{lidar}} = \{\text{Aerosol, AtmosTemperature, Wind, AtmosHumidity, LidarData, Measure, OtherFactor, GeoInfo, AtmosPressure}\}$

$A^C_{\text{lidar}}(\text{Aerosol}) = \{\text{lidarratio, extinctioncoefficient, scatteringcoefficient}\}$

$A^C_{\text{lidar}}(\text{AtmosTemperature}) = \{\text{averagetemperature, temperaturedeparture}\}$

$A^C_{\text{lidar}}(\text{Wind}) = \{\text{winddirection, windspeed}\}$

$A^C_{\text{lidar}}(\text{AtmosHumidity}) = \{\text{watervapourdensity, relativehumidity, absolutehumidity, dewpoint, frostpoint, mix-ingratio}\}$

$A^C_{\text{lidar}}(\text{LidarData}) = \{\text{scatteringtype, wavelength, singlepulsetenergy, method, signalnoiseratio, rangeresolution}\}$

$A^C_{\text{lidar}}(\text{Measure}) = \{\text{azimuth, elevationangle, measuringtime, measuringheight, datavalue}\}$

$A^C_{\text{lidar}}(\text{Measure}) = \{\text{name, age, email}\}$

$A^C_{\text{lidar}}(\text{OtherFactor}) = \{\text{altitude, latitude, longitude, gravityacceleration, pressure}\}$

$A^C_{\text{lidar}}(\text{GeoInfo}) = \{\text{altitude, latitude, longitude, gravityacceleration, pressure}\}$

$A^C_{\text{lidar}}(\text{AtmosPressure}) = \{\text{altitude, latitude, longitudinal gravityacceleration, pressure}\}$

$R_{\text{lidar}} = \{\text{Measure(LidarSystem, AtmosHumidity), Measure(LidarSystem, Aerosol), Measure(LidarSystem, Wind), Measure(LidarSystem, AtmosTemperature), Control(Measurer, LidarSystem), MeasureValueof(LidarData, Aerosol), MeasureValueof(LidarData, AtmosTemperature), MeasureValueof(LidarData, Wind), MeasureIn(Aerosol, GeoInfo), MeasureIn(Wind, GeoInfo), MeasureValueof(LidarData, AtmosHumidity), MeasureIn(AtmosTemperature, GeoInfo), MeasureIn(AtmosHumidity, GeoInfo), MeasureResultof(LidarData, LidarSystem), MeasureResultof(LidarSystem, LidarData), SubConcept(OtherFactor, GeoInfo), SubConcept(OtherFactor, AtmosPressure)\}$

$A^R_{\text{lidar}}(\text{Measure}) = \{\text{starttime, endtime, address}\}$

$A^R_{\text{lidar}}(\text{Control}) = \{\text{starttime, endtime, address}\}$

$A^R_{\text{lidar}}(\text{MeasureValueof}) = \{\text{accuracy}\}$

$A^R_{\text{lidar}}(\text{MeasureIn}) = \{\text{correlation}\}$

$A^R_{\text{lidar}}(\text{MeasureResultof}) = \{\text{accuracy}\}$

$X_{\text{lidar}} = \{\text{infer(MeasureValueof(LidarData, Aerosol) \\&\\ MeasureResultof(LidarData, LidarSystem)) \rightarrow Measure(LidarSystem, Aerosol), infer(MeasureValueof(LidarData, AtmosTemperature) \\&\\ MeasureResultof(LidarData, LidarSystem)) \rightarrow Measure(LidarSystem, AtmosTemperature), infer(MeasureValueof(LidarData, Wind) \\&\\ MeasureResultof(LidarData, LidarSystem)) \rightarrow Measure(LidarSystem, Wind)\}$
The structured diagram of $O_{\text{lidar}}$ is shown in Figure 2.

The OODM’s definition of atmospheric lidar data ontology $O_{\text{lidar}}$ is based on Definition 2, shown in Box 1. The class $OtherFactor$ is abstracted by the classes of $GeoInfo$ and $AtmosPressure$. The method $GetData$ ($time$: $DateTime$): $DataSet$ returns the data of class $LidarSystem$ or $Aerosol$, and the parameter is the observed times. The $measure$ and $control$ represent that a lidar system can measure amount of data and an engineer who would be able to control lidars, respectively.

Based on a combination of the storage model of OODM (Figure 2) and relation database, we constructed the data storage structure of lidar data ontology $O_{\text{lidar}}$. For the experiments, we compare the new storage model of the OODM with the extendable Native XML and with the vertical mode of the RMDB in different ontology’s instance scale. Experimental setup CPU is Intel(R) Core(TM)2 Duo E7500 2.93 GHz, and the OS is windows server 2003 SP2, and the RMDB is Sql server 2005. The query language of Native XML storage model is XPath, and the query language of OODM and RMDB is sql language. We have investigated three types of query: querying all the instances of a class named $A$; searching all instances of the parent of a class named $B$; querying all the instances named $C$. Taking the limited memory into consideration, four different sizes of atmospheric lidar data ontology instances would be used, $1 \times 10^3$, $2 \times 10^3$, $1 \times 10^4$, and $2 \times 10^4$. Figures 3, 4, and 5 are the comparisons of query times for different data storage models, respectively.

Figures 3, 4, and 5 show that the XML costs the longest query time compared to the other two storage models. Figure 6 shows that increase in the number of instances results in a bigger size of data file and leads to a longer query time, which can be related to the speed of parsing the file and the read-write speed of the disk. Frequent reading and writing files will definitely cost much more time. In addition, the XML method must load the files into the memory for every operation, and thus a large amount of memory usage.
\[ C_s = \{ \text{Aerosol, GeoInfo, AtmosTemperature, Wind, AtmosPressure, AtmosHumidity, Lidarsystem, LidarData, Measurer} \} \]
\[ A_s = \{ \text{lidarratio, extinctioncoefficient, scatteringcoefficient, altitude, latitude, longitude,…} \} \]
\[ D_s = \{ \}
\]
Class AtmosPressure is a OtherFactor type-is
Record
\[
\begin{align*}
\text{altitude}: & \text{Float} \\
\text{latitude}: & \text{Float} \\
\text{longitude}: & \text{Float} \\
\text{gravityacceleration}: & \text{Float} \\
\text{pressure}: & \text{Float}
\end{align*}
\]
End
Class GeoInfo is a OtherFactor type-is
Record
\[
\begin{align*}
\text{altitude}: & \text{Float} \\
\text{latitude}: & \text{Float} \\
\text{longitude}: & \text{Float} \\
\text{gravityacceleration}: & \text{Float} \\
\text{pressure}: & \text{Float}
\end{align*}
\]
End
Class OtherFactor type-is
Record
\[
\begin{align*}
\text{GeoInfo, AtmosPressure}
\end{align*}
\]
End
Class Aerosol type-is
Record
\[
\begin{align*}
\text{lidarratio}: & \text{Float} \\
\text{extinctioncoefficient}: & \text{Float} \\
\text{scatteringcoefficient}: & \text{Float} \\
\text{GetData(time:DateTime):} & \text{DataSet}
\end{align*}
\]
End
Class Lidarsystem type-is
Record
\[
\begin{align*}
\text{scatteringtype}: & \text{String} \\
\text{wavelength}: & \text{Float} \\
\text{singlepulseenergy}: & \text{Float} \\
\text{method}: & \text{String} \\
\text{signalnoiseratio}: & \text{Float} \\
\text{rangeresolution}: & \text{Float} \\
\text{GetData(time:DateTime):} & \text{DataSet} \\
\text{measure}: & \text{Set of LidarData[(1, n)]}
\end{align*}
\]
End
Class LidarData type-is
Record
\[
\begin{align*}
\text{azimuth}: & \text{Float} \\
\text{elevationangle}: & \text{Float} \\
\text{measuringtime}: & \text{DateTime} \\
\text{measuringheight}: & \text{Float} \\
\text{datavalue}: & \text{Float}
\end{align*}
\]
End
Class Measurer type-is
Record
\[
\begin{align*}
\text{name}: & \text{String} \\
\text{age}: & \text{Int} \\
\text{email}: & \text{String} \\
\text{control}: & \text{Set of LidarData[(1, n)]}
\end{align*}
\]
End
Class Measure type-is
Record
\[
\begin{align*}
\text{starttime}: & \text{DateTime} \\
\text{endtime}: & \text{DateTime}
\end{align*}
\]
Box 1: Continued.
Box 1: Definition of \( O_{\text{lidar}} \)'s OODM.

```java
address: String
...
```

7. Conclusions

In this paper, a new ontology storage model is developed for lidar data based on object-oriented model to maximize the query efficiency and minimize the data storage file size by the representation method and mapping mechanism. The model is very useful for lidar data as (1) the lidar data ontology structure can be easily transformed into the OODM structure; (2) the relationship of lidar data ontology can be clearly expressed and stored; and (3) the OODM of lidar data have high expansibility and query efficiency. Some

method can easily obtain the object data due to its simplified data table.

causes an inefficiency operation. In query method type C and ontology with size \( 1 \times 10^4 \) and \( 2 \times 10^4 \), the data records are more than one million, which will exceed the total usable memory size of the testing machine after 49s and 91s, respectively. It reveals that, with the increase of atmospheric lidar data instances of ontology, XML storage model will consume much more time and memory compared to the other methods. Obviously, the OODM and RMDB storage methods are superior to the Native XML method in query time and data file size. Figures 3, 4, and 5 also show that the consuming time of the RMDB method is greater than that of the OODM in all three types of queries. The RMDB method is associated with the data in the form of the tables which joined with each other in the querying process. The OODM

Figure 3: Comparison of type A’s query times.

Figure 4: Comparison of type B’s query times.

Figure 5: Comparison of type C’s query times.

Figure 6: Comparison of data file sizes.
extensions, which are optimized method for processing the
temporal data such as efficient query method and storage
model, are further addressed to consider more practical
factors.

Disclosure

This paper is an extended version of the International Con-
ference on Computer Science and Technology (CST2016).

Competing Interests

The authors declare that there is no conflict of interests
regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science
Foundation of China (NSFC) (Grants nos. 61675166 and
61605157).

References

scientific metadata model to incorporate derived data,” Future

absorption FTIR spectroscopy: characterization of retrievals and
first results from a novel optical design instrument at a new
NDACC complementary station,” Journal of Atmospheric and

over four AERONET sites: an overview,” Atmospheric Environment,

network for monitoring Asian dust and air pollution aerosols,”
in Proceedings of the IEEE International Geoscience and Remote

precipitation variation on drought in China in last 50 years,”

regional characteristics of the temperature changes in China
based on complex network,” Acta Physica Sinica, vol. 58, no. 10,

spectroscopic rotational Raman lidar with visible wavelength
fiber Bragg grating for atmospheric temperature measurement,”

of response metrics for an ontology-based Automated Intrusion

“An ontology-based data integration approach for web analytics
20–34, 2016.

an authentic survey,” Journal of Intelligent Systems, vol. 25, no. 4,
pp. 539–553, 2016.

cient web ontology storage considering hierarchical knowledge
for jena-based applications,” Journal of Information Processing

large ontologies stored in relational databases: the OntoMinD

interface for relational databases,” in Proceedings of the 2nd
International Conference on Intelligent Computing, Communica-
tion and Convergence (ICCC ’16), pp. 487–492, Odisha, India,
January 2016.

object-oriented databases,” Knowledge-Based Systems, vol. 76,


[16] E. Hammann, A. Behrendt, F. Le Mounier, and V. Wulfmeyer,
“Temperature profiling of the atmospheric boundary layer with
rotational Raman lidar during the HD(CP)2 Observational
Prototype Experiment,” Atmospheric Chemistry and Physics, vol.

the atmospheric detection,” Journal of Quantitative Spectroscopy

vertical and horizontal plus cubic spline interpolation for Mie
scattering lidar profile data,” Acta Physica Sinica, vol. 63, no. 15,
Article ID 154204, 2014.

atmospheric lidar data storage model based on ontology,” in
Proceedings of the The International Conference on Computer
Science and Technology (CST ’16), pp. 589–596, Shenzhen,
China, January 2016.

based review of the literature,” Computer-Aided Design, vol. 80,
p. 9–18, 2016.

oriented data model built for blind navigation in outdoor space,”

formal model of the Semantic Web Service Ontology (WSMO),”

ontology storage method research,” Advances in Information

model for effective ontology management in Semantic Web
environment,” Journal of Zhejiang University: Science A, vol. 10,
Submit your manuscripts at https://www.hindawi.com