

Article

Open, Sharable, and Extensible Data Management for the Korea National Aquatic Ecological Monitoring and Assessment Program: A RESTful API-Based Approach

Meilan Jiang ¹, Karpjoo Jeong ^{2,3,*}, Jung-Hwan Park ⁴, Nan-Young Kim ⁴, Soon-Jin Hwang ^{4,*} and Sang-Hun Kim ⁵

¹ Department of Advanced Technology Fusion, Konkuk University, 120 Neungdong-ro, Gwangjin-gu, Seoul 05029, Korea; meela@konkuk.ac.kr

² Department of Internet & Multimedia Engineering, Konkuk University, 120 Neungdong-ro, Gwangjin-gu, Seoul 05029, Korea

³ Institute for Ubiquitous Information Technology and Applications, Konkuk University, 120 Neungdong-ro, Gwangjin-gu, Seoul 05029, Korea

⁴ Department of Environmental Health Science, Konkuk University, 120 Neungdong-ro, Gwangjin-gu, Seoul 05029, Korea; nang12@hotmail.com (J.-H.P.); celeste0@daum.net (N.-Y.K)

⁵ Watershed Ecology Research Team, Water Environment Research Department, National Institute of Environmental Research, Hwangryong-ro 42, Seogu, Incheon 22689, Korea; haemy@korea.kr

* Correspondence: jeongk@konkuk.ac.kr (K.J.); sjhwang@konkuk.ac.kr (S.-J.H.); Tel.: +82-2-450-3510 (K.J.); +82-2-450-3748 (S.-J.H.); Fax: +82-2-452-3410 (K.J.); +82-2-450-5062 (S.-J.H.)

Academic Editors: Young-Seuk Park and Soon-Jin Hwang

Received: 29 January 2016; Accepted: 6 May 2016; Published: 13 May 2016

Abstract: Implemented by a national law, the National Aquatic Ecological Monitoring Program (NAEMP) has been assessing the ecological health status of surface waters, focusing on streams and rivers, in Korea since 2007. The program involves ecological monitoring of multiple aquatic biota such as benthic diatoms, macroinvertebrates, fish, and plants as well as water quality and habitat parameters. Taking advantage of the national scale of long-term aquatic ecological monitoring and the standardization of protocols and methods, the datasets in NAEMP provide many opportunities for various advanced comparative and synthetic studies, policy-making, and ecological management. In order to realize these potentials and opportunities, we have developed a RESTful API-based data management system called OSAEM (the Open, Sharable and Extensible Data Management System for Aquatic Ecological Monitoring), which is designed to be open, sharable, and extensible. In this paper, we introduce the RESTful API-based data management approach, present the RESTful API for the OSAEM system, and discuss its applicability. An OSAEM prototype system is currently available on a commercial cloud service (Amazon EC2) but the system remains under active development.

Keywords: aquatic ecological monitoring; management of ecological data; data sharing; RESTful API

1. Introduction

Implemented by a national law, the National Aquatic Ecological Monitoring Program (NAEMP) aims to assess the ecological health status of lotic environments in Korea by means of biological health indices focusing on benthic diatoms, macroinvertebrates, fish and aquatic plants [1]. Every six months, NAEMP has been conducting ecological monitoring on a national scale at 960 sites (whose number has grown from 540 in the beginning and will reach 3000 by 2018) in major rivers and their tributaries since 2007. Protocols and methods are standardized for monitoring and the same set of those

standardized protocols and methods is used for collecting data at all the sites in NAEMP. The datasets from NAEMP are very different from those produced by individual or group projects because of both the national scale of the long-term aquatic ecological monitoring and the standardization of the protocols and methods. The NAEMP datasets are so standardized that they are highly comparable and easy to integrate. Therefore, the NAEMP datasets provide many opportunities for various advanced comparative and synthetic studies, policy-making, and ecological management, in addition to the ecological health assessment of lotic water.

In order to realize such potentials and opportunities fully, effective management and sharing of the ecological monitoring data is crucial [2–6]. However, the development of a data management system for ecological monitoring is really challenging because not only aquatic ecosystems *per se* but also approaches to ecological studies are very complicated and diverse. Furthermore, the integration of datasets from different ecological data management systems is even more challenging due to the exponential combination of such complexity and diversity [7–9].

In this paper, we present a data management system called OSAEM (the Open, Sharable and Extensible Data Management System for Aquatic Ecological Monitoring) designed to facilitate and promote the sharing of NAEMP datasets. Furthermore, the design of the OSAEM system focuses on enabling a variety of software systems, rather than human users, to access the NAEMP datasets as easily and flexibly as possible. Data management in the OSAEM system is based on the web technology called the RESTful API [10,11]. There are a number of active efforts to apply the RESTful API technology to the management of scientific and engineering data [12–15]. In these efforts, the RESTful APIs are used to support the efficient management of software services. As such, the conventional RESTful APIs are service-oriented.

In this paper, we present a novel, data model-oriented approach to the RESTful API. The data model-oriented RESTful API intends to make the management of ecological monitoring data:

- *Open*. From the OSAEM system, potential users can easily find information about what kind of data is available, what data models are applied to data, and what services are provided for data, without in-depth understanding of the internal design of the data management system. Such information is crucial for ecological analyses and syntheses.
- *Sharable*. New programs (e.g., internal analysis code or external information systems in other projects) can easily be developed to access datasets in the OSAEM system. The interface to data services in the OSAEM system is well defined and easy to use. Such service interface is crucial for integrating datasets from different projects for synthetic analyses.
- *Extensible*. Extensions or changes to the datasets in the OSAEM system are easily and efficiently supported. For example, new categories of living organisms, methods, or sites are easily added for the extension of monitoring and ecological studies.

For the last three decades, there have been active research efforts to develop data management systems for the sharing of ecological monitoring datasets [2,16–21]. In general, there are two kinds of approaches to data sharing: metadata-level sharing and data-level sharing. In the metadata-level data sharing, metadata that satisfies standards such as EML (Ecological Metadata Language) or Darwin Core are created and shared [16,17,22–25]. Such metadata describes basic information (e.g., keywords, locations, investigators, and related documents) about real monitoring data, but usually fails to provide sufficient information about the specific data structures of monitoring data. Therefore, this approach does not allow for the development of software programs (e.g., analysis code) to process monitoring datasets automatically. DataONE is a most notable example of this approach [16,17,21].

In data-level sharing, the data models (e.g., database schemas) for monitoring data are defined and monitoring datasets are organized and stored in data management systems according to the models. Since the data models are available, this approach allows for the development of software programs to analyze datasets automatically. CUAHSI is a most notable example of this approach [18,19].

However, this approach requires us to understand complicated data models, software tools, and service interfaces in such data management systems. For example, the CUASHI relational data model for ecological data called ODM (Observations Data Model) consists of a large number of database tables. In addition, a large number of various data services and software tools are provided and need to be understood for the development of data analysis software [26].

As opposed to these conventional approaches, the RESTful API-based data management approach in the OSAEM system combines data models and service interfaces both logically and physically. This feature makes the RESTful API-based approach really simple and intuitive to understand and to use. Furthermore, this approach does not require special software tools to access the datasets in the OSAEM. Generic software libraries or tools can be used. Therefore, the OSAEM system significantly facilitates the sharing of ecological data.

In addition, there have recently been active research efforts to manage ecological data as linked data in the global ecological research community [27,28]. However, the OSAEM system does not yet support linked data because we believe the ecological research community in Korea currently prefers ecological data in a traditional database style. In spite of the current design, the OSAEM system can be easily extended to support linked data because every data entity in the OSAEM system is represented as a web resource with a unique URI (Uniform Resource Identifier).

In this paper, we do not address issues in the design of databases in the OSAEM system because data management is based on the RESTful API and the design of databases is intentionally made transparent to the user or client software. This paper is structured as follows. Section 2 provides a brief overview of NAEMP. In this section, we discuss the potential opportunities and values of the NAEMP datasets as reference data for various ecological studies and management. In addition, we raise challenging issues about data management for NAEMP. In Section 3, we introduce the RESTful API-based approach to data management. In Section 4, we present the main features of the OSAEM system, focusing on the RESTful API: URIs, representations, and services. The RESTful API is intended for programming, not for the human user. In Section 5, we explain the web portal of the OSAEM to allow the user to access the NAEMP datasets in a GUI-based, user-friendly manner. In Section 6, we conclude this paper with discussions and future work.

2. Brief Overview of NAEMP

NAEMP monitors five major rivers and their tributaries in Korea due to a national law. The program was established in 2007, and since then, the number of sampling sites has grown from 540 to 960, covering the entire nation [1]. The total number of monitoring sites will gradually increase to 3000 by 2018. Figure 1 shows the spatial distribution of monitoring sites in NAEMP. All the datasets collected with the same standardized protocols and methods are comparatively and synthetically analyzed in order to assess the ecological health status of the national lotic ecosystem by means of developed biological health indices including benthic diatoms, macroinvertebrates, fish, and aquatic plants [29–34].

Every six months, NAEMP monitors the physio-chemical and biological parameters shown in Table 1. However, biological and habitat parameters (currently fish, benthic diatoms, and macroinvertebrates) are the main criteria used to assess the ecological health of the rivers and streams. At each monitoring site, the same set of parameters is measured with the same set of standardized protocols and methods during almost the same period of time. The locations of monitoring sites are chosen in order to allow the datasets to be scientifically representative of lotic ecosystems in Korea. The monitoring results at each site are quality-controlled during the lab analyses. Some parameters are observed and recorded at the sites on standardized forms (*i.e.*, field survey forms).

The lab analysis results and field survey paper sheets are sent to the NAEMP central data management team and organized into standardized spreadsheet files. The NAEMP data management team manages those spreadsheet files in a centralized manner. Figure 2 shows the data collection process for NAEMP.

Due to the standardized protocols and methods applied to every monitoring site, the datasets in NAEMP have great potential and value as reference datasets for various ecological studies and management. However, NAEMP currently lacks effective dataset management and data services for other ecological studies. Currently, the datasets are managed as spreadsheet files (*i.e.*, Microsoft Office Excel files) in NAEMP. Excel Macro programs embedded in the dataset files are used to generate biological health indices. Such spreadsheet file-based management of datasets is very simple, easy, and flexible, but inevitably *ad hoc* and error-prone. With this data management approach, it is very difficult to search for or locate data items of interest in the datasets, to share only some parts of the datasets (e.g., excepting sensitive or private data) with other users or information systems outside the NAEMP, and to support extensions to ecological monitoring such as the addition of new living organisms, sites, or methods.

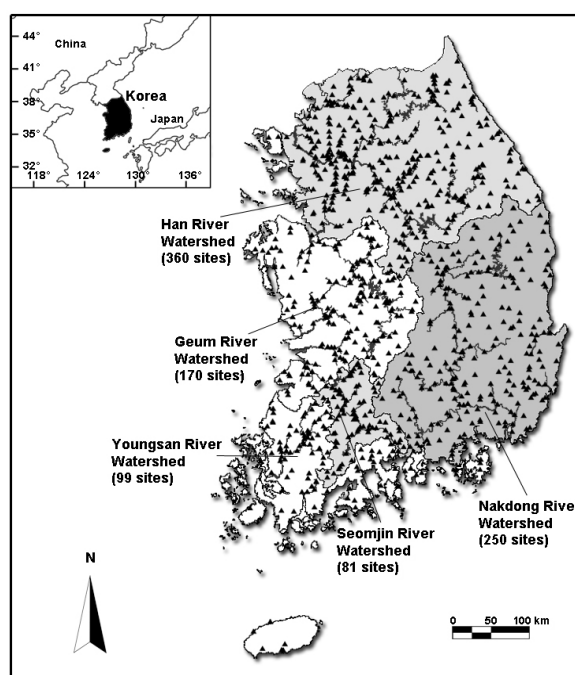


Figure 1. The monitoring sites in the five major rivers for the National Aquatic Ecological Monitoring Program in Korea. The triangles indicate 960 sites that are currently monitored.

Table 1. The parameters measured for fish.

Category	Parameter	Type	Unit	Method
Site Information	Water Basin	Text		
	River	Text		
	GPS/Address	Text		
	Stream Type	Text		
	Sampling Method	Text		
	Weather	Text		Observe
Biological Factor	Species	Text		
	Number of Species	Numeric Data		Observe
	Tolerance Guild	Text		
	Trophic Guild	Text		
	Habitat Guild	Text		
	Protected Species	Text		
	Exotic Species	Text		
	Abnormal Individuals	Text		
	Number of Abnormal individuals	Numeric Data		Observe

Table 1. Cont.

Category	Parameter	Type	Unit	Method
Environmental Factor	Substrata Structure	Numeric Data	%	Observe
	Type of Water Flows	Numeric Data	%	Observe
	Canopy	Numeric Data	%	Observe
	Vegetation Cover	Numeric Data	%	Observe
	Odor	Text		Observe
	Plant Structure	Numeric Data	%	Observe
	Land Use	Numeric Data	%	Observe
	Depth	Numeric Data	cm	Measure
	Stream Width	Numeric Data	m	Measure
	Water Velocity	Numeric Data	cm/sec	Measure
	Water Temperature	Numeric Data	°C	Measure
	Conductivity	Numeric Data	μS/cm	Measure
	Turbidity	Numeric Data	NTU	Measure
	pH	Numeric Data		Measure
	DO	Numeric Data	mg/L	Measure

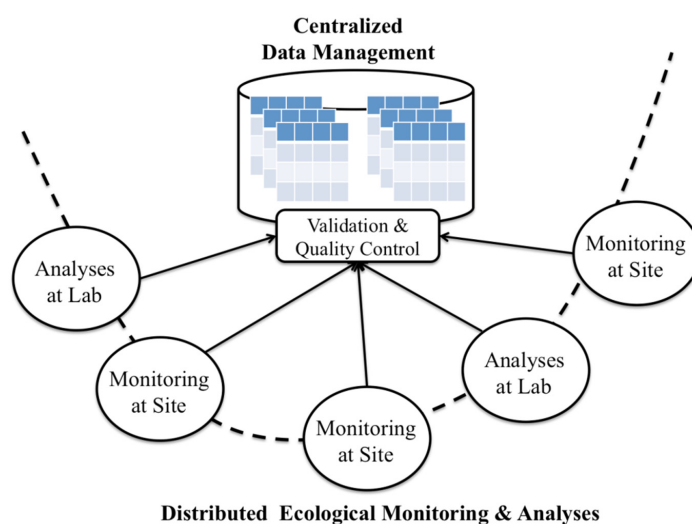


Figure 2. NAEMP data collection and management structure.

NAEMP must address serious challenges with respect to data management, in addition to converting the datasets in spreadsheet files to well-defined tables in a database system. First, there are no universally acceptable standards for such a national-scale, index-based ecological health assessment. Therefore, NAEMP has actually been developing effective health indices over the years and may need to change them for optimization (e.g., adding new living target organisms and parameters, changing methods, or adding new sites) in the future. These changes generally require substantial redesign of data models, data storage, and data services. However, the amount of such redesign on the other components of the information system (e.g., analyses, syntheses, or reporting) must be minimized.

Second, taking into consideration the fact that NAEMP continues to evolve or advance in terms of not only the total size of the data but also science and technology, there will be great potential or requirements for new analyses and syntheses for the assessment of ecological health. Such new analyses or syntheses must be easily testable, addable and supportable.

Finally, if the datasets in NAEMP are used as a kind of reference data for other aquatic ecological studies in Korea, other users or information systems for other projects outside NAEMP will need to access the datasets in NAEMP. For example, there may be new synthetic analyses to integrate data from NAEMP and those from other projects. Such data integration must be supported in an easy and efficient manner.

The OSAEM data management system is developed to address these challenging issues. For the rest of this paper, we explain the main features of the OSAEM system, focusing on the RESTful API and the web portal.

3. RESTful API-based Approach to Data Management

3.1. RESTful API

In this section, we briefly introduce REST (Representational State Transfer), a set of architectural and design styles for software services on the World Wide Web, and then present a novel data model-oriented RESTful API for data management [9]. The REST styles are based on technical concepts such as web resources, URIs (Uniform Resource Identifier), representations and services. Software services available on the Web that satisfy these REST principles are called RESTful web services. The set of resources, URIs (e.g., <https://en.wikipedia.org/wiki/Fish>), representations, and RESTful web services is called a RESTful API [10].

More specifically, major REST principles relevant to data management are summarized as follows:

- *Resource-oriented*. Everything that needs to be referenced or managed is treated as a logical entity called a *web resource* (hereafter, just *resource*). A resource can be either a real document (e.g., an HTML file or an image file) or a logical entity associated with software services.
- *Universally addressable*. Every resource that is either a real document or a logical entity has a URI assumed to be unique on the Internet. This URI allows the user or the application software to refer and access the resource from any computer on the Internet.
- *Representation-based*. Every resource including a logical entity associated with a software *service* is presented and accessed as if it were a document (or a data object). The document dynamically created by the associated software service for the logical entity is called a representation. In REST, the data structure and format of the representation for each logical entity must be well defined and available to the public. The representation of a resource can be text data, binary data (*i.e.*, data understandable only to particular machines or software), an image, an audio file, and a video. In REST, the most widely used syntax for representations is JSON (JavaScript Object Notation) [35].
- *Serviceable*. The RESTful API assumes the HTTP protocol for communication [36]. In the HTTP protocol, the URI for a resource can be considered to be the Internet address of software services associated with the resource. In the HTTP protocol, four types of software services can be assigned to each URI: POST, GET, PUT, and DELETE. The POST, GET, PUT, and DELETE types are intended for the Create, Read, Update, and Delete operations, respectively.

There are also some other architectural principles but since they are system-specific, they are not covered in this paper which is intended to address data management issues. Please refer to Roy Fielding's article for a complete explanation of REST principles [10].

We present Algorithm 1 and Figure 3 to illustrate how a RESTful web service works. Algorithm 1 shows an example of a representation in JSON. A data object in the JSON format is basically a list of key and value pairs enclosed by curly braces. A key plays the role of a data field in a relational database. A value can be a number, a Boolean value, a text string, another JSON object, or an array of JSON objects. Arrays of JSON objects are enclosed by square brackets. Having a JSON object or an array of JSON objects as a value allows JSON to support a hierarchical data model. In Algorithm 1, the JSON data object consists of two key-value pairs where the keys are "Name" and "Job". The value for the key "Name" is another JSON object.

```

{
  "Name": {
    "firstName": "John",
    "lastName": "Smith"
  },
  "Job": "Teacher"
}

```

Algorithm 1. A representation in JSON.

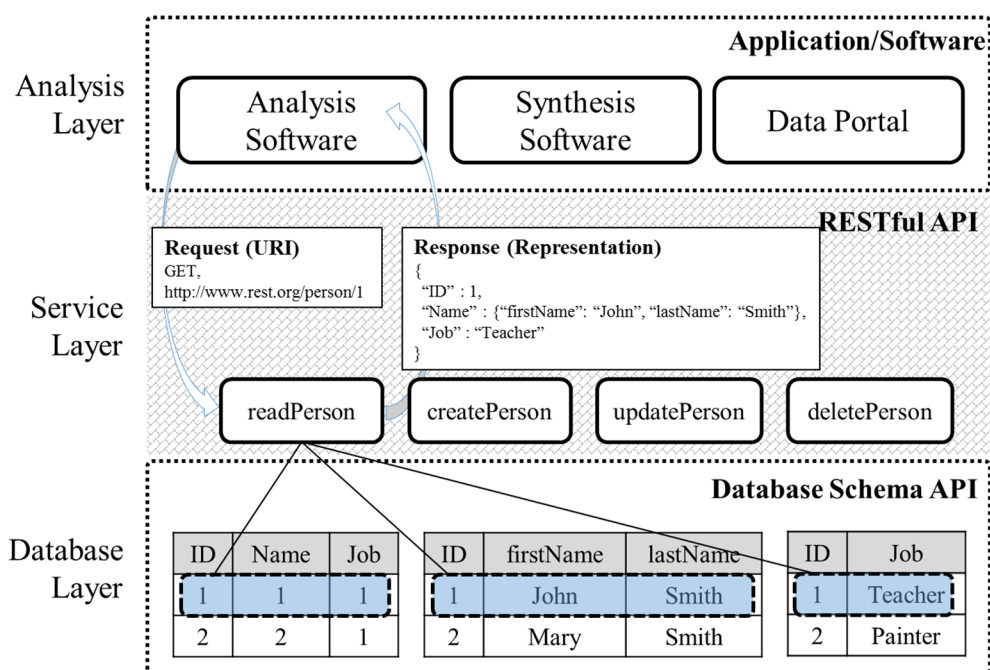


Figure 3. Web architecture for RESTful web services.

Figure 3 shows how a RESTful web service works on the Internet. In this example, a logical data entity or document (called a web resource, or just a resource in REST) is assumed to exist for a certain person on a web server where the entity has a URL (Uniform Resource Locator) or URI (Uniform Resource Identifier): <http://www.rest.org/person/1>. In fact, the logical entity does not exist physically on the server. However, if the URI of the logical entity is referenced on a web browser (e.g., entered into the URL bar window), a new real document is dynamically created from the database and returned by the web server as if the logical entity were a real document. The dynamically created real document is called a representation in REST.

In this example, instead of simply returning a real document file in the file system for the URI, the web server invokes the readPerson software service associated with the URI, then readPerson generates a document dynamically, and finally the web server returns the newly created document. In fact, readPerson obtains and merges three data records from three DB (Database) tables (one record from each DB table, respectively) into a new document (called a representation in REST) for the URI. The syntax for data in this example will be explained in Section 4. In REST, software services, therefore, appear as if they were documents. Such software services on web servers are called RESTful web services (hereafter, just web services).

3.2. Data Model-Oriented RESTful API

Conventionally, the REST principles and APIs are used to model and define the interfaces for web services, although they have great potential for other kinds of modeling. In this paper, we propose

a novel approach to the RESTful API where the REST principles are applied to the design of data models for data management. The approach is called the *data model-oriented RESTful API* where services are not explicitly defined but implicitly assumed.

The *data model-oriented RESTful API* is designed to support the following resource model:

- *Resource classification.* Every resource is classified as either an instance resource or a collection resource.
- *Instance resource.* Each individual resource is considered an instance resource. Each instance resource has a unique URI and is accessed as a representation.
- *Collection resource.* A collection resource represents the set of all the instance resources of the same type. Each collection resource also has a unique URI.
- *Representation schema.* The representation schema is defined to be the data model or structure for the representation of a resource. The representations of all the instance resources in a certain collection resource must be based on the same representation schema. Syntactically, the representation schema is defined to be a list of key and data-type pairs enclosed by angle braces. However, the representation schema (created only as a document) is currently intended to help the user to understand the data models for resources, but is not used in system implementation.
- *Implicit CRUD (Create, Read, Update and Delete) services.* According to the HTTP protocol, each resource (its URI) is assumed to be associated with four types of services: Create, Read, Update and Delete. The Read service for a collection resource is designed to be a database query. The HTTP-based communication allows each resource to be independently accessed by those services.

This resource model has several advantages over other conventional RESTful APIs and data management systems.

- First, the resource model makes the RESTful API simple and intuitive to understand and use. Although the REST principles allow a great amount of generality and flexibility, they effectively do not enforce simplicity and consistency. This resource model combines and unifies data models and services logically and physically into only URIs and representations. The user can assume the HTTP CRUD services for each resource.
- Second, the resource model makes the RESTful API look like a database system. In fact, the resource model is logically similar to the database model. The instance and collection resource are analogous to the database record and tables, respectively. The representation schema is similar to the database schema. The URI for a resource can be considered to be the key for a database record. Therefore, this resource model can substantially help the software developer to understand the design of data management and to develop analysis software.
- Third, the resource model is managed independently of underlying database schemas that are usually much more complicated than the resource model. Figure 3 illustrates the independence between the resource model and underlying database schemas. Although the resource model looks like a database model, it is in fact the model for service interfaces (*i.e.*, RESTful APIs). Both the resource model and the underlying database design can be changed without affecting the other. This feature facilitates the support for hosting and portability significantly.
- Finally, if the resource model is publicly open and available to any user or information systems, no special custom-built software is required to access datasets. There are generic HTTP protocol-based client libraries widely available that can be used to invoke software services associated with the resources. Therefore, when any research group wants to develop analysis software, they do not need any special software from the NAEMP project and can easily develop their own analysis software only with publicly available generic HTTP protocol-based libraries

Because of these advantages, the *data model-oriented RESTful API* can effectively address the challenging issues in the sharing of ecological monitoring data discussed in Sections 1 and 2.

4. OSAEM: The Open, Sharable, and Extensible Data Management System for Aquatic Ecological Monitoring

In this section, we present a data management system for NAEMP datasets called OSAEM (the Open, Sharable, and Extensible Data Management System for Aquatic Ecological Monitoring) that is designed to support the data model-oriented RESTful API in Section 3.

4.1. Resource Model

First, major data components of the NAEMP are modeled and managed as instance and collection resources:

- *Living organisms.* Each individual species of fish, benthic diatoms and macroinvertebrates is modeled as a resource. For example, fish species such as “*Lethenteron camtschaticum* Tilesius” and “*Cyprinus carpio* Linnaeus” are treated as separate resources in OSAEM. Since 150 fish species are currently monitored in NAEMP, there are 150 resources in OSAEM. Currently, the OSAEM system does not support datasets for benthic diatoms and macroinvertebrates.
- *Sites and rivers.* The individual sites where ecological monitoring is carried out are modeled as resources. The rivers are also handled as resources. In NAEMP, each of the 960 sites is currently considered a separate resource.
- *Parameters and methods for observation.* The individual parameters are treated as resources. Therefore, this design requires all observation parameters to be explicitly defined and registered. Similarly, the individual measurement and analysis methods are managed as resources.
- *Observation data.* Each individual instance of observation is modeled as a separate resource. For example, if the water temperature at a certain site is measured 10 times, the 10 measurement results are considered separate resources.

The instance and collection resources are managed as follows:

- *Instance resources.* A single data entity (e.g., an individual fish species or an individual site) is managed as an instance resource. An instance resource is logically analogous to a database record in relational databases.
- *Collection resources.* All instance resources of the same type are managed as one collection resource. For example, the Fish collection resource is defined as the set of all instance resources for fish species. Currently, there are eight collection resources in OSAEM: Fish, Site, River, Variable, Method, Source, Unit and Observation. Collection resources are logically analogous to database tables in relational database systems.

The organization of resources is illustrated in Figure 4 where two collection resources (Fish and Site) and their instance resources are shown. In the example, URI_BASE in URIs denotes the common prefix for every URI in OSAEM and is explained in Section 4.2. URI_BASE/fishes and URI_BASE/sites are the URIs of the Fish and Site collection resources, respectively. URI_BASE/fishes/1 and URI_BASE/sites/1 are the URIs of instance resources for specific fish species and site, respectively. The URI patterns are also explained in Section 4.2.

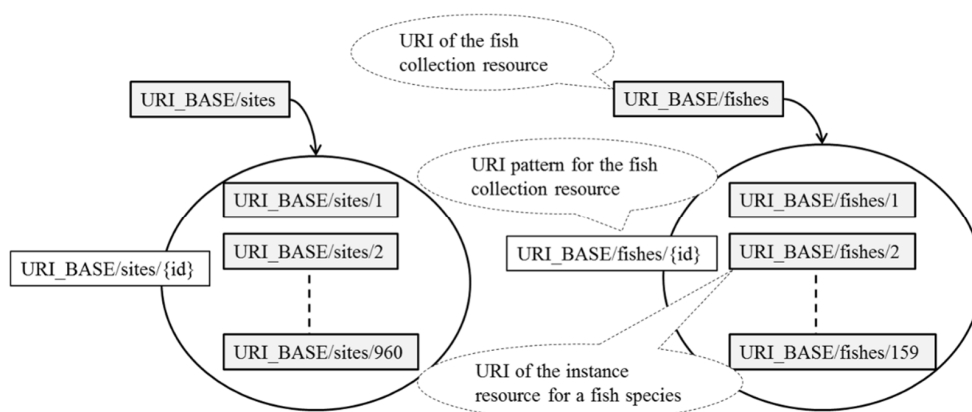


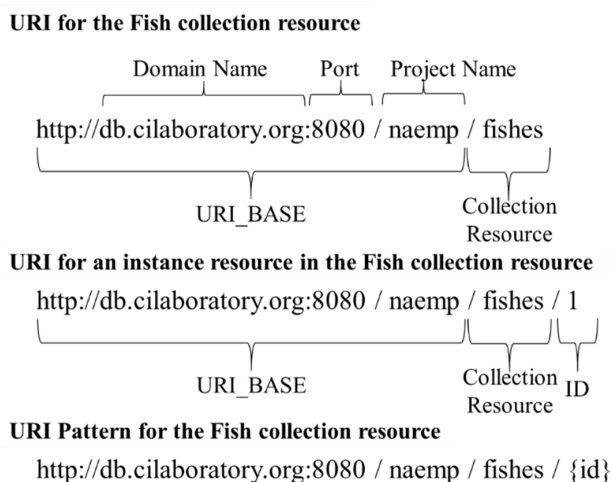
Figure 4. Organization of instance and collection resources. URI_BASE is the common prefix for every URI in OSAEM and is explained in Section 4.2. URI_BASE = <http://db.cilaboratory.org:8080/naemp>.

4.2. URI Scheme

In OSAEM, we support a simple scheme for assigning URIs to the instance and collection resources. This URI scheme intends to make the management of URIs intuitive and predictable in spite of a large number of instance resources. The URI scheme consists of:

- URI structure.
- URIs and URI patterns for the collection resources.
- System-assigned numeric IDs for the instance resources.

First, the structure of every URI consists of three components in OSAEM: the URI base, the name of the collection resource, and, optionally, the system-assigned resource ID only for an instance resource. The URI base is the common prefix for every URI in OSAEM. The URI base is composed of the Internet network address of the web server (e.g., <http://db.cilaboratory.org:8080/>) and a web application name (currently, just/naemp) added to indicate the NAEMP project. Scheme 1 illustrates an example of the URI structure. Currently, the OSAEM prototype system is on a web server whose Internet network address is <http://db.cilaboratory.org:8080>, but the address will be changed to a permanent web server later. For the rest of this paper, we use the symbolic keyword URI_BASE in every URI to denote the URI base.



Scheme 1. URIs and URI pattern.

Second, the URI for every instance resource in the same collection resource is based on the same pattern called the URI pattern. Therefore, each collection resource is associated with the URI pattern shared by the instance resources. The URI pattern for a collection resource consists of the URI for the collection resource and a variable (*i.e.*, a placeholder) for a system-assigned numeric ID for a specific instance resource. In Scheme 1, the URI pattern for the Fish collection resource is `URI_BASE/fishes/{id}`, where `{id}` denotes a numeric ID for an instance resource. The URI for an instance resource (*i.e.*, a fish species) in the Fish collection resource is `URI_BASE/fishes/1`.

Together with their URIs and URI patterns, all eight collection resources are listed in Table 2.

Table 2. URIs and URI patterns for the collection resources.

Collection Resources	URI	URI Pattern	Number of Instance Resources
Fish	<code>URI_BASE/fishes</code>	<code>URI_BASE/fishes/{id}</code>	159
Site	<code>URI_BASE/sites</code>	<code>URI_BASE/sites/{id}</code>	960
River	<code>URI_BASE/rivers</code>	<code>URI_BASE/rivers/{id}</code>	760
Variable	<code>URI_BASE/variables</code>	<code>URI_BASE/variables/{id}</code>	12
Unit	<code>URI_BASE/units</code>	<code>URI_BASE/units/{id}</code>	11
Method	<code>URI_BASE/methods</code>	<code>URI_BASE/methods/{id}</code>	5
Source	<code>URI_BASE/sources</code>	<code>URI_BASE/sources/{id}</code>	17
Observation	<code>URI_BASE/values</code>	<code>URI_BASE/values/{id}</code>	139567

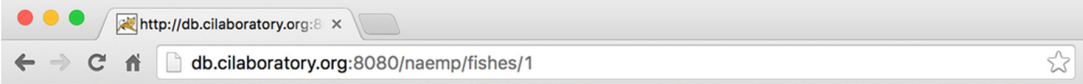
4.3. Representation Schema

In OSAEM, a resource (either a real document file or a logical entity associated with a software service) is handled as a data object (or a document) at run time. Therefore, every resource must be defined and handled as if it were a real document. For example, if a human user enters the URI of the resource for the fish species Arctic lamprey (in fact, `URI_BASE/fishes/1`) into the URL bar of a web browser, then the user will receive a document (*i.e.*, data) for the resource. Figure 5 gives a snapshot of a web browser window displaying the data for the fish species Arctic lamprey. Web browsers do not print the data in a readable format with proper indentations and line breaks; the same data is well formatted in Figure 6. Such a formatting service (*e.g.*, <http://codebeautify.org/jsonviewer>) is available online.

The realization of a resource as a document or data object is called a representation. In RESTful APIs, resource representations can be based on various formats: JSON, XML, HTML, or multimedia formats (*e.g.*, PNG, JPEG, MP3, and MP4). In OSAEM, JSON is used as the main data format for representations [35]. JSON is a simple text format in the JavaScript programming language but is widely used for a variety of Internet services.

In OSAEM, the representation of every instance resource in the same collection resource (*e.g.*, the Fish collection resource) is based on the same data model or structure called the *representation schema*, whose name was chosen to show its resemblance to the database schema. The representation schema facilitates the management of representations. In OSAEM, the representation schema is similar to the JSON syntax but contains data types (enclosed by a pair of angled brackets) instead of values.

In OSAEM, there are 14 representation schemas. Appendix A shows the list of all the representation schemas. Algorithm 2 shows the representation schema for the Fish collection resource. In fact, the representation schema simply shows the keys and their data types. Figure 5 shows the representation for the fish species Arctic lamprey based on the representation schema.



```
{
  "family": "철성장어과",
  "order": "철성장어목",
  "invasiveSpecies": "",
  "trohicGuild": {
    "term": "O",
    "definition": "잡식 중"
  },
  "habitatGuild": {
    "term": "-",
    "definition": " "
  },
  "toleranceGuild": {
    "term": "IS",
    "definition": "중간 중"
  },
  "fishID": 1,
  "fishClass": "두갑강",
  "scientificName": "Lethenteron camtschaticum Tilesius",
  "species": "철성장어",
  "endemicSpecies": " ",
  "endangeredSpecies": "멸종위기 2급",
  "naturalMonument": null,
  "imageLink": null,
  "feature": {
    "featureID": 1,
    "featureName": "철성장어",
    "featureType": "Fish",
    "description": null
  }
}
```

Figure 5. Snapshot of a web browser displaying the representation data for Arctic lamprey.

Result : Beautify Json

```
1 {
2   "family": "철 성장 어 과",
3   "order": "철 성장 어 목",
4   "invasiveSpecies": "",
5   "trohicGuild": {
6     "term": "O",
7     "definition": "잡 식 중"
8   },
9   "habitatGuild": {
10    "term": "-",
11    "definition": " "
12  },
13  "toleranceGuild": {
14    "term": "IS",
15    "definition": "중 간 중"
16  },
17  "fishID": 1,
18  "fishClass": "두 갑 강",
19  "scientificName": "Lethenteron camtschaticum Tilesius",
20  "species": "철 성장 어",
21  "endemicSpecies": " ",
22  "endangeredSpecies": "멸 종 위 기 2 급",
23  "naturalMonument": null,
24  "imageLink": null,
25  "feature": {
26    "featureID": 1,
27    "featureName": "철 성장 어",
28    "featureType": "Fish"
29  },
30  "description": null
31 }
```

Figure 6. Well-formatted representation data for the fish species Arctic lamprey generated by an online formatting service (<http://codebeautify.org/jsonviewer>).

```
{
  "fishID": <integer>,
  "class": <string>,
  "order": <string>,
  "family": <string>,
  "species": <string>,
  "scientificName": <string>,
  "toleranceGuild": <string>,
  "trohicGuild": <string>,
  "habitatGuild": <string>,
  "invasiveSpecies": <string>,
  "endemicSpecies": <string>,
  "endangeredSpecies": <string>,
  "naturalMonument": <string>,
  "imageLink": <string>,
  "description": <string>
}
```

Algorithm 2. Representation schema for the Fish collection resource.

The representation of a collection resource is simply the entire list of the representations of all its instance resources. All the representations in the list must conform to the representation schema for the collection resource. Algorithm 3 shows a representation (many details are omitted) for the Fish collection resource that a user can easily retrieve by entering its URI (URI_BASE/fishes) into the URL bar of any web browser. Thus, it is really simple for a user to determine which instance resources exist in a certain collection resource, as long as the user knows the URI for the collection resource. However, the OSAEM does not allow the user or other software to retrieve the representation of the Observation collection resource because the number of instance resources in the collection resource is very large.

Algorithm 4 shows the representation schema for the Observation collection resource. An instance resource of the Observation collection resource contains the data collected from a single monitoring activity. The data includes a value (*i.e.*, what value was collected), a site ID (*i.e.*, where monitoring occurred), an entity ID (*i.e.*, what species was monitored), a variable ID (*i.e.*, what parameter was measured), and a method ID (*i.e.*, what method was used). Each of these (site ID, entity ID, variable ID, and method ID) references an instance resource in one of the Site, Fish, Variable, and Method collection resources, respectively. The real URI for each instance resource can be automatically generated with such a system ID and the corresponding URI pattern.

```
[
  { "fishID": 1, "scientificName": "Lethenteron camtschaticum Tilesius" ... },
  { "fishID": 2, "scientificName": "Lethenteron reissneri Dybowski" ... },
  { "fishID": 3, "scientificName": "Anguilla japonica Temminck and Schlegel" ... }, ...
  ...
]
```

Algorithm 3. Representation for the Fish collection resource.

```
{
  "valueID": <integer>,
  "dateTime": <string>,
  "surveyYear": <integer>,
  "surveyTerm": <integer>,
  "dataValue": <double>,
  "site": <integer (siteID)>,
  "latitude": <string>,
  "longitude": <string>,
  "source": <integer (sourceID)>,
  "entity": <integer (speciesID)>,
  "variable": <integer (variableID)>,
  "method": <integer (methodID)>
}
```

Algorithm 4. Representation schema for the Observation collection resource.

4.4. RESTful Web Services

In OSAEM, a resource is serviceable because its URI is associated with software services. That is, the URI of each resource is used as the Internet network address for those software services. In OSAEM, the following software services are provided:

- *Four CRUD services* (explained in Section 3.2) for every instance resource. The CRUD services include the Create, Read, Update and Delete services. They are used to manage instance resources.
- *Query services* for collection resources. Query services are used to search instance resources of interest from collection resources.

4.4.1. CRUD Services for the Instance Resources

In OSAEM, every instance resource (*i.e.*, each URI) is assumed to have a set of four CRUD software services. Therefore, when instance resources must be managed (e.g., create, read, update, or delete them), these services can be invoked. However, the collection resources do not have their own CRUD services, with the exception of the Read operation, because management of the collection resources is really critical and is supported only through a separate admin tool available to the human system administrator.

In the system implementation, a set of CRUD services is in fact assigned to the URI pattern for each collection resource, instead of the URIs of individual instance resources, because the CRUD services are the same for every instance resource in the same collection resource. In other words, every individual resource in a collection resource shares the same set of CRUD services in the OSAEM system.

As explained in Section 3.2, software services associated with URIs are invoked according to the HTTP protocol, which is the URI-based request-response communication model. In the protocol, a client sends a URI-targeted request message to a server; then the server carries out the request on the URI and returns the result as a response to the client. In the protocol, there are four major request types: POST, GET, PUT, and DELETE.

Figure 7 illustrates how CRUD services are associated with the resources for Fish and invoked at runtime:

- First, the four CRUD services are associated with the URI pattern for the Fish collection resource: `URI_BASE/fishes/{id}`. The CRUD services are `createFishSpecies`, `readFishSpecies`, `updateFishSpecies`, and `DeleteFishSpecies`.
- Second, a client system generates CRUD service request messages for a certain fish species according to the HTTP protocol and sends them to the web server. The request message contains three components: the request type (*i.e.*, POST, GET, PUT, or DELETE for the create, read, update or delete operations, respectively), the URI for the target resource, the header information for attributes, and, optionally, the message body (usually a resource representation). In this example, for the fish species Arctic lamprey, whose URI is `URI_BASE/fishes/1`, HTTP requests are shown. The GET type of request message consists of the string text for the request type (*i.e.*, “GET”) and the URI for Arctic lamprey. However, the POST type of request message contains a special URI (that is `URI_BASE/fishes/new`) instead of `URI_BASE/fishes/1` because the URI does not exist at the time of the creation request.
- Third, when it receives a HTTP request message for a URI, the web server invokes a software service (in fact, a RESTful web service) according to the request type.
- Finally, the web service (*i.e.*, `createFishSpecies` for the POST request and `readFishSpecies` for the GET request) performs the request operation and generates a response message as a result. The web server then returns the result to the client system. For the GET type of request message, the response contains the representation of the target resource (*i.e.*, Arctic lamprey).

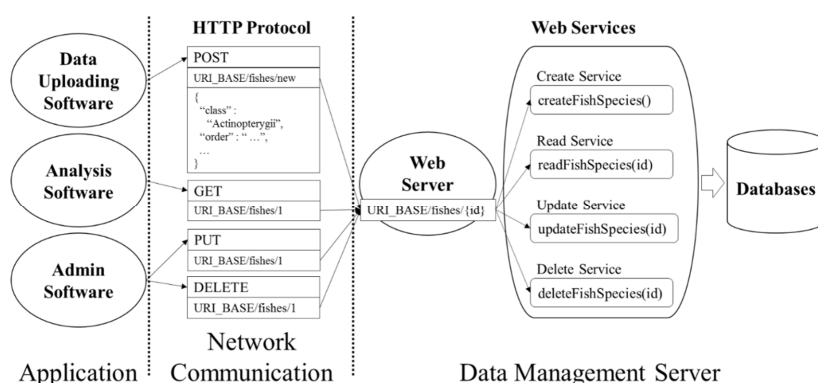


Figure 7. CRUD services for the fish instance resource.

4.4.2. Query Services for the Collection Resources

In addition to the CRUD services for the individual management of each instance resource, the OSAEM system also supports query services to search for instance resources of interest through collection resources. The OSAEM system does not support an official query language but, rather, a set of query services based on the representation schema. These query services are associated with the URIs of the collection resources because a query is a collection resource-level operation.

The logical model for a query is defined as follows:

- A query specifies how to select instance resources of interest from a target collection resource.
- Syntactically, a query is logically defined to be a series of query conditions on a representation schema where each query condition consists of a key and a list of matchable values for the key (in fact, a JSON array object). Therefore, a query itself is a JSON data object.
- The query result is a list of representations that match the query. The result is generated as a JSON array.

Figure 8 illustrates the logical model for the query in OSAEM. In the figure, the representation schema for the Site collection resource is shown on the left. Two queries about the collection resource are given on the right. The top query consists of only one query condition whose key is “siteName” and whose matchable values are the list of two site names. The bottom query is composed of two query conditions and those sites whose representations satisfy both queries are selected.

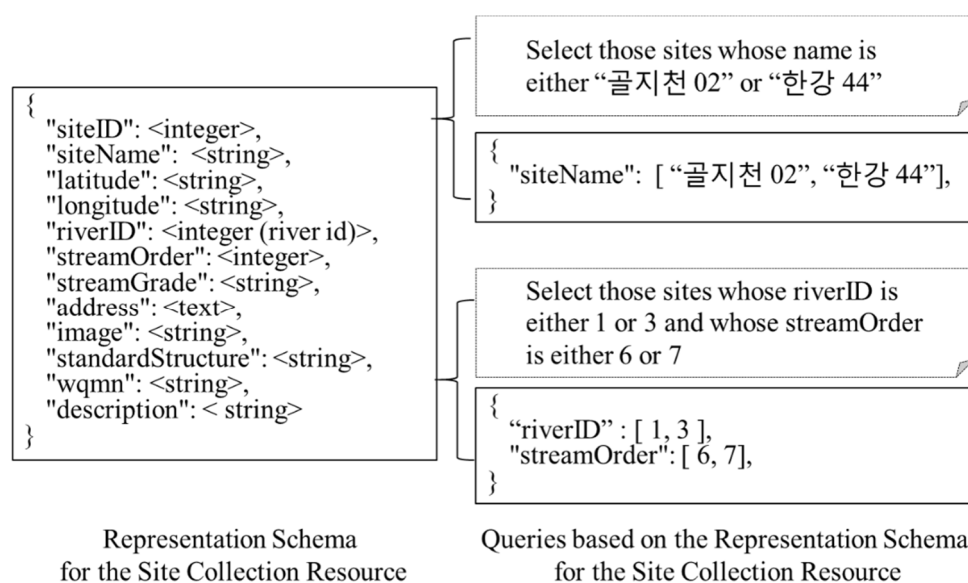


Figure 8. Logical model for the query in OSAEM.

The query model is supported in two ways:

- *GET requests with URIs.* In the HTTP protocol, a URI can have a number of additional query conditions. In OSAEM (in fact, in conventional RESTful web services), each query condition consists of a key and a value (any string).
- *POST requests with queries in JSON.* A query in JSON is directly submitted as a POST request. Queries with complicated query conditions are difficult to express in URIs. In this case, POST requests with queries in JSON are used instead of GET requests with URIs.

First, queries based on the GET request take advantage of the query feature of the URI syntax in the HTTP protocol. The HTTP protocol allows the URI to consist of the address and the query

parts. The question mark “?” is used as the delimiter to separate the address and query parts in the URI. The query part consists of a list of query conditions separated by the ampersand sign “&”. Each query condition is a pair of a query parameter (in fact, a key in the OSAEM query model) and a value combined with the equal sign “=”.

Algorithm 5 shows both an example of a query expressed as a URI for the Site collection resource and the HTTP GET request message with the URI. The query URI consists of the address part (*i.e.*, URI_BASE/sites) and the query part (“?riverID=1&riverID=3&streamOrder=6&streamOrder=7”). The query part consists of two query conditions: “riverID=1, riverID=3” and “streamOrder=6, and streamOrder = 7”. The riverID and streamOrder parameters are keys of the representation schema for the Site collection resource and the repeated parameters are recognized as an array for each parameter.

http://db.cilaboratory.org/naemp/sites?riverID=1&riverID=3&streamOrder=6&streamOrder=7
GET /naemp/sites? riverID=1&riverID=3&streamOrder=6&streamOrder=7
HTTP/1.1
Host: db.cilaboratory.org:8080

Algorithm 5. GET request with URIs as a query for the Site collection resource.

In comparison with queries based on the GET request, queries based on the POST request are simple to create, but cannot be used in a web browser. They require the HTTP software library to invoke queries. A query (in fact, a JSON data object) is simply included as the message body of the POST request message. The use of the POST request for querying is not exactly consistent with the conventional semantics (*i.e.*, the creation of a resource) for the POST operation recommended in the HTTP protocol. However, since it is strongly recommended that the GET request not have a message body, the POST request is often used for complicated queries.

4.4.3. Query Services for the Observation Collection Resource

In this section, we explain the query services for the Observation collection resource in detail because those queries are the most important for the ecological studies. Therefore, the collection resource requires a kind of relational database join operation to compare and merge data from multiple collection resources. However, the other collection resources in the project datasets have no direct relationships with each other.

Figure 9 shows the relationships between the Observation and other collection resources. For example, the join operation is needed for a query such as ‘select observation data that is collected at the site whose name is “골치천 02”’. In the current design of the OSAEM system, such a join operation is not fully supported yet. However, such a join operation can be easily programmed in OSAEM. Currently, only some primitive join operations are available for the Observation collection resource:

- The query services for every Collection resource are extended to enable the join mode. The join mode is enabled only for the Observation collection resource. That is, no new join operation is added.
- When the join operation is enabled, the representation of an instance resource in the Observation collection resource is automatically expanded to replace the system IDs for instance resources in the other collection resources such as Site and Fish and become their real representations.

Algorithm 6 illustrates the join mode for the Observation collection resource. The only difference between queries with and without the join mode is whether the query parameter “join” is added and set to “on”. The algorithm shows the result of a query with the join mode enabled. If the join mode is disabled in the query, then the representation shown in Figure 9 is retrieved.

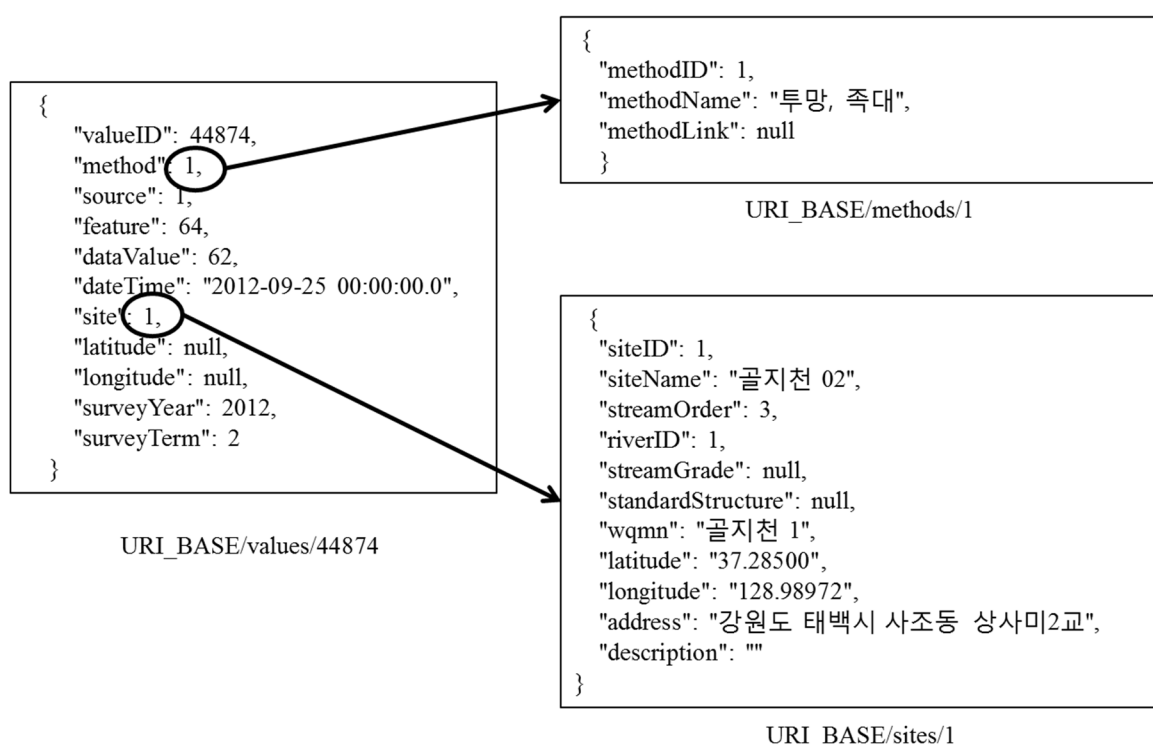


Figure 9. Relationships between the Observation and other collection resources.

```

URI_BASE/values?valueID=44874&join=on
{
  "valueID": 44874,
  "method": {
    "methodID": 1,
    "methodName": "투망, 족대",
    "methodLink": null
  },
  ...
  "dataValue": 62,
  "dateTime": "2012-09-25 00:00:00.0",
  "site": {
    "siteID": 1,
    "siteName": "골지천 02",
    ...
  },
  ...
}
  
```

Algorithm 6. Query with the join mode enabled: URI_BASE/values?valueID=44874&join=on.

5. Web Portal

In OSAEM, the RESTful API is intended for software (e.g., programming synthetic analysis code), not for the human user, although the user can test most of the services with only a web browser. The API will be used by not only a variety of analysis, visualization, and reporting software but also by information systems in other projects for synthetic research. For the end user, OSAEM provides a web portal, a user-friendly GUI system based on the RESTful API. The main goal of the OSAEM web portal is to allow the user to access NAEMP datasets through an intuitive and convenient GUI, instead of the programming-oriented RESTful API. However, the OSAEM web portal uses the RESTful API explained in Section 4.

Currently, the OSAEM web portal consists of two major components:

- GUI services to manage the project datasets: the Fish, Site, River, Variable, Method, and Source collection resources
- GUI services to search through the observation datasets (*i.e.*, only the Observation collection resource).

5.1. CRUD Services and Query Services for the Project Datasets

The web portal allows a human user to manage the project datasets via an intuitive and interactive GUI as follows:

- First, the user selects a collection resource such as Fish or Site in the web portal. Then, the web portal displays all the instance resources in the selected collection resource on the web browser.
- Second, the user chooses instance resources of interest from the selected collection resource.
- Finally, the user requests CRUD operations for those selected instance resources in the collection resource. In the collection resource, the user can read, update, or delete an existing instance resource (e.g., a particular fish species or site). In addition, the user can create a new instance resource in the collection resource.

Figure 10 shows the GUI window of the web portal for the management of project datasets. In the window, when the user selects the Project Management button in the top command menu bar, the web portal shows the list of collection resources for the project datasets in a pop-up window. If the user selects a collection resource (e.g., Site) in the pop-up window, then the web portal displays all the instance resources in the collection resource in a tabular form in the window. Since the number of instance resources in the collection resources for project datasets is small, the web portal displays all instance resources in the selected collection resource in the GUI window at once.

Figure 10 shows the snapshot of the GUI window displaying all the instance resources in the Site collection resource. In this case, the web portal gets the data from the OSAEM server by sending a GET request with “URI_BASE/sites” as the URI. Then, the web portal in fact receives the representation of the Site collection resource from the OSAEM server as the response shown in Algorithm 7.

ID	Site	Basin	River	Stream Type	Stream Order	Address
1	골지천 02	한강	골지천	지류	3	강원도 태백시
2	골지천 04	한강	골지천	지류	2	강원도 삼척시
3	골지천 10	한강	골지천	지류	5	강원도 정선군
4	송천 03	한강	송천	지류	3	강원도 평창군
5	송천 08	한강	송천	지류	4	강원도 정선군
6	골지천 12	한강	골지천	지류	5	강원도 정선군
7	오대천 01	한강	오대천	지류	2	강원도 평창군
8	오대천 03	한강	오대천	지류	4	강원도 평창군
9	오대천 07	한강	오대천	지류	4	강원도 정선군
10	한강 43	한강	한강	본류	5	강원도 정선군

Figure 10. GUI window for the Site collection resource.

```
[
  {
    "siteID": 1,
    "siteName": "골지천 02",
    "streamOrder": 3,
    "wqmn": "골지천 1",
    "address": "강원도 태백시 사조동 상사미 2 교",
    ... ..
  },
  {
    "siteID": 2,
    "siteName": "골지천 04",
    "streamOrder": 2,
    "wqmn": "번천",
    "address": "강원도 삼척시 하장면 숙암리 번천리측 지류",
    ... ..
  }
  ... ..
]
```

Algorithm 7. Representation for the Site collection resource.

Once the instance resources are displayed in the GUI window, the user can read, update or delete a particular instance resource in the window by clicking on the corresponding button (*i.e.*, the Detail button for Read and Update, and the Delete button for Delete). In addition, the user can also create a new instance resource in the collection resource by clicking on the add button. Figure 11 shows the detailed view provided for a specific site when the detail button for the site resource is clicked.

NAEMP Project Management Observation Data Search Assessment Sign Up Login

Site List Search: search...

Basin:
 River:
 Stream Type: 지류/분류/기타
 Site:
 Stream Order:
 Site ID:

ID	Site	Basin	River	Stream Type	Stream Order	Address	
1	골지천 02	한강	골지천	지류	3	강원도 태백시	Detail Delete
2	골지천 04	한강	골지천				
3	골지천 10	한강	골지천				
4	송천 03	한강	송천				
5	송천 08	한강	송천				
6	골지천 12	한강	골지천				
7	오대천 01	한강	오대천				
8	오대천 03	한강	오대천				
9	오대천 07	한강	오대천				
10	한강 43	한강	한강				

Site Detail

SiteID: 1

SiteName: 골지천 02

RiverName: 골지천

Latitude: 37.28500

Longitude: 128.98972

StreamOrder: 3

StreamGrade:

Address: 강원도 태백시 사조동 상사미2교

Structure:

WQMN: 골지천 1

Image:

Description:

[Save](#) [Cancel](#)

Figure 11. GUI window for the detailed view and update for a site.

Currently, the web portal does not support query services for project datasets because their sizes are small. In other words, it always retrieves all the instance resources from the OSAEM server and displays all of them in the window at once.

5.2. Query Services for the Observation Datasets

Although the RESTful API supports a variety of ways to search for observation data, the OSAEM web portal currently allows the user to search for observation data through the Observation collection resource in two ways:

- *Site-based search.* The site-based search is intended to support site-specific data search and analyses. The user selects sites of interest in the GUI window. Then, the web portal retrieves all the observation data collected at the selected sites and displays them in a tabular form in the web browser.
- *Species-based search.* The species-based search is intended to support comparative and synthetic data search and analyses. The user selects observation targets of interest (e.g., fish species) in the GUI window. Then, the web portal shows all of the observation data for those targets collected at all of the sites.

Selection of the Observation Data Search button in the top command menu bar creates a pop-up window with two choices: Site and Species. If the Site menu is chosen, the web portal displays all the sites and allows the user to filter out those sites not of interest by specifying matching conditions about site attributes. In addition, the user can specify times of interest. Then, the web portal retrieves the observation data collected at those selected sites during the specified times.

Figures 12 and 13 show the GUI windows for the site selection and the display of observation data collected at the selected sites during specified times. In the web portal, the species-based search also works in a similar way. Those retrieved observation datasets are displayed in a tabular form by default. In addition, they can be graphically displayed in charts. Figure 14 shows observation datasets in charts.

ID	Basin	River	Site	Stream Order	Stream Type
	한강	양양남대천	search	search	All
<input type="checkbox"/> 337	한강	양양남대천	양양남대천 08	3	기타
<input type="checkbox"/> 338	한강	양양남대천	양양남대천 01	3	기타
<input type="checkbox"/> 339	한강	양양남대천	양양남대천 02	3	기타
<input type="checkbox"/> 341	한강	양양남대천	양양남대천 03	4	기타

Figure 12. GUI window for the site-based observation data search.

Search Result				
				<input type="text"/> > Value > <input type="text"/>
<input type="text" value="search"/>	<input type="text" value="search"/>	<input type="text" value="search"/>	<input type="text" value="search variable"/>	
양양남대천 01	Water	수질	T-N	1.603
양양남대천 01	Water	수질	Chl-a	0
양양남대천 01	Water	수질	NO3-N	1.23
양양남대천 01	Water	수질	BOD	1.2
양양남대천 01	Water	수질	NH3-N	0.001
양양남대천 01	Water	수질	T-P	0.014
양양남대천 01	Water	수질	PO4-P	0.002
양양남대천 02	Water	수질	BOD	1.1
양양남대천 02	Water	수질	Chl-a	0.9
양양남대천 02	Water	수질	NO3-N	0.71

Figure 13. Observation data collected at selected sites.

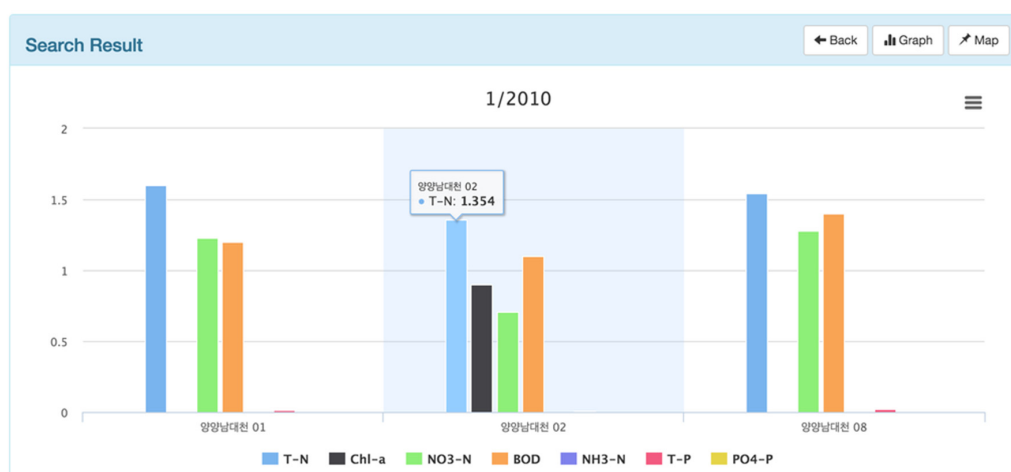


Figure 14. Observation data in the chart window.

6. Discussion and Conclusions

In this paper, we presented the data management system called OSAEM (the Open, Sharable, and Extensible Data Management System for Aquatic Ecological Monitoring), developed to manage datasets from the Korea National Aquatic Ecological Monitoring and Assessment Program (NAEMP) [11,29–34]. In NAEMP, the datasets are collected every six months from 960 sites according to standardized protocols and methods (the number of sites will be increased to 3000 by 2018). Therefore, they are highly comparable and easy to integrate with respect to both species and sites, and provide great potential and many opportunities for advanced comparative and synthetic analyses in various ecological studies and management. However, the realization of such potential and opportunities requires effective data management and sharing of the datasets from NAEMP.

The OSAEM system is designed to facilitate and promote data sharing of the NAEMP datasets. In OSAEM, the management and sharing of the datasets are based on web technology called the RESTful API (the set of resources, URIs, representations and RESTful web services) [10]. In this paper, we explained how the NAEMP datasets are modeled and managed as the RESTful API. The RESTful API for the NAEMP datasets unifies the data models and service interfaces both logically

and physically. This feature significantly facilitates the design and implementation of the application software (e.g., analysis code) to access the datasets.

In this paper, we presented two main features of the OSAEM system: the RESTful API and web portal. In OSAEM, the RESTful API is mainly intended for software rather than the human user, although it is very logical and easy for the user to understand and test with only a web browser. In addition to the RESTful API, a web portal is provided to help the human user access the datasets in the OSAEM system in an easy and interactive manner. The web portal uses the RESTful API to access the datasets in the OSAEM server.

With respect to data sharing, this RESTful API-based data management approach has important advantages over the traditional database-based approach. First, the data models in the publicly available RESTful API (*i.e.*, instance resources, collection resources, URIs, URI patterns, representation schemas, and web services) are simple and intuitive to understand, and significantly facilitate the development of software (e.g., analysis code and visualization code) to access the datasets. In the data models, every entity or object (e.g., a fish species, a site, a parameter, or a method) in ecological monitoring is explicitly referenced and can be individually managed with URIs and representation schemas.

Second, the URIs for data objects (*i.e.*, resources) in the RESTful API can be directly invoked from any computer on the Internet without the installation of any special client software. In other words, the same URIs are used not only to identify data objects (e.g., a specific fish species) but also to invoke corresponding software services. Furthermore, datasets can be easily accessed with simple, generic HTTP clients including web browsers. In addition, the representation schema for the fish species is both its logical data organization and the data format of the response message from the service invocation. Therefore, data sharing can be significantly simplified.

For example, the URI for a fish species (e.g., URI_BASE/fishes/1) is both its unique ID and the Internet network address of the software service used to access the data for the fish species. With an ordinary web browser such as Google Chrome or Microsoft Internet Explorer, any user can access most datasets easily by entering URIs into the URL address bar of the web browser.

Finally, the data models in the RESTful API are not actual database schemas but separate logical models supported by software web services, independent of the underlying database schemas. This two-layered design of the data models makes data analysis or visualization (*i.e.*, using client or application software) independent of the underlying database schemas, and therefore allows the underlying databases to be extended or modified without affecting the software. For example, during the development of the OSAEM prototype system, we have been able to modify the underlying database schemas without changing the RESTful API at all, and *vice versa*.

However, the RESTful API-based data management in OSAEM currently has some drawbacks and limitations that need to be further addressed in future work. First, there is no explicit query language or engine available, although HTTP-based CRUD services are provided for important and frequent types of queries. Therefore, only a predefined set of queries can be supported and complex queries such as the relational join operation are not available. If a new type of query is needed, then a new web service must be developed.

Second, query processing requires more performance overhead because web services for query processing cause additional performance overheads and are not as optimized as database query engines. However, the size of the NAEMP datasets (*i.e.*, hundreds of thousands of data objects) is, fortunately, not large enough to cause serious performance overhead on conventional server computers. These performance issues will be addressed together with query engines in future work.

Third, the web portal currently allows the user to access a very limited subset of services from the current RESTful API. Furthermore, a variety of analysis or visualization code can be added to the web portal. Such extensions to the web portal are planned for 2016.

Currently, there are no data sharing policies established in NAEMP. Since the datasets are generated from the government program, they are generally assumed to be open to everyone. However, effective data sharing requires well-specified policies.

Finally, there is currently insufficient support for management tools, and especially little support for security such as authentication and authorization. For example, the catalogs of resources and services are created and made available on web pages in an *ad hoc* way in the current OSAEM system.

Currently, the OSAEM system is under active development to address and improve the above issues and limitations. In addition, a prototype system is installed and is being tested on the Amazon EC2 cloud service. The current prototype for the web portal is available at <http://db.cilaboratory.org:8080/naemp>. The current prototype system supports the monitoring datasets for fish and water quality. The addition of monitoring datasets for benthic diatoms and macroinvertebrates is planned for 2016.

Acknowledgments: This study was performed under the “National Aquatic Ecosystem Health Survey and Assessment” project in Korea and supported by the Ministry of Environment and the National Institute of Environmental Research, Korea. The authors are grateful to survey members involved in the project. The authors also thank the anonymous reviewers for their help in improving the scientific content of the manuscript.

Author Contributions: Meilan Jiang designed and implemented the OSAEM system. Karpjoo Jeong designed the RESTful API and led the overall system development work. Nan-Young Kim managed the datasets and contributed to the design of the data models. Jung-Hwan Park and Sang-Hun Kim worked on the analysis of requirements for services and contributed to the design of services. Soon-Jin Hwang planned and coordinated the entire development project for the OSAEM system, and also contributed to the overall system design.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Representation Schemas.

Collection Resources	Representation Scheme
Fish	<pre> { "fishID": <integer>, "class": <string>, "order": <string>, "family": <string>, "species": <string>, "scientificName": <string>, "toleranceGuild": <string>, "trophicGuild": <string>, "habitatGuild": <string>, "invasiveSpecies ": <string>, "endemicSpecies ": <string>, "endangeredSpecies ": <string>, "naturalMonument ": <string>, "imageLink": <string>, "description": < string> }</pre>
Site	<pre> { "siteID": <integer>, "siteName": <string>, "latitude": <string>, "longitude": <string>, "river": <integer (riverID) >, "streamOrder": <integer>, "streamGrade": <string>, "address": <string>, "image": <string>, "standardStructure": <string>, "wqmn ": <string>, "description": < string> }</pre>

Collection Resources	Representation Scheme
River	<pre>{ "riverID": <integer>, "riverName": <string>, "basin": <string>, "waterSystem": <string>, "midWatershed ": <string>, "subWatershed ": <string>, "classification": <string>, "image": <string>, "description": < string> }</pre>
Variable	<pre>{ "variableID": <integer>, "variableName": <string>, "valueType": <string>, "unit": <integer (unitID)>, "description": <string> }</pre>
Unit	<pre>{ "unitID": <integer>, "unitName": <string>, "unitNameLong": < string> }</pre>
Method	<pre>{ "methodID": <integer>, "methodName": <string>, "methodLink": < string> }</pre>
Source	<pre>{ "sourceID": <integer>, "institution": <string>, "investigator": <string>, "phone": <string>, "email": <string>, "description": <string> }</pre>
Observation	<pre>{ "valueID": <integer>, "dateTime": <string>,, "surveyYear": <integer>, "surveyTerm": <integer>, "dataValue": <double>, "site": <integer (siteID)>, "latitude": <string>, "longitude": <string>, "source": <integer (sourceID)>, "entity": <integer (fishID)>, "variable": <integer (variableID)>, "method": <integer (methodID)> }</pre>

References

1. Lee, S.-W.; Hwang, S.-J.; Lee, J.-K.; Jung, D.-I.; Park, Y.-J.; Kim, J.-T. Overview and application of the National Aquatic Ecological Monitoring Program (NAEMP) in Korea. *Ann. Limnol. Int. J. Lim.* **2011**, *47*, S3–S14. [[CrossRef](#)]

2. Michener, W.K. Ecological data sharing. *Ecol. Inform.* **2015**, *29*, 33–44. [[CrossRef](#)]
3. Michener, W.K.; Beach, J.H.; Jones, M.B.; Ludäscher, B.; Pennington, D.D.; Pereira, R.S.; Rajasekar, A.; Schildhauer, M. A knowledge environment for the biodiversity and ecological sciences. *J. Intell. Inf. Syst.* **2007**, *29*, 111–126. [[CrossRef](#)]
4. Michener, W.K.; Jones, M.B. Ecoinformatics: Supporting ecology as a data-intensive science. *Trends Ecol. Evol.* **2012**, *27*, 85–93. [[CrossRef](#)] [[PubMed](#)]
5. Hampton, S.E.; Strasser, C.A.; Tewksbury, J.J.; Gram, W.K.; Budden, A.E.; Batcheller, A.L.; Duke, C.S.; Porter, J.H. Big data and the future of ecology. *Front. Ecol. Environ.* **2013**, *11*, 156–162. [[CrossRef](#)]
6. Molloy, J.C. The open knowledge foundation: Open data means better science. *PLoS Biol.* **2011**, *9*, e1001195. [[CrossRef](#)] [[PubMed](#)]
7. Madin, J.S.; Bowers, S.; Schildhauer, M.P.; Jones, M.B. Advancing ecological research with ontologies. *Trends Ecol. Evol.* **2008**, *23*, 159–168. [[CrossRef](#)] [[PubMed](#)]
8. Madin, J.; Bowers, S.; Schildhauer, M.; Krivov, S.; Pennington, D.; Villa, F. An ontology for describing and synthesizing ecological observation data. *Ecol. Inform.* **2007**, *2*, 279–296. [[CrossRef](#)]
9. Reichman, O.J.; Jones, M.B.; Schildhauer, M.P. Challenges and opportunities of open data in ecology. *Science* **2011**, *331*, 703–705. [[CrossRef](#)] [[PubMed](#)]
10. Fielding, R.T.; Taylor, R.N. Principled design of the modern Web architecture. *ACM Trans. Internet Technol. TOIT* **2002**, *2*, 115–150. [[CrossRef](#)]
11. Guinard, D.; Trifa, V.; Wilde, E. A resource oriented architecture for the Web of Things. In Proceedings of the Internet of Things (IOT), Tokyo, Japan, 29 November–1 December 2010; pp. 1–8.
12. Barry, M.G.; Purcell, M.E.; Eck, B.J.; Hayes, J.; Arandia, E. Web services for water systems: The iWIDGET REST API. *Procedia Eng.* **2014**, *89*, 1120–1127.
13. Bianchi, L.; Paganelli, F.; Pettenati, M.C.; Turchi, S.; Ciofi, L.; Iadanza, E.; Giuli, D. Design of a RESTful Web information system for drug prescription and administration. *Biomed. Health Inform. IEEE J.* **2014**, *18*, 885–895. [[CrossRef](#)] [[PubMed](#)]
14. Paganelli, F.; Turchi, S.; Bianchi, L.; Giuli, D. An information-centric and REST-based approach for EPC Information Services. *J. Commun. Softw. Syst.* **2013**, *9*, 14–23.
15. Li, L.; Wu, C. Design and describe REST API without violating REST: A Petri net based approach. In Proceedings of the 2011 IEEE International Conference on Web Services (ICWS), Washington, DC, USA, 4–9 July 2011; pp. 508–515.
16. Allard, S. DataONE: Facilitating eScience through collaboration. *J. eScience Librariansh.* **2012**, *1*, e1004. [[CrossRef](#)]
17. Michener, W.K.; Allard, S.; Budden, A.; Cook, R.B.; Douglass, K.; Frame, M.; Kelling, S.; Koskela, R.; Tenopir, C.; Vieglais, D.A. Participatory design of DataONE—Enabling cyberinfrastructure for the biological and environmental sciences. *Ecol. Inform.* **2012**, *11*, 5–15. [[CrossRef](#)]
18. Horsburgh, J.S.; Tarboton, D.G.; Maidment, D.R.; Zaslavsky, I. A relational model for environmental and water resources data. *Water Resour. Res.* **2008**, *44*, W05406. [[CrossRef](#)]
19. Horsburgh, J.S.; Tarboton, D.G.; Piasecki, M.; Maidment, D.R.; Zaslavsky, I.; Valentine, D.; Whitenack, T. An integrated system for publishing environmental observations data. *Environ. Model. Softw.* **2009**, *24*, 879–888. [[CrossRef](#)]
20. Whitlock, M.C. Data archiving in ecology and evolution: Best practices. *Trends Ecol. Evol.* **2011**, *26*, 61–65. [[CrossRef](#)] [[PubMed](#)]
21. Berkley, C.; Bowers, S.; Jones, M.B.; Madin, J.S.; Schildhauer, M. Improving Data Discovery for Metadata Repositories through Semantic Search. In Proceedings of the 2009 International Conference on Complex, Intelligent and Software Intensive Systems, Fukuoka, Japan, 16–19 March 2009.
22. Jones, M.B.; Berkley, C.; Bojilova, J.; Schildhauer, M. Managing Scientific Metadata. *J. IEEE Internet Comput.* **2001**, *5*, 59–68. [[CrossRef](#)]
23. Greenberg, J.; White, H.C.; Carrier, S.; Scherle, R. A Metadata Best Practice for a Scientific Data Repository. *J. Libr. Metadata* **2009**, *9*, 194–212. [[CrossRef](#)]
24. Michener, W.; Vieglais, D.; Vision, T.; Kunze, J.; Cruse, P.; Janée, G. DataONE: Data observation network for earth-preserving data and enabling innovation in the biological and environmental sciences. *D-lib Mag.* **2011**, *17*, 1–12. [[CrossRef](#)]

25. Wieczorek, J.; Bloom, D.; Guralnick, R.; Blum, S.; Döring, M.; Giovanni, R.; Robertson, T.; Vieglaiss, D. Darwin Core: An evolving community-developed biodiversity data standard. *PLoS ONE* **2012**, *7*, e29715. [[CrossRef](#)] [[PubMed](#)]
26. Whiteaker, T.; Ernest, T. CUAHSI Web Services for Ground Water Data Retrieval. *Ground Water* **2008**, *46*, 6–9. [[CrossRef](#)]
27. Patton, E.W.; Seyed, P.; Wang, P.; Fu, L.; Dein, J.; Bristol, S.; McGuinness, D.L. SemantEco: A semantically powered modular architecture for integrating distributed environmental and ecological data. *Future Gener. Comput. Syst.* **2014**, *36*, 430–440. [[CrossRef](#)]
28. Moura, A.D.C.; Porto, F.; Poltosi, M.; Palazzi, D.C.; Magalhães, R.P.; Vidal, V. Integrating ecological data using linked data principles. In Proceedings of Joint V Seminar on Ontology Research in Brazil (ONTOBRAS) and VII International Workshop on Metamodels, Ontologies and Semantic Technologies (MOST), Recife, Brazil, 19–21 September 2012; pp. 156–167.
29. Stoddard, J.L.; Herlihy, A.T.; Peck, D.V.; Hughes, R.M.; Whittier, T.R.; Tarquinio, E. A process for creating multimetric indices for large-scale aquatic surveys. *J. N. Am. Benthol. Soc.* **2008**, *27*, 878–891. [[CrossRef](#)]
30. Karr, J.R. Assessment of biotic integrity using fish communities. *Fisheries* **1981**, *6*, 21–27. [[CrossRef](#)]
31. Hering, D.; Johnson, R.K.; Kramm, S.; Schmutz, S.; Szoszkiewicz, K.; Verdonschot, P.F.M. Assessment of European streams with diatoms, macrophytes, macroinvertebrates and fish: A comparative metric-based analysis of organism response to stress. *Freshw. Biol.* **2006**, *51*, 1757–1785. [[CrossRef](#)]
32. Johnson, R.K.; Furse, M.T.; Hering, D.; Sandin, L. Ecological relationships between stream communities and spatial scale: Implications for designing catchment-level monitoring programmes. *Freshw. Biol.* **2007**, *52*, 939–958. [[CrossRef](#)]
33. Johnson, R.K.; Goedkoop, W.; Sandin, L. Spatial scale and ecological relationships between the macroinvertebrate communities of stony habitats of streams and lakes. *Freshw. Biol.* **2004**, *49*, 1179–1194. [[CrossRef](#)]
34. Kelly, M.G.; Whitton, B.A. The Trophic Diatom Index: A new index for monitoring eutrophication in rivers. *J. Appl. Phycol.* **1995**, *7*, 433–444. [[CrossRef](#)]
35. Crockford, D. The application/json media type for JavaScript Object Notation (JSON). 2006. Available online: <http://tools.ietf.org/html/rfc4627> (accessed on 27 November 2015).
36. Fielding, R.T.; Gettys, J.; Mogul, J.C.; Frystyk, H.F.; Masinter, L.; Leach, P.; Berners-Lee, T. *Hypertext Transfer Protocol—HTTP/1.1*. Internet RFC 2616; The Internet Society: Reston, VA, USA; June; 1999.
37. Murugesan, S. Understanding Web 2.0. *IT Prof.* **2007**, *9*, 34–41. [[CrossRef](#)]
38. Seo, D.; Lee, J. Web_2.0 and five years since: How the combination of technological and organizational initiatives influences an organization's long-term Web_2.0 performance. *TELE* **2016**, *33*, 232–246. [[CrossRef](#)]



© 2016 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC-BY) license (<http://creativecommons.org/licenses/by/4.0/>).