Journal of Computer and Communications, **2014, 2, 6-12**
Published Online January 2014 (http://www.scirp.org/journal/jcc)
http://dx.doi.org/10.4236/jcc.2014.21002

A System for Managing Oceanographic Metadata Using XML and XQuery

Elena Partescano, Alberto Brosich, Alessandra Giorgetti

Istituto Nazionale di Oceanografia e di Geofisica Sperimentale (OGS), Sgonico, Italy.
Email: epartescano@ogs.trieste.it

Received October 25th, 2013; revsed November 20th, 2013; accepted November 28th, 2013

ABSTRACT

Metadata, by definition, is information associated with data, covering where, how, when and by whom the data were acquired. The chance to take part in European projects such as the EU-SeaDataNet (the Pan-European infrastructure for ocean and marine data management) made it necessary to use XML (Extensible Markup Language) as a standard file format for sharing data and metadata. At present, the Italian National Oceanographic Data Centre (OGS/NODC) has all its data and metadata contained in an Oracle relational database, and the metadata is managed using XML formats and schema (XSD; XML Schema Definition), giving common vocabularies for parameters, instruments, ships, etc. in agreement with European project standards. One problem with XML is the dynamic change in metadata schemas (XSD), necessitating development of a system which is flexible and capable of managing the changes. This paper describes a system for managing oceanographic metadata using XML files and the functionalities provided by the Oracle database. To better manage the XML format, we chose to load into the OGS/NODC database the whole XML file, using a dedicated field. The database Oracle gives us the flexibility to manage the XML format locally, within the OGS/NODC information system, using the XML DB (XML DataBase) Oracle features. This, through the use of XMLType, allows the inclusion of the XML into the database. Furthermore, through the use of the XQuery functions it is possible to create a set of views through which information contained in the XML can be viewed more immediately in a relational form. Using the XML and the XQuery functions, it is possible to store, extract and manage different kinds of information that might be exchanged at the European level. Moreover, with a RESTful (Representational State Transfer) Web Service, we have a simple and standard interface for rapidly and easily creating, modifying and deleting records containing XML files inside the database. Finally, through the use of a RESTful Web Service, it is possible to decouple the application from the database, so that through the use of software that manages HTTP URLs, such as the Mikado (SeaDataNet project), the XML file can be inserted, updated and deleted inside the database without the need for a direct connection to it.

KEYWORDS

Database; XML; Web Service; XQuery

1. Introduction

The term metadata is defined as “a set of data that describes and gives information about other data” [1]. Data and metadata are closely related. Both are vital to complete and catalogue a data-set in oceanography, given that the data values are georeferenced and closely linked to the measurement systems.

The metadata is the information describing the data and defining the features of the data, such as the acquisition conditions and the quality, contents, ownership and policies of the data. In the past, the metadata was considered of secondary importance. When computers became available for storing data, the importance of metadata grew; consequently, at the same time the techniques for storing the metadata developed [2]. Nowadays, to ma-

nage metadata it is necessary to carry out the same CRUD (Create, Read, Update and Delete) actions used to manage the data.

In the course of time, different management systems have been produced.

At the present time, the XML format and the associated schema languages (XSD, Schematron etc.) seem to be the best solution for managing this kind of data because XML is a versatile tool and may be used in different contexts to represent information in a manner independent of the standard used. XML allows definition of a series of rules capable of defining the contents and the structures of the documents. In addition, XML is extremely functional, using a wide range of applications, especially when it is necessary to exchange and share data with heterogeneous databases.

This paper describes an example for managing oceanographic metadata, using a relational database and XML with associated technologies.

The Italian National Oceanographic Data Center (OGS/NODC) has collected into the database more than 360 thousand stations and 200 million measurements of physical and biogeochemical parameters, of current, wave motion, sea level and meteorological data (**Figure 1**).

The information in the OGS/NODC data archiving system is fully validated, continuously updated and accessible through a dedicated web portal, available on the internet (http://nodc.ogs.trieste.it). All the metadata are public whereas the access to the archived data is subjected to a Data Policy, defined at data-set level, in agreement with the data providers [3].

Metadata is the fundamental element from which it is possible to discovery data in a database. It is the element collecting all the information that constrains the data at spatial and temporal positions. Since the early 1990s, in the framework of the European projects focusing on metadata management (EDMED, European Directory of Marine Environmental Data; MTPII/MATER, Mass Transfer and Ecosystem Response; MEDAR/MEDATLAS II, Mediterranean Data Archaeology and Rescue; EuroGOOS-EDIOS, European Directory of the Ocean-observing System; SeaSearch, a Pan-European Network for Ocean & Marine Data and Information Management; and SeaDataNet and SeaDataNet2, the Pan-European Infrastructure for Ocean & Marine Data Management) a set of metadata directories was created and is maintained in order to provide a low-level inventory for tracking oceanographic data collected at sea and to better manage the oceanographic metadata. These directories are: the European Directory of Marine Environmental Data sets (EDMED), the European Directory of Marine Environmental Research Projects (EDMERP), the Cruise Summary Reports (CSR = former ROSCOP) and the European Directory of the initial Ocean-observing Systems (EDIOS) (**Figure 2**).

Through the use of the standard metadata directories, common vocabularies, common metadata transport formats and common quality control protocols, the interoperability is guaranteed. Currently, different ISO standards are applied to the metadata; a common file format is also necessary to enable information exchange. The ISO 19115 content model was chosen, together with the XML format and exchange schemas (XSD). In this context, the use of XML and the associated schema (XSD) makes

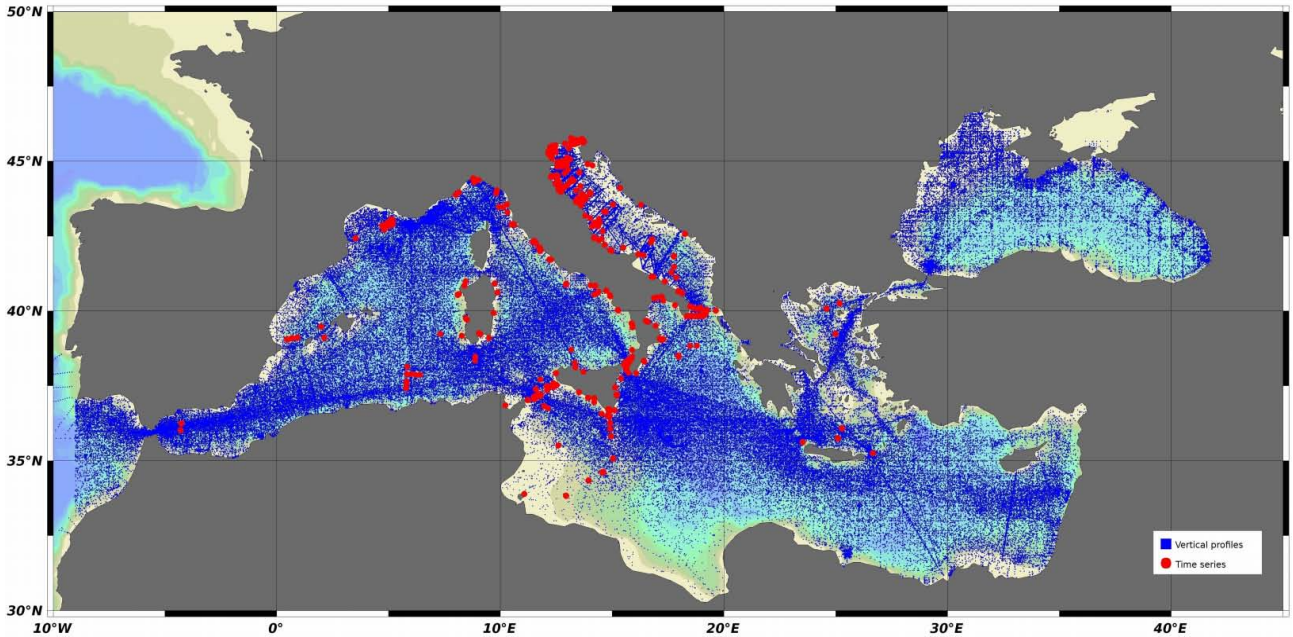

Figure 1. Distribution of the data loaded in the OGS/NODC database.

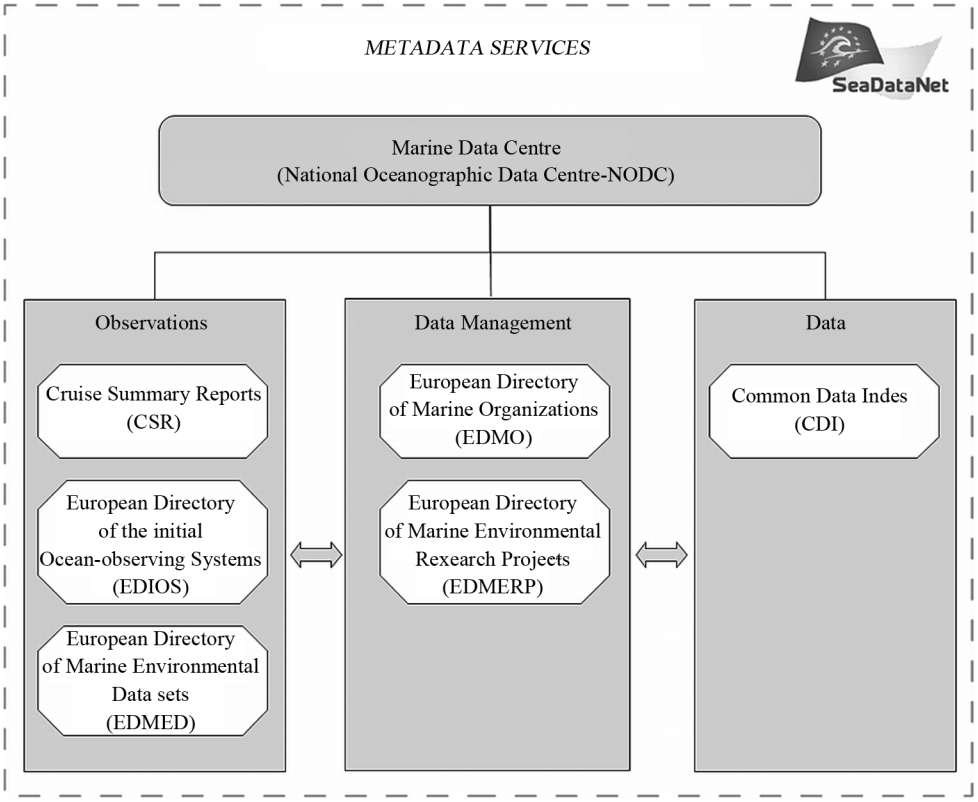

Figure 2. Metadata directories overview ([4], modified).

it an ideal metadata transport format, especially considering the need to follow European standards.

Archiving data and metadata using a relational database offers many advantages but also some disadvantages. The prominent disaggregation of the information inside an RDBMS increases the cost of potential adaptations of the data schema. For this reason, the adoption and the management of the European catalogs have been a gradual process.

Management strategies have changed with time: starting with the CSRs it was sufficient to alter the existing tables whereas with EDMED and EDMERP it has been necessary to add some specific tables to the relational schema already present. In these cases we mapped XML schemata to a relational schema; onto this schema we built the management tools for adding and updating the information and to recreate the XML files for exchange purposes.

The change in the XML schemata (due to the adaptation to international standards) imposed new management strategies. On one hand the cost of the adaptation of the relational schema and of the related management software was too high; on the other, a management tool for the XML files became available inside the SeaDataNet project. The archiving of the catalog elements would have been done as simple physical files but we could not afford to lose the connection with the information (data and metadata) already present in our relational database. The necessity for a flexible system, capable of adapting with the dynamic changes in the XML schema, has led us to direct storage and retrieval of XML files into the relational database currently installed at the NODC (Oracle RDBMS), using the Oracle XML DB feature to load and manage the information.

The primary aim of this paper is to explain a viable system for managing oceanographic metadata using direct storage in an Oracle database of XML files and to describe its functionalities. All the Oracle XML functionalities are backed by the W3C (World Wide Web Consortium) and they give the opportunity to collect, query, update and modify the XML using SQL (Structured Query Language; [5]).

The XML files inserted directly into the Oracle database make it possible to administer the information associated with the European metadata directories.

At present, in our database we can manage the XML files of the European directories associated with the Marine Environmental Research Projects (EDMERP) and the Marine Environmental Data sets (EDMED) (**Figure 2**).

We used the XMLQuery and XPath expressions to determine the attribute or the node in the file and consequently it was possible to create the views used to visualize the information contained in the XML files as

defined by Adam [1].

In addition, we developed a RESTful Web Service to manage (create and update) these XML files, decoupling the client applications from the database. In this way, we can use any application capable of handling HTTP URLs such as Mikado, a tool developed by the SeaDataNet project for the generation of XML files with the information included in the European metadata directories.

2. Materials and Methods

2.1. XML

Through the use of XML format we will be able to manage a large amount of organized and searchable information, simply implementing the system through the use of a table with two fields, one for handling the XML file (XMLType) and one for the primary key.

The Oracle database uses the XML DB feature, which enables storage and retrieval of the XML [5]. With XMLType, SQL developers have the power of the relational database while working in the context of XML (**Figure 3**).

The XML DB (**Figure 4**) also makes it possible to insert the XML schema, enabling a check on the structure of the XML loaded into the database. For storing XML metadata files, a specific field in the Oracle database (XMLType) can be used. XMLType is an object and a column can contain any well-formed XML document; the presence of an XML schema can control and validate the conformity of the XMLType column (**Figure 3**).

We thus have confirmation that the XML in the database is validated following the XML schema and all the XML have the same XML structure. This functionality ensures that the XML files are well-formed and valid.

To extract the data, as defined by Adams [5], the Oracle XML DB provides the SQL functions defined in the SQL/XML developing standard, so the functions will change in the future. To query and access XML content it is possible to use the functions employing XQuery or XPath expressions to recognize the nodes and analyze the XML structure.

In addition, the Application Program Interfaces (APIs) in PL/SQL and Java can facilitate the handling of the XMLType, if we need to write some procedures.

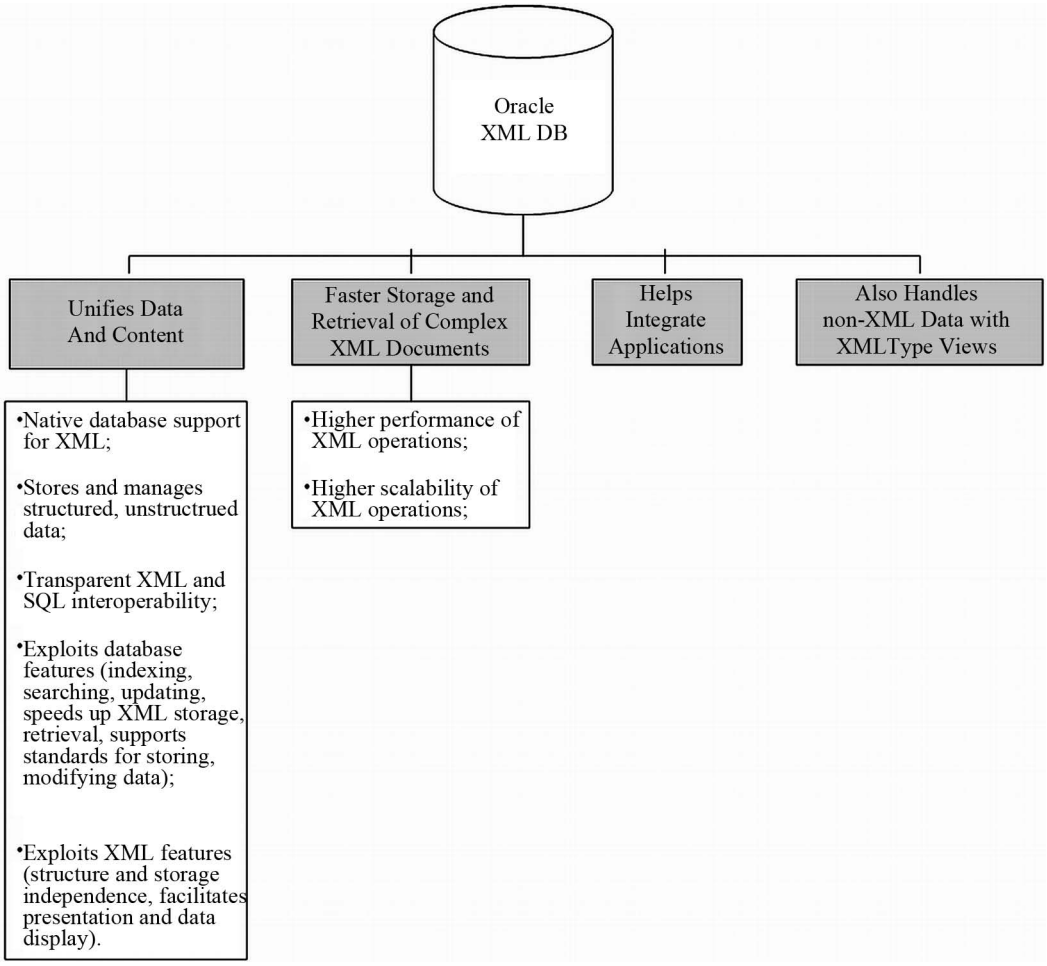

Figure 3. XMLType storage (modified from Adams, 2008).

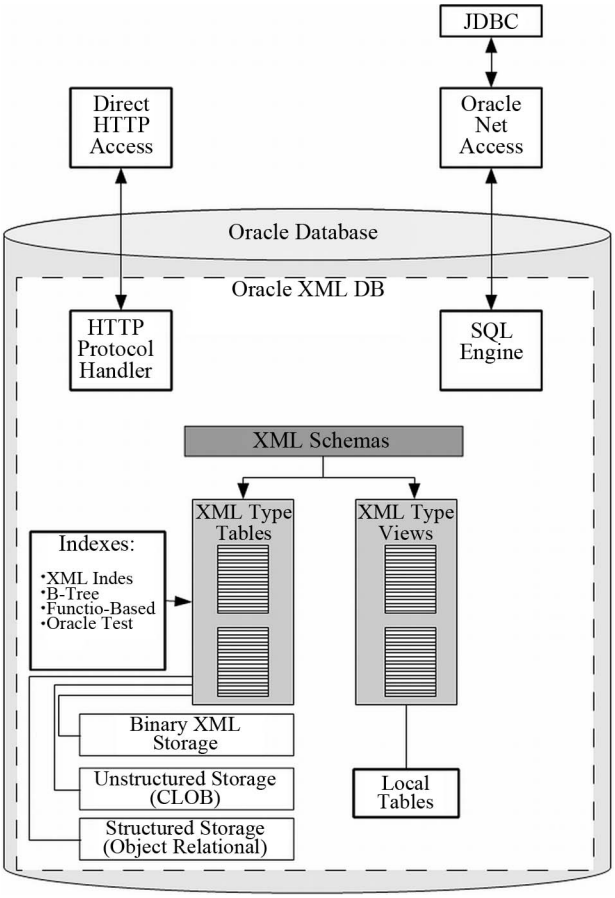

Figure 4. **XML DB Benefits (modified from Adams, 2008).**

2.2. XQuery

XQuery (**Figure 5**) makes it possible to execute queries on XML documents.

To access the information in the XML file, it is necessary to use the SQL/XML standard function XMLQuery.

The XPath expression permits identification of the single node or attribute value in the file demanded by the XMLQuery.

The SQL/XML standard functions (XPath and XMLQuery) are used to create the views (**Figure 6**) to aggregate in a relational way the information contained in the XMLType column in the Oracle database.

These views were created with SQL functions such as EXTRACT to select the value of a node. XQuery is created on XPath expressions and, as stated by Adams [5], XQuery 1.0 and XPath 2.0 share the same data model and use the same functions and operators. To select and organize the data contained in the XML documents, XQuery is the main tool.

2.3. Web Service

The software Mikado was only able to read from a local file system. We decided that the simplest modification to

```
SELECT
cat_id,
extractva-
lue(xml_data,'Metadata/dataIdInfo/idCitation/resAltTitle[@type="project
reference"]') as prj_id,
extractvalue(xml_data,'Metadata/mdContact/rpOrgName') as collate_centre,
extractvalue(xml_data,'Metadata/mdContact/rpOrgName/@SDNIdent')
as collate_centre_edmo_code,
extractvalue(xml_data,'Metadata/dataIdInfo/idCitation/resTitle') as prj_title,
extractva-
lue(xml_data,'Metadata/dataIdInfo/idCitation/resAltTitle[@type="project
acronym"]') as prj_acronym,
extractva-
lue(xml_data,'Metadata/dataIdInfo/idCitation/aggrInfo/aggrDSName/resTitle') as
associated_programme,
extractva-
lue(xml_data,'Metadata/dataIdInfo/dataExt/tempEle/TempExtent/exTemp/TM_G
eometricPrimitive/TM_Period/begin') as begin_year,
extractva-
lue(xml_data,'Metadata/dataIdInfo/dataExt/tempEle/TempExtent/exTemp/TM_G
eometricPrimitive/TM_Period/end') as end_year,
extractvalue(xml_data,'Metadata/dataIdInfo/idAbs') as summary,
extractvalue(xml_data,'Metadata/distInfo/distTranOps/onLineSrc/linkage')
as prj_website,
extractvalue(xml_data,'Metadata/mdDateSt') as currency_date,
extractvalue(xml_data,'Metadata/dataIdInfo/idCitation/resRefDate/refDate')
as revision_date,
extractvalue(xml_data,'Metadata/dataIdInfo/tpCat/TopicCatCd/@value')
as theme_project,
extractValue(xml_data,'Metadata/mdLang/languageCode/@value') as languages,
extractva-
lue(xml_data,'Metadata/dataIdInfo/idCitation/citRespParty[role/RoleCd/@value=
"pointOfContact"]/rpIndName') as prj_coord_name,
extractva-
lue(xml_data,'Metadata/dataIdInfo/idCitation/citRespParty[role/RoleCd/@value=
"pointOfContact"]/rpOrgName') as prj_coord_institute
FROM catalogs_xml
```

Figure 5. **Example of XQuery to extract data.**

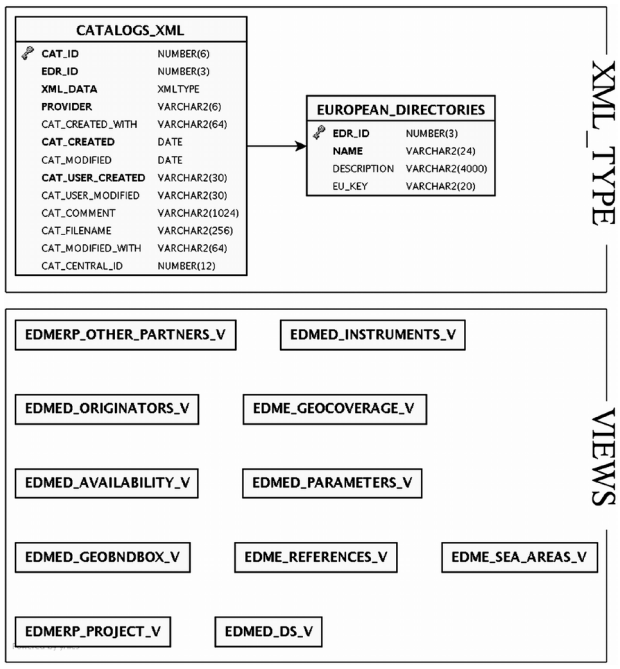

Figure 6. **The views needed to manage the XML files.**

ask of the software developers was to add the ability to archive using the HTTP protocol following the REST principle, where some HTTP methods (POST, GET, PUT, DELETE) are, respectively, used to perform the man-

agement operations (Insert, Read, Update, Delete) (Figure 7). The RESTful Web Service written from scratch is the counterpart receiving these HTTP requests.

It accepts the requests (authenticated when they require the insertion, the updating or the deleting of information) and acts as interface with the relational database using the traditional JDBC API, as it is written in the Java language. Our application uses the data type XMLType from Oracle: it is not standard and thus it was not possible to use an ORM (Object Relational Mapping) framework such as Hibernate because it is not natively supported.

The Web Service is able to insert files about any catalog using a URL with the pattern: "ws/CatalogElement/{catalog}/[{id}]", where the string "ws" is the path that identifies the Web Service and it can be defined during the deployment phase.

The string "CatalogElement" describes the resource (in the RESTful meaning), while the pattern "{catalog}/{id}" describes the single catalog element (EDMED or EDMERP) in the database. For example, using the pattern "EDMED/123", it is possible to visualize the EDMED with "id" 123. The "id" is created by the database when a new record is loaded; therefore it is unnecessary in the case of a POST request (when a new resource is created). On this occasion the new "id" is included in the HTTP reply so that the Mikado software can display it to the user. In addition the "id" is inserted into the XML file to guarantee an unequivocal association between the document and the database. To do this in the database the input procedure uses a trigger that immediately updates the "id" which has been added.

If the catalog doesn't exist in the database an error message will be provided. To add a new catalog it is necessary to add a new row into the table containing the catalogs. At the same time that a new XML file is inserted into the database, the user-name, the creation timestamp, the last modification timestamp and the user-agent of the HTTP client used are saved in the table of the catalog elements.

In detail, the allowable operations are (Figure 7): the reading of an XML document using the GET method, which returns the content of the XMLType field of the record associated with the "id" included in the URL; the updating of the XML file using the PUT method, which allows the overwriting of the document previously inserted into the database. The DELETE function doesn't exist in the Mikado software, but the user can delete the XML file using an external script by a software capable of sending an HTTP request (for example "curl" in UNIX/Linux), or using the traditional SQL instructions inside the relational database.

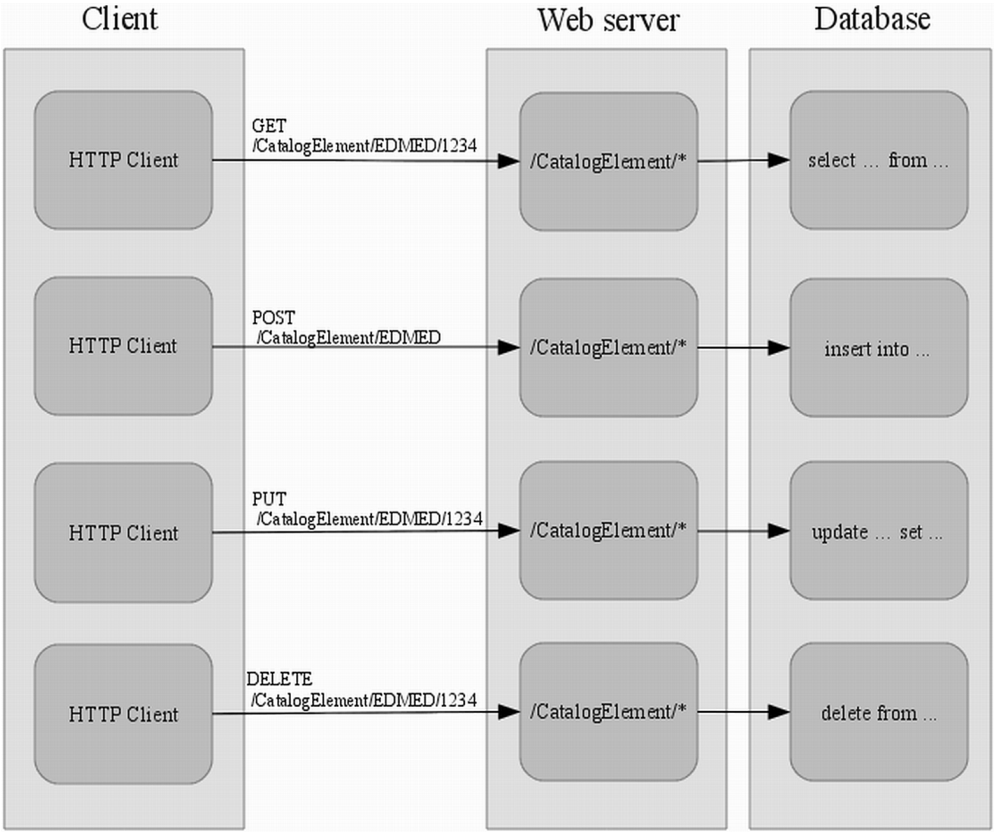

Figure 7. The allowable operations in the Web Service.

3. Discussions and Conclusions

Metadata and data are two components closely related to one another; the handling of standards and vocabularies that constrain the information contained therein are facilitated through the use of a format such as XML. XML has now become a well established exchange format for metadata; it is able to store information in a local database directly using an XML file, and it facilitates the management procedures and exchange of information.

The need for a system to manage the metadata (storing, modification...), capable of adapting quickly if the XML schema changes or a new directory is introduced, led us to evaluate carefully the objectives mentioned above, the available technologies and the cost of development. There was an urgent need to change the approach; this solution was made easier by the existence of a tool to manage the XML, developed within the project SeaDataNet (Mikado). The new strategy allowed the use of XML files as a tool not only to exchange but also to store the metadata. This new approach made the mapping between the XML schemata and the relational database unnecessary. Using Mikado enabled us to manage the metadata, specifically its storing and modification; this permitted us to focus on the second part of the problem: the connection between the metadata directories and the information in the relational database. For this purpose, the only possible solution was to store the XML files directly in the RDBMS, using Oracle's XML functionalities. These functionalities enable the use of the Oracle database (RDBMS, currently used in the Italian NODC) as an "XML database", adopting XMLType for storing the native XML and using the associated functions for their management: the insert, update and query.

To retrieve the information contained inside the XML file we apply the XQuery technology, which is a W3C (World Wide Web Consortium) technology used to extract data from XML documents. Through these technologies, we created some views to have a relational view of the information contained in the XML files. These views permit use of the traditional SQL to query the metadata. The final missing element was a system for archiving the XML files in the database using the Mikado software. For this, we built a RESTful Web Service to get and update the XML files into the database, and we adopted an HTTP protocol used by the Mikado software to insert and store the XML file in the database.

Using a system like this we have the advantage that every single element is independent of the others, and the independence among elements guarantees us more versatility. This has the benefit that changes in the database don't affect the working of the system; in the same manner, there is no obligation to use the Mikado software for managing the XML data and there is complete autonomy from the Web Service used and the technology implemented. Some possible disadvantages are that to use the XML files (having a complex structure and thus hard to manage), we practically have to create the views and use an application (such as Mikado) to manage the metadata.

Future Developments

In the future, we hope, given the versatility of XML, to manage not only the information associated with the data but also the data itself, thus ensuring greater ease of insertion, extraction, sharing, etc. during data handling itself.

Acknowledgements

This work was supported by the EU-SeaDataNet (the Pan-European infrastructure for ocean and marine data management).

REFERENCES

[1] Oxford Dictionaries "Oxford Dictionaries," Oxford University Press, 2010. http://oxforddictionaries.com/definition/metadata.

[2] A. Sen, "Metadata Management: Past, Present and Future," *Decision Support Systems*, Vol. 37, No. 1, 2004, pp. 151-173. http://dx.doi.org/10.1016/S0167-9236(02)00208-7

[3] D. Schaap and R. K. Lowry, "SeaDataNet—Pan-European Infrastructure for Marine and Ocean Data Management: Unified Access to Distributed Data Sets," *International Journal of Digital Earth*, Vol. 3, No. S1, 2010, pp. 50-69. http://dx.doi.org/10.1080/17538941003660974

[4] S. Iona, M. Fichaut, A. Che-Bohnenstengel, D. Schaap and L. Rickards, "User Manual and Instructions for Updating EDMED, EDMERP, EDIOS, EDMO and CSR, Deliverable 4.6," 2012. http://www.seadatanet.org/content/download/11284/75105/file/SDN2_D46_NC_UserManualforUpdatingMetadataDirectories.pdf

[5] D. Adams, "Oracle® XML DB Developer's Guide11g Release 1 (11.1)," 2008. http://docs.oracle.com/cd/B28359_01/appdev.111/b28369/toc.htm