International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 20 (2017) pp. 9879-9884


# A Study on Extension of XMDR-DAI Considering the Relationships between Classes

**Sung-Uk Choi*, Seok-Jae Moon****

** Department of Computer Science, Incheon National University, Academy-ro, Yeonsu-gu, Incheon, 406-772, Korea*

***Department of Information System Kwangwoon University Graduate School of Information Contents, 20 Kwangwoon-ro, Nowon-gu, Seoul 139-701, Korea*

**Corresponding author: Seok-Jae Moon, Ph.D.*

**Abstract**

Recently, interest in Big Data has increased and many application systems and services have been developed. As a result, discussions for managing and utilizing big data standardization from the perspective of big data are continuing in the field of metadata standards. In this paper, we propose eXMDR-DAI (Extension of XMDR-DAI), which is an extended interface that allows interoperability between data, considering the relationship between metamodel classes. The eXMDR-DAI solves metadata information conflicts when sharing and integration data between systems in a big data-based cloud environment. It also supports the efficient operation of information system processes. The eXMDR-DAI defines the conceptual relationships between metadata and provides better performance than basic approach in terms of query modeling and system referential integrity.

**Keywords**: MetaData Registry, MRA, XMDR-DAI, Data interoperability, Big Data

## INTRODUCTION

XMDR (eXtended Metadata Registry) is an international standard for managing standardized data through the registration and authentication of metadata [1, 2]. It also improves data interoperability through the sharing of metadata specifications and semantics [3, 4]. Using the characteristics of XMDR, XMDR-based systems are developed and operated in various big data integration fields. [5]. Several systems based on XMDR modify some of the conceptual models defined in the metadata standard during construction. In addition, we develop data access methods by presenting different storage models. This is a heterogeneous structure of the metadata that is managed by the management, which causes difficulties in sharing and reusing data based on the meta information stored in the XMDR system. Also, since the interface for data access is dependent on each system, in order to retrieve the data elements desired by the user, there is an overhead to identify each interface structure. To deal with these problems, the ISO/IEC JTC 1/SC 32/WG4, the ISO/IEC 13249-8 Metadata Registry Access (MRA) standard has been proposed. MRA has the purpose of accessing meta information based data stored in a class defined in XMDR. However, MRA provides only basic data types for metadata access and does not provide detailed specifications. This leads to practical utility and functionality limitations of MRA. Therefore, it is required to expand the query interface required for data sharing and access based on metadata that can provide detailed and various functions. In this paper, we propose eXMDR-DAI (Extension of XMDR-DAI), which is an extended interface that allows inter-system data interoperability considering the relationship between metamodel classes. This solves the problem of meta information collision in interoperability between information systems in a big data based cloud environment. It also supports the efficient operation of information system processes. EXMDR-DAI is designed for enterprise-class business cloud environments. This is a business process that integrates data between independently operated local databases. And middleware that can handle collection, migration and integration. The composition of this paper is as follows. Section 2 introduces the related work and Section 3 describes the eXMDR-DAI configuration. In Chapter 4, comparative evaluation is conducted to verify the feasibility, and conclusions and future studies are described in Chapter 5

## RELATED WORK

XMDR-DAI [6, 7] is a middleware that supports data interoperability in the business collaboration process in a cloud environment. It is based on metadata schema information and consists of *MSO*, *InSO*, *MLoc* and *MDR* as shown in Fig 1. This thesaurus metadata schema information of local databases. And we have solved the problem of heterogeneous collision between metadata by mapping the metadata schema specified as standard. In addition, *MLoc* registers the physical location and access rights information of the local databases in conjunction with the MSO. Then, the

data migration and transaction process required for data access are defined so that the business process message can be transmitted to the corresponding location. Also, *InSO* is a thesaurus constructed by mapping association information between actual instance values and defined by considering semantic, similarity, and validity among instance values. Finally, *MDR* is managed by registering the metadata object schema of each local database. It consists of global schema area and local schema area.

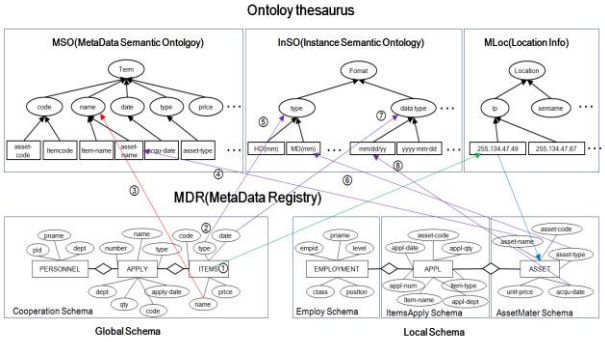


**Figure 1.** Composition of XMDR-DAI

## EXTENSION OF XMDR-DAI

This chapter describes the design model of eXMDR-DAI (Extension of XMDR-DAI) which improves the problem of XMDR-DAI. The concept defined in XMDR-DAI in Section 2.2 does not consider the relationship between classes. Each schema provided only a mapping definition type corresponding to a heterogeneous conflict. In other words, since it does not reflect various relationship information between classes defined in XMDR-DAI, additional cost is required when users want to access data from various classes. Figure 2 is a metadata model that includes the inheritance relationship between classes and classes defined in XMDR-DAI. *MSO_Item* inherits *MetadataSchema_Item*, and *InSO_Item* and *MLoc_Item* are subclasses of *MSO_Item*. Thus, various relationship information is defined in XMDR-DAI. However, the XMDR-DAI maps classes to a single user-defined type and does not design a way to deal with the relationships between classes.

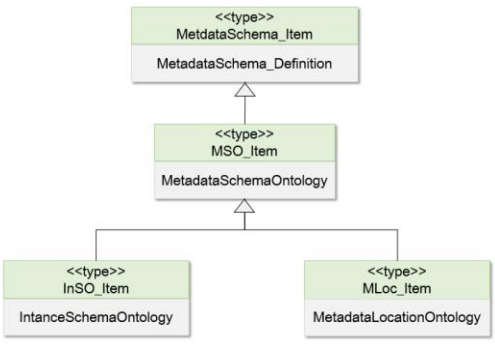


**Figure 2.** Part of the XMDR classes

As shown in Figure 2, inheritance relation information is missing because the relationship information between *MSO_Item* class and *MetadataSchema_Item* class is not defined. An additional interface implementation is required for the user to retrieve the information of the superclass. Therefore, the proposed eXMDR-DAI should reflect the relationship information existing in XMDR so that this additional cost does not occur. The following describes the relationship information in eXMDR-DAI presented in this paper in detail and presents the design configuration.

### Nested structured type

One type may have multiple attributes, each with its own data type. The data type is divided into "Predefined type", that is, a predefined type and a user defined type. A predefined type is a type defined by default in the database system, such as integer, char, and date. However, the *MutiMedia_Content* type described in Figure 3 is made up of attributes having a user-defined type, not a predefined type. In other words, Picture-Format and *Video_Encoding* are nested structure types, and nested structure types can be used as attributes of tables as well as other types of attributes. Multiple types of enclosing types can be defined in a single user type, and new enclosing types can be created in the enclosing type. This *MutiMedia_Content* type appears to have two attributes. However, since each property is defined as a user-defined type, it includes all the properties of Picture-Format and *Video_Encoding* type.

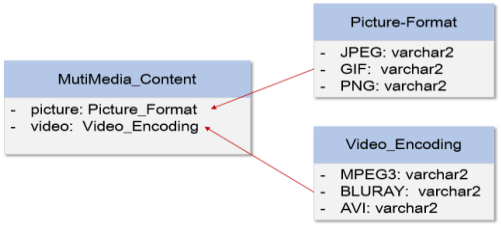


**Figure 3.** Example of nested structure type relationship

### Association-Relation

This eXMDR-DAI expresses relationships between classes through Association, Aggregation, Composition, and Inheritance. This section describes the relationship between the classes represented in eXMDR-DAI with each example and the solution for the interface.

**a) Association - Relationship processing method.**

Figure 4 shows an example of association-relationship. To apply an association-relationship to a local database, a primary key and a foreign key constraint are required. However, primary key and foreign key constraints cannot be specified when creating a user-defined type, and key constraints can be

specified only when declaring a typed table, which is a table based on a user-defined type. A type table is for storing actual data in a physical database based on a user-defined type. Table 1 is a type table based on a user-defined type.

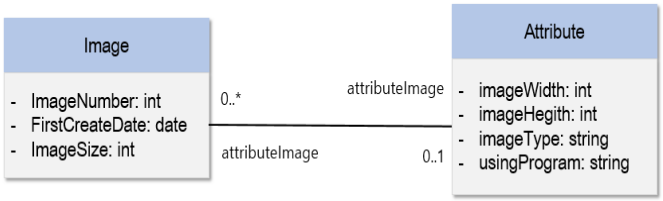


**Figure 4.** Example of association-relationship

**Table 1.** Creation example multimedia_PK

```
create type multimedia_PK as Entity (
field_01 number,
    field_02 varchar2(40),
    field_03 blob
);
```

```
create table multimedia_table_PK of multimedia_PK;
```

Since the generated type table is a table based on the *multimedia_PK* type, it includes all the attributes of the *multimedia_PK* type. \Also, you cannot add any other attributes other than the type of the *multimedia_PK* type that is the underlying type when creating the type *table multimedia_table_PK*. Table 2 shows examples of applying primary key and foreign key constraints to the declared type table *multimedia_PK* table. And field_01 specified in the *multimedia_PK* type as the primary key.

**Table 2.** Creation example multimedia_table_PK

```
create table multimedia_table_PK of multimedia_PK (
    primary key (field_01)
);
```

Because you can not specify a primary key when you create a custom type, you specify the primary key when you declare a typed table. The following is an example of a foreign key definition referring to the primary key specified in Table 1. The *multimedia_table_FK* table is a type table based on the *multimedia_FK* type and has two attributes field_1 and field_2. In Table 3, of these two attributes, field_2 refers to the primary key of *multimedia_PK*. Therefore, use foreign key (field_2) syntax to specify field_2 as foreign key.

**Table 3.** Creation example multimedia_table_FK

```
create type multimedia_FK as Entity (
    field_01 number,
    field_02 varchar2(40),
);
```

```
create table multimedia_table_FK ( foreign key (field_02) references multimedia_table_PK (field_01)
ON DELETE SET NULL);
```

Finally, by adding the 'ON DELETE SET NULL' constraint, when the primary key is deleted, a constraint is added that initializes the foreign key values of the associative table to NULL values. This will change the foreign key values to NULL when deleting the relationship between the two tables. Figure 5 shows an example of the association-relationship existing in eXMDR-DAI.

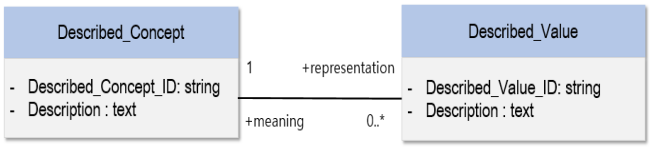


**Figure 5.** Example of association relationship in the XMDR-DAI

In Figure 6, the *Described_Concept* class has the *Described_Concept_ID* attribute and the *Described_Value* class has the *Described_Value_ID* attribute respectively. Since there is no specification of primary key and foreign key in current standard XMDR, in this paper, ID value that can identify each class is generated and defined as primary key. It also creates a foreign key to reference the primary key to handle the primary key and the foreign key relationship.

**b) Aggregation-associations and Composition-associations processing method**

Aggregation and Composition are a special form of association-relationship. It is necessary to define the primary key and the foreign key in order to express the aggregation of the aggregate in the database to guarantee the independence of life cycle of each class. Add an 'ON DELETE CASCADE' constraint to express complex associations. The 'ON DELETE CASCADE' constraint is a constraint that deletes the foreign key when the primary key is deleted. It reflects the characteristics of the complex association with the life cycle of the whole class and partial class being the same. Figure 6 is an example of aggregate and complex associations included in the eXMDR-DAI metamodel. As mentioned earlier,

relationships cannot be reflected when creating each type. Therefore, each type of typed table is created, and at the same time, the relationship is handled by specifying the primary key and the foreign key and adding the constraint.

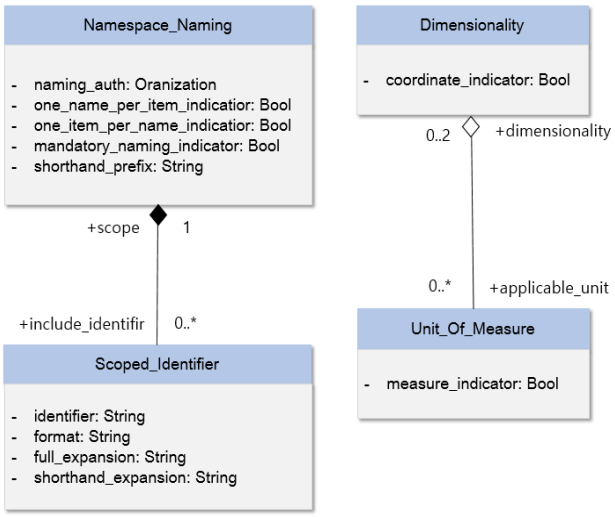


**Figure 6.** Example of aggregation and composition relationship in the XMDR-DAI

### b) Inheritance relationship processing method

The inheritance relation, which is one of the most important concepts of object orientation, is defined as follows. If one class (*SubClass*) inherits another class (*SuperClass*), the inherited *SubClass* means that the function of *SuperClass* can be used as it is. In addition, *SubClass* defines its own functions in addition to functions inherited from *SuperClass*. Figure 7 is an example of inheritance. The *Sub_Class* class includes all of the attributes *Super_Field1*, *Super_Field2*, and *Super_Field3* included in *Super_Class*. It also includes *Sub_Field1* and *Sub_Field2*, which are attributes inherent to *Sub_Class*.

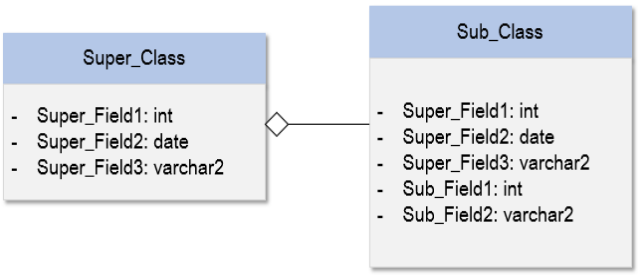


**Figure 7.** Example of inheritance relationship

Figure 8 is an inheritance in MRA-based XMDR-DAI. The *Registration_Auth* class, which inherits the Organization class, has attributes owned by the Organization class, as well as its own properties. Then, create a *Registration_Auth* type that inherits the Organization type. The generated *Registration_Auth* type includes the attributes of Organization, which is the parent type, as well as the *registration_auth_identifier* and *documentation_lang* attributes it contains. Apply primary and foreign keys to SuperClass and SubClass to make inheritance stronger. In this way, SubClass not only includes the structure of SuperClass, but also can associate with SuperClass.

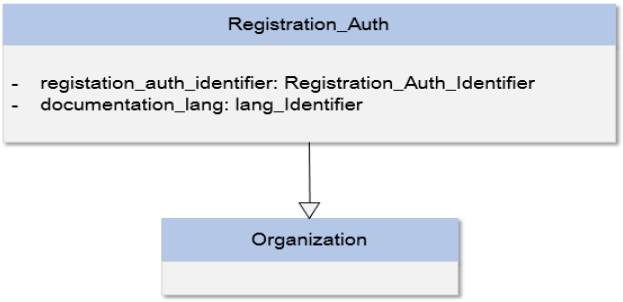


**Figure 8.** Inheritance relationship in the XMDR-DAI

## EVALUATION

The following prerequisites are defined for the experiment. First, XMDR and eXMDR-DAI have the same conditions except for the existence of relationship information. The remaining conditions include both the name and number of the table, the name and number of the existing attribute in the table, and the stored data. In addition, the classes defined in the XMDR standard are built into a single table. The query to be executed on the system is defined as a simple search query and a complex search query. Complex search queries include association, aggregation-association, complex-association, and inheritance. A simple query is a query that retrieves data from a single table. A complex search query, on the other hand, is a query that retrieves data by taking into account relationships in two or more tables. Therefore, if an XMDR system performs complex search queries to obtain accurate results, additional operations are required to refine the result data.

### Query Modeling Time

This section defines the process to be performed on the actual system based on the query type defined in Section 3.2. In addition, the defined process is formulated to compare query processing efficiency using XMDR and eXMDR-DAI. The experiment measures the time through the simulation of the formulated query modeling process. The user performs the following four steps in order to query the system. 1) Metadata extraction: The process of analyzing meta-model information. 2) Checking relationship information: The process of accessing the actual database system to search for relations and attributes, and to verify that relationship information is defined. 3) Creating Relationship Information: The process of defining relationship information in the system when relationship information is not defined. 4) Query Generation:

The process of performing the query. In the process of generating a query, it is the whole process from inputting a query to an actual database system to deriving a result value. To illustrate this process, Table 4 shows the variable definitions for comparing the system with XMDR applied and the system querying with eXMDR-DAI.

**Table 4.** Variable description for comparing the system with XMDR applied and the system querying with eXMDR-DAI

| Variable Name | Variable Description |
|---|---|
| *QMT 1* | Query Modeling Time |
| *TMA 2* | Time required for metadata model analysis |
| *NRQ 3* | Number of relationships included in the query |
| *DRMT 4* | Time comparison of database relationships and metadata model classes |
| *MADRT 5* | Time comparison of class attribute and database relationship property of metadata model |
| *IRT 6* | The time required to identify the relationship request is defined in the database. |
| *RCT 7* | Relationship creation time |
| *QET 8* | Query execution time |

The procedure for querying the system with eXMDR-DAI using the variables defined in Table 4 is shown in Equation 1. This includes query modeling, metamodel analysis, and access to the actual database system to validate relations and attributes, and query execution. The time to make sure XMDR applied system is a defined relationship information because it does not reflect the relationship information is required. Furthermore, additional time is required to define the relationship between the required numbers if you do not have a defined relationship information. This time increases as the number of relation information included in the query increases.

$$QMT = TMA + DRMT + MADRT + \sum_{i=1}^{NRQ} IRTi + \sum_{j=1}^{NRQ} RCTj + QET \quad \ldots\ldots\ldots (1)$$

The eXMDR-DAI proposed in this paper shortens the query modeling time because it already reflects the relationship information of the metadata. Equation 2 is a formula for calculating query time on a system with eXMDR-DAI applied. Since this reflects the relationship information, it is possible to omit the procedure for confirming and creating the relationship information. Therefore, after analyzing the metamodel, querying can be performed as soon as the process of accessing the local database system and checking the relation and attributes is completed. In query modeling of a system with eXMDR-DAI, the number of relation information does not affect the overall modeling time. This reduces the overall query modeling time.

$$QMT = TAM + DRMT + MADRT + QET \quad \ldots\ldots..(2)$$

Figure 9 is a graph comparing query modeling time according to the number of relation information. Since the query modeling time is dependent on the individual differences in the intervening process, the variables used in the formula are tested with a random value between 0 and 1. In order to obtain more reliable results through sufficient simulation, a total of 100 experiments are performed to calculate the average value. Since eXMDR-DAI reflects the relationship information of metadata, it is necessary to check whether relation information is defined and to omit time ($\sum_{j=1}^{NRQ} RCTj$, $\sum_{i=1}^{NRQ} IRTi$) to generate. Therefore, the total query modeling time does not increase even though the number of relation information NRQs included in the query increases. On the other hand, the existing model XMDR increases the query modeling time because $\sum_{j=1}^{NRQ} RCTj$ and $\sum_{i=1}^{NRQ} IRTi$ increase as the number of NRQ increases. Experimental results show that the larger the number of relation information, the larger the difference is, and the eXMDR-DAI is not affected by the number of relations information.

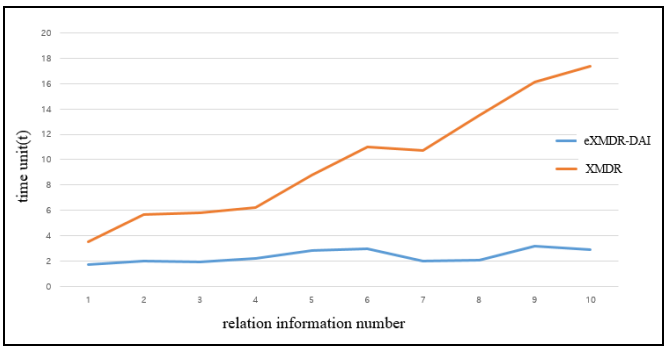


**Figure 9.** Time comparison of query modeling in XMDR and eXMDR-DAI

## CONCLUSION

This paper proposes extended interface eXMDR-DAI. In eXMDR-DAI, we implemented branch relationship information (association, aggregation association, complex association, inheritance) to the system. In order to evaluate the query processing efficiency of the proposed eXMDR-DAI, we compared it with the existing XMDR that does not reflect relationship information. We define the query modeling time and the number of query filtering. eXMDR-DAI showed the effectiveness of query modeling by omitting the relationship information confirmation step and generation time by reflecting metamodel relationship information. In addition, the

reference integrity of the system is guaranteed by reflecting the relation information, and when compared with the basic XMDR, it shows high accuracy when performing complex search query. In the future, we need to develop an implementation model using a more diverse database model and database management system, and study the features of each model. We also provide data access for each local database participating in collaborative work to meet the requirements for efficient operation of enterprise business processes. This should be studied to facilitate data collection, movement, and merging with data integration using standard queries.

**REFERENCES**

[1] KECK, K. D.; MCCARTHY, J. L. XMDR: Proposed Prototype Architecture Version 1.01 (February 2005). 2005.

[2] Bargmeyer, B. "eXtended Metadata Registries (XMDR)." Presentation at the 7th NKOS 2005. 2005.

[3] J. Davies, S. Harris, Ch. Crichton, A. Shukla, J. Gibbons, “Metadata standards for semantic interoperability in e-government,” Proc. of the 2nd International Conference on Theory and Practice of Electronic Governance, pp.67-75, 2008.

[4] Moon, SeokJae, GyeDong Jung, and YoungKeun Choi. "A Study on Cooperation System design for Business Process based on XMDR in Grid." International Journal of Grid and Distributed Computing 3.3 (2010): 1-12.

[5] L. Bountouri, C. Papatheodorou, V. Soulikias, M. Stratis, “Metadata interoperability in public sector information,” Journal of Information Science, Vol.35, No.2, pp.204-231, 2008.

[6] Lee, Jong-Sub, Sung-Uk Choi, and Seok-Jae Moon. "Mobile Cloud Hospital Information System based on XMDR-DAI." (2016).

[7] Jung, Kye-Dong, Seok-Jae Moon, and Jin-Mook Kim. "Data access control method for multimedia content data sharing and security based on XMDR-DAI in mobile cloud storage." Multimedia Tools and Applications (2016): 1-17.

[8] Australian Institute of Health and Welfare, Metadata Online Registry (METeOR), http://meteor.aihw.gov.au/ (accessed September 2012).

[9] US Health organizations (multiple), US Health Information Knowledgebase (USHIK), http://ushik.ahrq.gov (accessed September 2012).

[10] US Environmental Protection Agency, Environmental DataRegistry, http://www.epa.gov/edr/ (accessed September 2012).

[11] US National Cancer Institute, Cancer Data StandardsRepository (caDSR), http://ncicb.nci.nih.gov/ NCICB/(accessed September 2012).

[12] U.S. Department of Homeland Security (DHS) and U.S. Department of Justice (DOJ), US National Information Exchange Model (NIEM), http://www.niem.gov/ (accessed September 2012).