Semantic Information Gathering Approach for Heterogeneous Information Sources on WWW

Ngamnij Arch-int
Peraphon Sophatsathit

Advanced Virtual and Intelligent Computing (AVIC) Research Center
Department of Mathematics, Faculty of Science, Chulalongkorn University, Bangkok, Thailand

Email: ngamnij@kku.ac.th, Peraphon.S@chula.ac.th

Abstract

The increasing demand for accessing heterogeneous information sources to support global applications and decision making requirements forces organizations to solve heterogeneity problems. One of the important problems stemming from accessing the heterogeneous data is semantic heterogeneity. A number of research efforts have been proposed to address this problem, ranging from mediator-based systems, description logic-based systems to content-descriptive metadata systems. In this paper, we propose a metadata dictionary as an assistant mechanism for solving semantic heterogeneity. The proposed metadata dictionary is designed based on domain ontology where the constituent components are defined in terms of object-oriented and set theory. An XML-based data model is employed to manipulate and express the metadata dictionary contents. The inherent flexibility of XML technology permits system-wide interoperability suitable for a Web-based environment.

Keywords: Heterogeneous Information Sources, Domain Ontology, XML-based Metadata Dictionary.

1 Introduction

Due to the increasing demand in gathering and integrating data from the existing Heterogeneous Information Sources (HIS) to achieve semantic interoperability, organizations have to solve various semantic heterogeneity [29] problems. These problems can be classified into four types as follows:

- *Naming conflicts*, encompassing two different kinds of conflict, namely, synonyms and homonyms. Synonyms are concerned with the semantically equivalent concepts (i.e., classes) or properties (i.e., attributes) defined by different names. Homonyms, on the other hand, are concerned with the semantically unrelated concepts or properties defined by the same name;
- *Data type conflicts*, concerning semantically equivalent properties defined with different data types;
- *Scaling conflicts*, concerning semantically equivalent properties defined with different scales (or units of measure); and

- *Generalization conflicts*, concerning semantically related concepts defined in different systems where the concepts in one system subsume the concepts in another system.

A number of systems have been proposed to cope with these problems. For example, the mediator-based systems [37] provide the inter-schema architecture for integrating access to data of different sources and converting data and queries into canonical formats via the mediator and wrapper components. Examples of such systems are TSIMMIS [16] and HERMES [1].

The description logic-based systems offer a different approach to elaborate source description by means of description logic [7, 8] for solving queries over multiple sources. Unlike the mediator approach, the description logic approach abstracts the heterogeneous sources from users through a global view. Examples of such systems are the Information Manifold [25] and the SIMS [3, 4].

The content-descriptive metadata systems [23] utilize annotation information that is tightly integrated with HTML as metadata to describe the contents of a Web document. Examples of such systems are the Ontobroker [12] and the SHOE [19].

All of the above systems employ the ontology approach as described in [14, 31, 33] to cope with heterogeneity problems. A survey and comparison of these systems can be found in [27, 36].

In this paper, we propose a metadata dictionary extended from the reference architecture proposed by [2] as a means for solving semantic heterogeneity problems. We consider the HIS consisting of structured data sources, such as database systems, and semi-structured data sources, such as XML documents [35]. The proposed metadata dictionary is modeled and designed based on domain ontology [17] to be a repository for storing conceptual level and physical level data descriptions, where the constituent components are defined in terms of object-oriented and set theory. In order to support system-wide interoperability suitable for a Web-based environment, we choose XML as a language for expressing the metadata dictionary contents, as well as providing flexibility and scalability in building and manipulating the ontology terminologies. These ontology terminologies are subsequently shared by the agents to access and retrieve real data from the underlying physical sources.

The main functionalities of metadata dictionary can be summarized as follows:

- Providing an abstract view for the application domain through the virtual schema on which users can pose their queries expressed over this virtual schema. The virtual schema abstracts the users from the underlying physical sources;
- Providing a mapping mechanism to translate the virtual schema into the associated physical schema; and
- Providing the physical source configurations that are necessary for search agents in accessing the HIS, such as physical source and concept (or entity) names, network location of each physical source, data model, query language, owner and permission of each physical entity, etc.

Our approach differs from other approaches from various standpoints, that is,

- The proposed approach provides a metadata dictionary as a knowledge repository that is flexible for agents to acquire knowledge dynamically, that is, agents are able to obtain knowledge from the metadata dictionary instead of predefined static source for each agent. This provision offers update flexibility of the knowledge in the metadata dictionary;
- A user's query posed over virtual schema is mapped directly to physical schema without loss of information in the query, that is, the user's query need not be rewritten in such a way to accommodate one ontology to another stored in dispersed sources, thus eliminating potential loss of information in query transformation process;
- The proposed approach defines domain ontology components based on object-oriented and set theory that aim to be a standard model which can be applied to real world metadata dictionary implementation by means of independent tools and languages; and
- The domain ontology is expressed in XML-based architecture that is easy for agents to gain the information from the metadata dictionary. This representation provides a means for consolidating data retrieved from various sources, while retaining consistent identification of the data semantics. Such a configuration is suitable for representing data from HIS in a Web-based environment.

The remainder of the paper is structured as follows. Section 2 presents a metadata dictionary based on ontology modeling technique. Section 3 presents the structuring of the XML-based metadata dictionary from domain ontology components. The XML-DTD metadata obtained in the process is also illustrated. Section 4 demonstrates how the proposed metadata dictionary solves semantic heterogeneity. Section 5 suggests further research extension. Section 6 summarizes the proposed work.

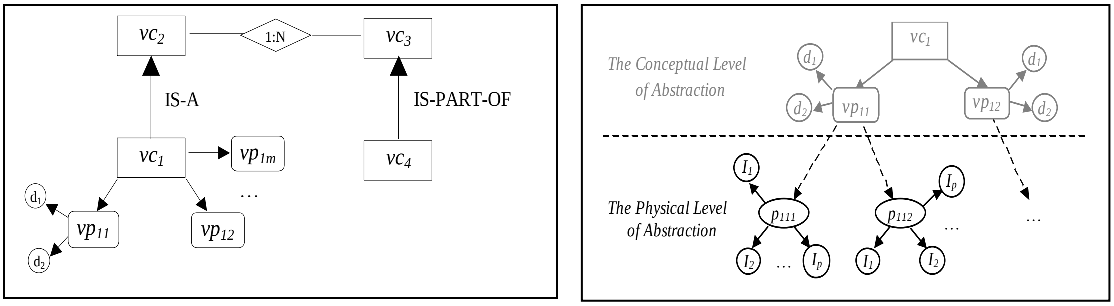
2 The Ontology-based Metadata Dictionary

In this section, we focus on modeling and designing the domain ontology, which is the fundamental building block of the metadata dictionary instead of the straightforward ontology construction [18, 22, 32, 33]. The resulting model is thus utilized in formal definitions of domain ontology components as illustrated in an Appendix.

Domain ontology has been modeled on the basis of a bottom-up design approach [10, 26, 34]. The modeling process involves the schema translation of the underlying physical information sources into the intermediate schemas via the E-R model [11], and schema integration of these intermediate schemas into a global conceptual schema in order to eliminate structural heterogeneity [5, 6, 26]. Details of schema translation and integration can be found in [28, 30].

Our approach focuses on extracting the ontology from the underlying global conceptual schema to obtain an explicit user-viewed representation. The ontology is systematically extracted into two levels of abstraction, namely, the conceptual level of abstraction and the physical level of abstraction.

(1) The conceptual level of abstraction. The global conceptual schema is restructured into virtual schema, which is an initial ontology represented by Extended Entity-Relationship (EER) model encompassing virtual concepts (or entities), virtual properties (or attributes), relationships, and construction rules. The ontology conceptualized on this level abstracts the users from physical information sources. Users can pose their queries in the form of this ontology rather than dealing with real data. A partial internal structure of domain ontology at this level is depicted in Figure 1 (a).



(a) Domain ontology at the conceptual level of abstraction.

(b) Domain ontology at the physical level of abstraction.

Figure 1. Two levels of domain ontology extracted from a global conceptual schema.

In this figure, boxes represent virtual concepts, whereas diamonds denote the relationships that hold among the virtual concepts. The virtual properties are shown as rounded rectangles attached to each virtual concept. In our model, we designate a virtual property as a class property, which forms its own property set called domain properties. The domain properties are represented by circles that encompass predefined type domains (e.g., integer, string, float, or char) and scaling domains or units of measure (e.g., kilogram, pound, US$, or AUS$). These domain properties are used to solve data type and scaling conflicts in which the same logical data items with different physical data types or unit types from HIS can be displayed in a uniform format. The relationships, such as IS-A, IS-PART-OF relationships, etc., link virtual concepts to exhibit their relationships. IS-A relationship is used to solve generalization conflicts and denoted by an arrow connecting a specific concept to a general concept. IS-PART-OF relationship is denoted by an arrow connecting a component concept to an aggregate concept. The construction rules are augmented from the diagram.

(2) The physical level of abstraction. This level provides a mapping mechanism to associate the virtual concepts and properties of a virtual schema with the corresponding physical concepts and properties of a global conceptual schema. A partial internal ontology structure is illustrated in Figure 1 (b). At this level, each virtual property is designed to hold its instances, called physical instances represented by ellipses to store the synonymous physical property names of the physical concepts in a global conceptual schema. Each physical instance defines its own properties, denoted by circles that encompass other physical information corresponding to the physical instance, such as physical data type, unit type, concept, and source. These physical instances are used to solve naming conflicts. The ontology on this level also holds physical source configurations describing the configurations of physical concepts in each physical source. These physical source configurations furnish necessary information to grant permission and knowledge for agents in accessing individual physical source.

3 The XML-based Metadata Dictionary

We exploit XML strengths in well-formedness, validity, and schema to represent the metadata dictionary. It is imperative that an XML document needs to be validated by rules defined through XML-DTD (Document Type Definition) representing XML data schematic description.

In the following sections, we structure the XML-DTD from the domain ontology components with a list of legal elements and attributes. The resulting XML-DTD will also be illustrated.

3.1 Structural Design of XML-DTD from Domain Ontology Components

The structural design of XML-DTD was set up to maintain their conceptual and physical correspondence and consistency into two levels as follows.

3.1.1 The conceptual level of design abstraction

To capture the semantic elements of the conceptual level modeling, the XML-DTD in this level must encompass all virtual concepts and their corresponding virtual properties and relationships. The resulting structure of the XML-DTD in this level is depicted in Figure 2, where each rectangle denotes an XML element or sub-element and each double-lined rounded rectangle represents an XML attribute within an element. White areas of nodes contain element or attribute names and shaded areas of nodes hold data elements or attribute values. A bracketing symbol of an element indicates that there are sub-elements within that element. A (+) symbol in front of an element indicates that there are one or more instances of that element, whereas a (?) symbol indicates zero or one instance of the element. Any element without a symbol represents exactly one instance of that element. The data elements and attribute values can be atomic values denoted by *string*, unique identifiers denoted by *id*, or identifier references denoted by *idref*. The identifier is used to reference an *id* of another element shown as a dashed arrow.

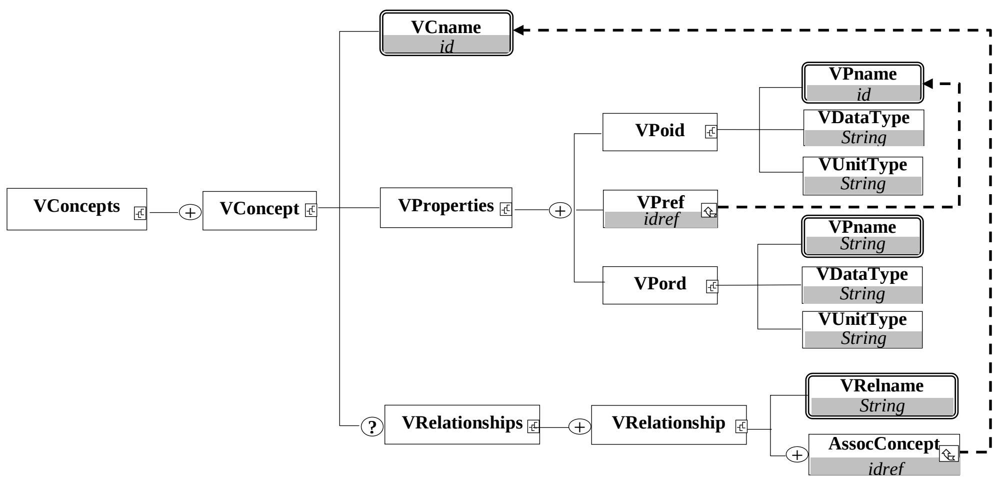


Figure 2. The XML-DTD structure at the conceptual level of design abstraction.

The XML-DTD structure in this level is described below. Structural domain ontology representation of components is given in the Appendix.

(1) Virtual concepts

- The root element `VConcepts` consists of one or more sub-elements `VConcept` denoting virtual concepts whose names are stored in the attribute `VCname` of `VConcept`. Each `VConcept` in turn consists of a sub-element `VProperties` denoting virtual properties and zero or one sub-element `VRelationships` denoting virtual relationships.

(2) Virtual properties

The element `VProperties` of each `VConcept` contains one or more sub-elements `VPoid`, `VPref`, and `VPord`.

- Each VPoid denotes an object identifier property or key. The name of VPoid is designated to the attribute VPname of VPoid.
- Each `VPref` denotes an object reference property or foreign key designated to store a virtual property name to its data element. Each data element of `VPref is` defined as *idref* to reference the virtual property name defined as *id* in `VPoid`.
- Each VPord denotes an ordinary property whose value is atomic value (e.g., integer, string). The name of VPord is designated to the attribute VPname of VPord.
- Each `VPoid` and `VPord` consists of sub-elements, `VDataType` and `VunitType`, whose data elements are designated to store the virtual data type (or predefined type domain) and unit type (or scaling domain), respectively. Note that a NULL value in a `VUnitType` element designates the property that is not of unit of measure.

(3) Relationships

- Each `VConcept` can associate with zero or more concepts, whose names are designated to the data elements of `AssocConcept`. The associated concept name of `AssocConcept` is defined as *idref*, pointing back to the already defined concept name in `VConcept`.
- The types of relationship, that is, associative, IS-A, and IS-PART-OF relationships, between `VConcept` and `AssocConcept` are designated to the attribute `VRelname` of `VRelationship`.

3.1.2 The physical level of design abstraction

The principal constituents of this level consist of the physical property names and other related physical information that are connected with the virtual properties and concepts in the conceptual level. This physical level is designed to incorporate the physical source configurations of physical concepts and sources. The overall XML-DTD structure is illustrated in Figure 3, where bold pictures denote the physical level of abstraction.

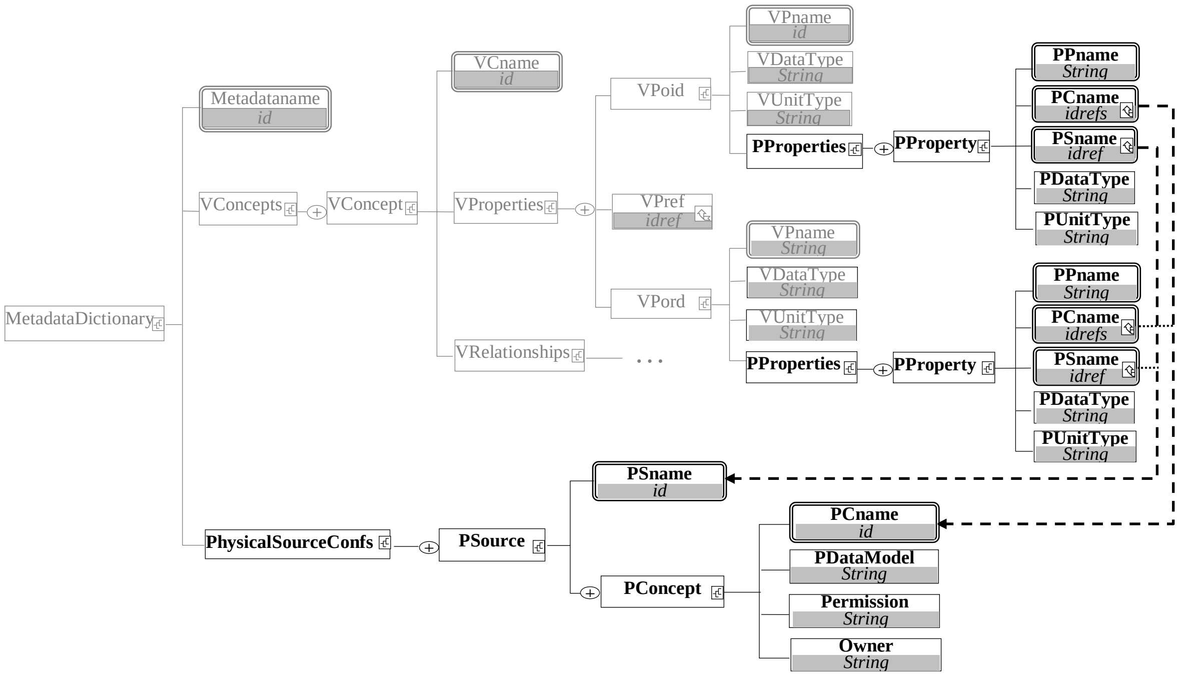


Figure 3. The XML-DTD structure at the physical level of design abstraction.

The XML-DTD structure in this level can be described as follows:

(1) Physical properties and other physical information

- Each `VPoid` and `VPord` at the conceptual level is designated to contain one or more synonymous physical properties, whose names are designated to the attribute `PPname` of `PProperty` of `VPoid` and `VPord`, respectively.
- Other physical information related to the physical property names of `VPoid` and `VPord`, such as the physical data type and unit type, are designated to the data elements `PDataType` and `PUnitType`, respectively. Similarly, the physical concept names and source names are designated to the attribute `PCname`, and `PSname`, respectively. The attribute `PCname` and `PSname` are defined as *idref* to reference the physical concept and source names defined as *id* in the physical source configurations.

(2) Physical source configurations

- The element `PhysicalSourceConfs` consists of one or more sub-elements `Psource`, whose names are designated to the attribute `PSname` of `PSource`.
- Each `PSource` consists of one or more sub-elements `PConcept`, whose names are designated to the attribute `PCname` of `PConcept`. The values of other physical configurations that associate to each physical concept (e.g., physical data

model, permission, owner) are designated to the data elements of `PDataModel`, `Permission`, and `Owner`, respectively.

3.2 XML-DTD Metadata Dictionary Structure

The structure of XML-DTD metadata dictionary, depicted in Figure 4, is constructed based on the design abstractions described in the earlier section and ordered according to Figure 2 and 3.

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE     MetadataDictionary [
   <!ELEMENT  MetadataDictionary     (VConcepts, PhysicalSourceConfs)>
   <!ATTLIST  MetadataDictionary     MetadataName   ID       #REQUIRED>
   <!ELEMENT  VConcepts              (VConcept)+>
   <!ELEMENT  VConcept               (VRelationships?, VProperties)>
   <!ATTLIST  VConcept               VCname         ID       #REQUIRED>
   <!ELEMENT  VRelationships         (VRelationship)+>
   <!ELEMENT  VRelationship          (AssocConcept)+>
   <!ATTLIST  VRelationship          VRelname (IS-A|IS-PART-OF|Associative) #REQUIRED>
   <!ELEMENT  AssocConcept           (#PCDATA)>
   <!ATTLIST  AssocConcept           VConcept       IDREF    #IMPLIED>
   <!ELEMENT  VProperties            (VPoid|VPord|VPref)+>
   <!ELEMENT  VPoid                  (VDataType, VUnitType, PProperties)>
   <!ATTLIST  VPoid                  VPname         ID       #REQUIRED>
   <!ELEMENT  VPord                  (VDataType, VUnitType, PProperties)>
   <!ATTLIST  VPord                  VPname         CDATA    #IMPLIED>
   <!ELEMENT  VPref                  (#PCDATA)>
   <!ATTLIST  VPref                  VPoid          IDREF    #IMPLIED>
   <!ELEMENT  VDataType              (#PCDATA)>
   <!ELEMENT  VUnitType              (#PCDATA)>
   <!ELEMENT  PProperties            (PProperty)+>
   <!ELEMENT  PProperty              (PDataType, PUnitType)>
   <!ATTLIST  PProperty              PPname         CDATA    #REQUIRED
                                     PCname         IDREFS   #REQUIRED
                                     PSname         IDREF    #REQUIRED>
   <!ELEMENT  PDataType              (#PCDATA)>
   <!ELEMENT  PUnitType              (#PCDATA)>

   <!ELEMENT  PhysicalSourceConfs    (PSource)+>
   <!ELEMENT  PSource                (PConcept)+>
   <!ATTLIST  PSource                PSname         ID       #REQUIRED>
   <!ELEMENT  PConcept               (PDataModel, Permission, Owner)>
   <!ATTLIST  PConcept               PCname         ID       #REQUIRED>
   <!ELEMENT  PDataModel             (#PCDATA)>
   <!ELEMENT  Permission             (#PCDATA)>
   <!ELEMENT  Owner                  (#PCDATA)>
]>
```

Figure 4. The XML-DTD metadata dictionary structure.

3.3 Construction Rules

In order to govern the update operations of the well-formed and valid XML document, a set of construction rules is set up to administer the correctness and consistency. Since XML document is tree-structured, the metadata dictionary contents are represented in a conventional tree structure. The following formulations outline the rules in constructing the XML metadata dictionary document.

Let vc_m and vc_n be virtual concepts in domain ontology. The virtual concepts and relationships make up the nodes and links of the XML document tree as follows:

(1) Virtual concepts

Referring to the XML-DTD structure in Figure 2, the `VConcepts` starts the root of all virtual concepts. Other associated concept names stored in data elements of `AssocConcept` become the child nodes. The relationship between `VConcept` and `AssocConcept` denotes a one-way traversal. In other words, if $n(vc_m)$ and $n(vc_n)$ denote virtual concept names of vc_m and vc_n, and are designated to `VCname` and `AssocConcept`, respectively, it is not necessary to store $n(vc_n)$ and $n(vc_m)$ to `VCname` and `AssocConcept` in the reverse direction.

(2) Relationships

To properly link the virtual concepts, we employ the following formal guidelines to preserve the structural design established in Section 3.1 as follows:

- If a concept vc_m associates with vc_n through IS-A, IS-PART-OF, or Associative relationships, the $n(vc_m)$ and $n(vc_n)$ are designated to the attribute `VCname` and the data element `AssocConcept`, respectively.
- For the N:M relationship, it is important to note that the N:M relationship holding its own properties apart from the participating concepts must be separately accounted for as a virtual concept in the ontology structure. Therefore, if a concept vc_k is a separate concept representing a relationship between the participating concepts vc_m and vc_n, the $n(vc_k)$ and $n(vc_m)$ are designated to the attribute `VCname` and the data element `AssocConcept`, respectively, meanwhile, the $n(vc_k)$ and $n(vc_n)$ are also designated to the attribute `VCname` and the data element `AssocConcept`, respectively.

4 Case Study

4.1 An Example of Semantic Heterogeneity

To illustrate how the proposed metadata dictionary solves the semantic heterogeneity, we demonstrate two different physical information sources in a University referred to as a domain of discourse, as shown in Figure 5 (a) and (b). The first information source, `Source1`, shown in Figure 5 (a), is a relational data model encompassing two relations `Staff_Member` and `Department`. The second information source, `Source2`, shown in Figure 5 (b), is an XML-DTD storing the information of `Instructor_Member` and `Course`, as well as the relationship, `Course_Teach`, which links between `Instructor_Member` and `Course`.

Staff_Member

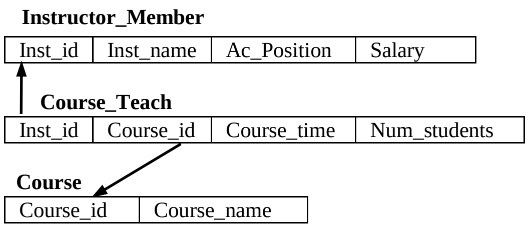
PhyProperty	PhyDataType	PhyUnitType
Staff_id	String	NULL
Staff_name	String	NULL
Salary	Float	US$
Dept_id	Integer	NULL

Department

PhyProperty	PhyDataType	PhyUnitType
Dept_id	Integer	NULL
Dept_name	String	NULL

(a) Physical information sources at Source1 represented by Relational Data model.

```
<?xml version="1.0"  standalone="yes"?>
<!DOCTYPE  Instructor_Course [
<!ELEMENT Instructor_Course        (InstructorMember_List,
Course_List)>
<!ELEMENT InstructorMember_List (Instructor_Member)+>
<!ELEMENT Instructor_Member  (Inst_name, Ac_Position, Salary,
Course_Teach)>
<!ATTLIST Instructor_Member Inst_id   ID   #REQUIRED>
<!ELEMENT Inst_name   (#PCDATA)>
<!ELEMENT Ac_Position  (#PCDATA)>
<!ELEMENT Salary   (#PCDATA)>
<!ELEMENT Course_Teach (Crs)+>
<!ELEMENT Crs (Course_time, Num_students)>
<!ATTLIST Crs   Course_id    IDREF  #REQUIRED>
<!ELEMENT Course_time (#PCDATA)>
<!ELEMENT Num_students (#PCDATA)>
<!ELEMENT Course_List (Course)+>
<!ELEMENT Course (Course_name)>
<!ATTLIST Course Course_id ID #REQUIRED>
<!ELEMENT Course_name (#PCDATA)>
]>
```



Relational schemas describing the relationships between concepts of XML-DTD in Figure 5 (b).

(b) Physical information source at Source2 represented by XML-DTD.

Figure 5. Two different data models of physical information sources.

This example not only illustrates the differences in data models and query languages, but also semantic heterogeneity. First, naming conflicts between attribute `Staff_id` in the relation `Staff_Member` of `Source1` and `Inst_id` in the element `Instructor_Member` of `Source2`. Both `Staff_id` and `Inst_id` are semantically equivalent properties, since they refer to the same fact. This is called a synonym conflict. Second, data type and scaling conflicts caused by the same attribute `Salary` of `Staff_Member` and `Instrutor_Member` have different predefined types and units of measure. Finally, generalization conflicts induce from the fact that the concept staff subsumes the concept instructor, since all instructors are staff. This example will serve as the basis for the ontology-based metadata dictionary design in the sections that follow.

4.2 Domain Ontology Representation

4.2.1 The conceptual level representation

The conceptual level of the ontology has been designed to solve data type, unit type, and generalization conflicts. The design is based on the proposed modeling technique outlined in Section 2 and illustrated by the EER model in Figure 6. Each virtual concept possesses its own virtual properties, for example, `Staff(st_id, st_name, st_salary, dept_id)`. The virtual property `st_id` is an object identifier or key, `st_name`, and `st_salary` are ordinary properties, and `dept_id` is an object identifier reference or foreign key. The virtual concept `Staff` relates to `Department` by an associative relationship. To solve data type and unit type conflicts, the object identifier and ordinary properties can further designate additional domain properties to specify predefined type and scaling domain. For example, the domain properties of `st_salary` are of the predefined type "Float" and scaling domain "US$". To solve generalization conflicts between the concepts `Staff` and `Instructor`, `Instructor` is designed to associate with `Staff` by an IS-A relationship, since `Instructor` is a subconcept of `Staff`. As such, `Instructor` inherits `st_id`, `st_name`, `st_salary`, and `dept_id` from `Staff`. Consequently, `Instructor` also associates with `Department` by an N:1 associative relationship.

As mentioned earlier, the relationship `course_teach` can define its own properties `crs_time` and `num_stu` in addition to those of the participating concepts `Instructor` and `Course`. Hence, the relationship `course_teach` is treated as a concept in the ontology.

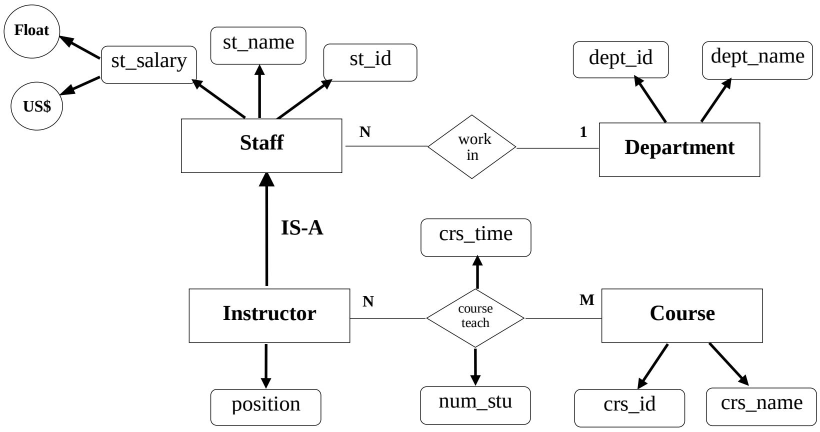


Figure 6. The logical ontology structure at the conceptual level of abstraction.

4.2.2 The physical level representation

The physical level of the ontology illustrated in Figure 7 is designed to solve the naming conflicts. Since synonym conflicts of the physical property `Staff_id` of `Staff_Member` and `Inst_id` of `Instructor_Member` are common encounters in HIS environment, synonym terms should be designed as the physical instances of the virtual property `st_id` through the instantiate relationships. Each physical instance, `Staff_id` for example, is the physical property name which can define its physical information properties for storing additional physical information associated with `Staff_id`. For example, the values of physical information properties named `PDataType`, `PUnitType`, `PCname`, and `PSname` of `Staff_id` are "integer," "NULL," "Staff_Member," and "Source1," respectively. This means that `Staff_id` is a physical property name having the physical data, and unit types, "integer," and "NULL," and the physical concept name "Staff_Member" which resides in "Source1."

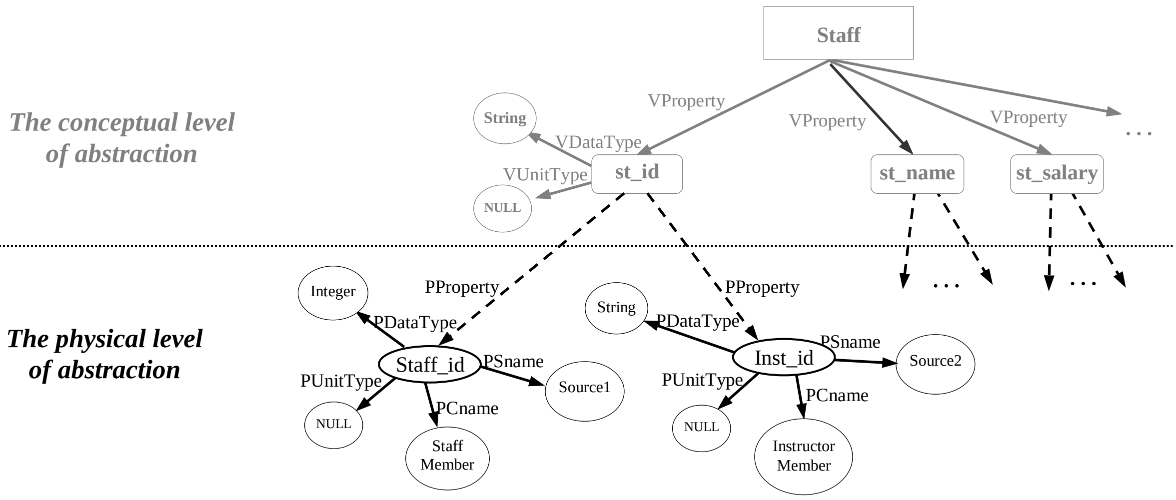


Figure 7. The portion of internal structure of the ontology at the physical level of abstraction.

4.3 The XML-based Metadata Dictionary Representation

The logical ontology structures based on existing entities in Figures 6 and 7 are translated into an XML-based metadata dictionary consisting of an XML-DTD (as shown in Figure 4) and an XML document. A partial XML document structure storing well-formed and valid data is given in Figure 8.

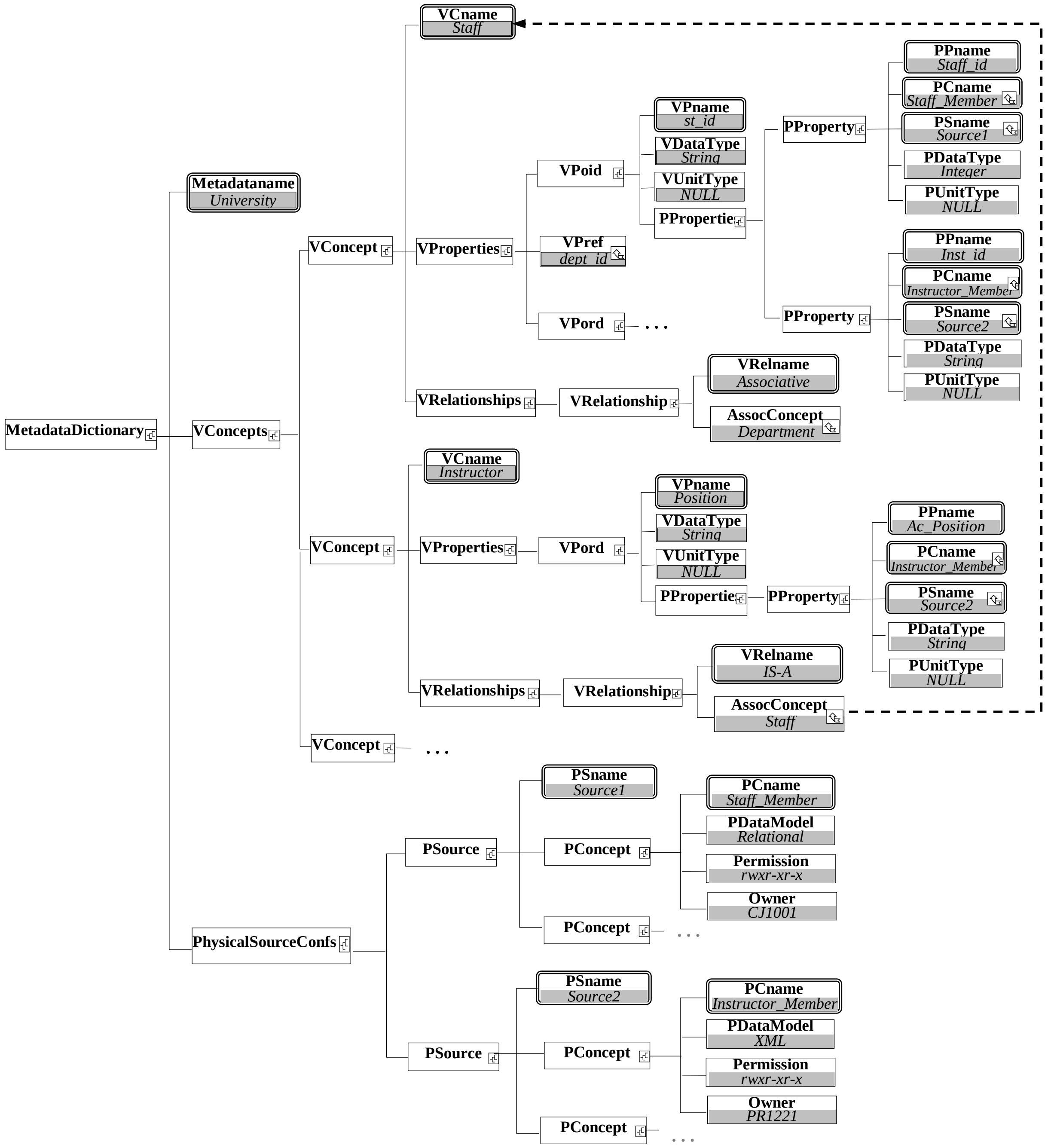


Figure 8. The portion of the XML document structure conforming to earlier XML-DTD.

In Figure 8, the IS-A relationship is transformed to a derived `VConcept` of its based `VConcept Staff` for relationship preserving and object derivation conformance. As such, the `VConcept Instructor` needs only define its own specialized properties, whereby all base relationships and properties are automatically inherited from its base class `Staff`. This fact reaffirms the proposed metadata dictionary principles of object orientation. The resulting XML document from Figure 8 is shown in Figure 9.

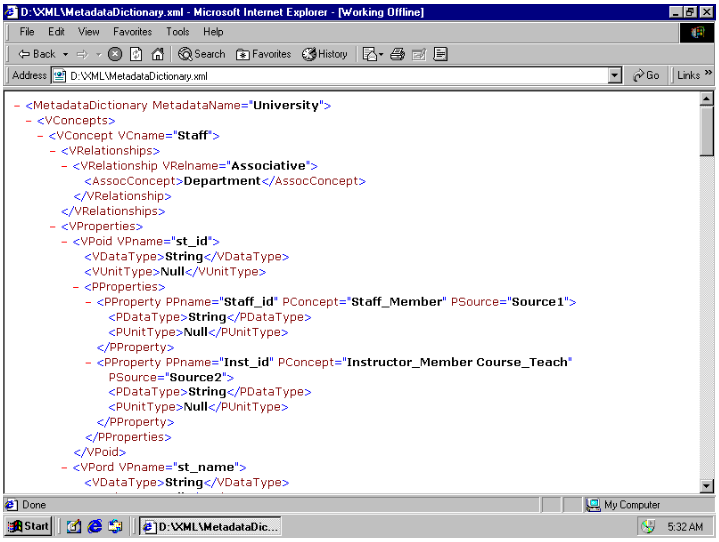


Figure 9. The portion of the XML document based on metadata dictionary.

We envision that the formality so introduced will enhance the formulation and design of more sophisticated metadata ontology-based components, in particular, rigorous verification that leads to correctness and reliable operations on HIS.

5 Future Work

Our future research will focus on querying and retrieving process model to access HIS via the mediator layer of the reference architecture [2] that involves primarily user interface agent, managing agent, and metadata dictionary. The unified access to HIS, by means of the metadata dictionary, will combine multiple schemas of various queries into an integrated schema. Some governing frameworks (in the form formal rules and algorithms) will be instituted to ensure that proper information is being stored and retrieved, whereby eliminating semantic heterogeneity.

Additional efforts will be placed on operational transformation and communication between the user interface agent and managing agent. A major task at this level is the decomposition of global transaction initiated by a user query into sub-transactions suitable for the underlying physical sources.

We will also enrich the proposed ontology-based metadata dictionary with advanced ontology language such as RDF/RDF Schema [24, 9] to enhance XML universal expressive power and syntactic interoperability [13] toward machine-processable Semantic Web [13, 20, 15, 21].

6 Conclusion

This work contributes to both theory and practice of HIS in many aspects. First, we presented a domain ontology model which is an abstract representation of the proposed metadata dictionary structure. Second, the proposed metadata dictionary scheme provides a mapping mechanism to associate user's requests posed at the conceptual level with the physical level, allowing direct access to stored information without loss of general query formulation. Third, the use of formal definition to represent domain ontology component design based on object-orientation serves as a systematic transformation from conceptual level to physical level. Such a conceptual-to-physical connection permits a straightforward means to plug-in/out autonomous information sources without affecting the overall system configuration. Fourth, choosing XML technology to express the contents of ontology components in the metadata dictionary renders maximal interoperability across heterogeneous systems which, in turn, offers system scalability by virtue of XML constructs. As such, metadata dictionary content management can be achieved by means of flexible XML data model.

References

[1] S. Adali and R. Emery, A uniform framework for integrating knowledge in heterogeneous knowledge systems, *Proceedings of the 11th International Conference on Data Engineering, Taipei, Taiwan, March 1995*, 513-521.

[2] N. Arch-int and P. Sophatsathit, A Reference Architecture for Integrating Heterogeneous Information Sources using XML and Agent Model, *Proceedings of the Joint Conference on Information Sciences, NC, USA, 8-14 March 2002*, 235-239.

[3] Y. Arens, C.Y. Chee, C.-N. Hsu and C. Knoblock, Retrieving and Integrating Data from Multiple Information Sources, *International Journal of Intelligent and Cooperative Information Systems* 2(2) (1993) 127-158.

[4] Y. Arens, C.-N. Hsu and C. Knoblock, Query processing in the SIMS information mediator. In: A. Tate (ed.), *Advanced Planning Technology, California, 1996*, 61-69.

[5] C. Batini and M. Lenzirini, A Methodology for Data Schema Integration in Entity-Relationship Model, *IEEE Transactions on Software Engineering* 10(6) (1984) 650-654.

[6] C. Batini, M. Lenzirini and S.B. Navathe, A Comparative Analysis of Methodologies for Database Schema Integration, *ACM Computing Surveys* 18(4) (1986) 323-364.

[7] A. Borgida, Description Logics in Data Management, *IEEE Transactions on Knowledge and Data Engineering* 7(5) (1995) 671-682.

[8] A. Borgida, R.J. Brachman, D.L. Mcguinness and L.A. Resnick, Classic: A structural data model for objects, *Proceedings of ACM SIGMOD International Conference on Management of Data, Portland, Oregon, USA, June 1989*, 58-67.

[9] D. Brickley and R. Guha, Resource Description Framework (RDF) Schema Specification, http://www.w3.org/TR/2000/CR-rdf-schema-20000327/, W3C Recommendation March 2000.

[10] S. Castano, V.D. Antonellis and S.D.C. Vimercati, Global Viewing of Heterogeneous Data Sources, *IEEE Transactions on Knowledge and Data Engineering* 13(2) (2001) 277-297.

[11] P.P. Chen, The Entity-Relationship Model – Toward a Unified View of Data, *ACM Transactions on Database Systems* 1(1) (1976) 9-36.

[12] S. Decker, M. Erdmann, D. Fensel and R. Studer, Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In: R. Meersman, et al. (eds.), *Database Semantics: Semantic Issues in Multimedia Systems, Boston, MA, USA, 1999*, 351-369.

[13] S. Decker, S. Melnik, F. Van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann and I. Horrocks, The Semantic Web: The Roles of XML and RDF, IEEE Internet Computing 4, September/October, 2000, 63-74.

[14] D. Fensel, Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce (Springer-Verlag, Berlin, 2001).

[15] D. Fensel, F.V. Harmelen, I. Horrocks, D.L. McGuinness, and P.F. Patel-Schneider, OIL: An ontology infrastructure for the Semantic Web, IEEE Intelligent Systems 16(2) (2001) 38-44.

[16] H. Garcia-molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos and J. Wisom, The TSIMMIS approach to mediation: data models and languages, *Journal of Intelligent Information Systems* 8(2) (1997) 117-132.

[17] T.R. Gruber, A translation approach to portable ontology specifications, *Knowledge Acquisition* 4(2) (1993) 199-220.

[18] T.R. Gruber, Toward Principles for the Design of Ontologies Used for Knowledge Sharing, *International Journal of Human-Computer Studies* 43(5/6) (1995) 907-928.

[19] J. Heflin, J. Hendler and S. Luke, SHOE: A Knowledge Representation Language for Internet Applications, *Technical Report CS-TR-4078 (UMIACS TR-99-71), Departments of Computer Science, University of Maryland at College Park*, 1999.

[20] J. Hendler, Agents and the Semantic Web, IEEE Intelligent Systems 16(2) (2001) 30-37.

[21] I. Horrocks, DAML+OIL: A description logic for the Semantic Web, IEEE Bulletin of the Technical Committee on Data Engineering 25(1) (2002) 4-9.

[22] D.M. Jones, T.J.M. Bench-capon and P.R.S. Visser, Methodologies for Ontology Development. In: J. Cuena, (ed.), *Proceedings of ITi and KNOWS Conference of the 15th IFIP World Computer Congress, Budapest, Hungary, August 1998*, 62-75.

[23] V. Kashyap and A. Sheth, Semantic Heterogeneity in Global Information Systems*:* The Role of Metadata, Context and Ontologies. In: M. Papazoglou and G. Schlageter (eds.), *Cooperative Information Systems*: *Current Trends and Directions* (Academic Press London, 1998).

[24] O. Lassila and R.R. Swick, Resource description framework (RDF) Model and syntax specification, http://www.w3.org/TR/REC-rdf-syntax/, W3C Recommendation February 1999.

[25] A.Y. Levy, A. Rajaraman and J.J. Ordille, Querying Heterogeneous Information Sources using Source Descriptions, *Proceedings of the 22nd International Conference on Very Large Databases, Bombay, India, September 1996*, 251-262.

[26] M.T. Özsu and P. Valduriez, Principles of Distributed Database System, Second Edition (Prentice-Hall, New Jersey, 1999).

[27] N.W. Paton, C.A. Goble and S. Bechhofer, Knowledge based information integration systems, *International Journal of Information and Software Technology* 42(5) (2000) 299-312.

[28] M.P. Reddy, B.E. Prasad, P.G. Reddy and A. Gupta, A Methodology for Integration of Heterogeneous Databases, *IEEE Transactions on Knowledge and Data Engineering* 6(6) (1994) 920-933.

[29] A.P. Sheth and J.A. Larson, Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases, *ACM Computing Surveys* 22(3) (1990) 183-236.

[30] A.P. Sheth, J.A. Larson, A. Cornellio and S. Navathe, A Tool for Integrating Conceptual Schemas and User Views, *Proceedings of the 4th International Conference on Data Engineering, Los Angeles, CA, February 1988*, 176-183.

[31] R. Studer, R. Benjamins and D. Fensel, Knowledge Engineering: Principles and Methods, *Data and Knowledge Engineering* 25(1-2) (1998) 161-197.

[32] M. Uschold, Building ontologies: Towards a unified methodology, *Technical Report AIAI-TR-197, University of Edinburgh*, 1996.

[33] M. Uschold and M. Gruninger, ONTOLOGIES: Principles, Methods and Applications. *The knowledge Engineering Review* 11(2) (1996) 93-155.

[34] P.E.V. Vet and N.J.I. Mars, Bottom-Up Construction of Ontologies, *IEEE Transactions on Knowledge and Data Engineering* 10(4) (1998) 513-526.

[35] W3C, World Wide Web Consortium, Extensible Markup Language (XML) 1.0 (Second Edition), *http://www.w3.org/TR/2000/REC-xml-20001006, W3C Recommendation 6-Oct-2000*.

[36] H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann and S. Hübner, Ontology-Based Integration of Information- A Survey of Existing Approaches. In: H. Stuckenschmidt, (ed.), *IJCAI-01 Workshop: Ontologies and Information Sharing, Seattle, WA, 2001*, 108-117.

[37] G. Wiederhold, Mediators in the Architecture of Future Information Systems, *IEEE computer* 25(3) (1992) 38-49.

Appendix: The Domain Ontology Components

The domain ontology components can be introduced as a general notation in formal semantics based on object-oriented and set theory. The domain ontology, denoted *D*, is defined as a quadruple

$$\boldsymbol{D} \;=\; <\boldsymbol{C}, \boldsymbol{R}, \boldsymbol{S}, \hat{\mathbf{C}}>, \text{ where}$$

- $C = \{vc_i \mid \forall\ i = 1 \ldots n\}$ is a finite set of all unique virtual concepts in *D*;
- *R* represents relationships between concepts or instances in *D* encompassing associative, IS-A, IS-PART-OF, and instantiated relationships;
- *S* represents physical source configurations; and
- $\hat{C}$ represents the construction rules established to govern operations over *C* and *R*, thus ensuring the correctness and consistency of the domain ontology.

These domain ontology components are defined as follows:

(1) Virtual concepts

A virtual concept $vc_k \in C$ is defined as a pair

$$\boldsymbol{vc_k} = <\boldsymbol{n(vc_k)}, \boldsymbol{P(vc_k)}>, \text{ where}$$

- $n(vc_k)$ is the virtual concept name which is unique within *D*; and
- $P(vc_k) = \{vp_{k1}, vp_{k2}, \ldots, vp_{km}\}$ is a finite set of unique virtual properties of the virtual concept vc_k.

These virtual properties can be classified into three groups, namely, *object identifiers*, *object identifier references*, and *ordinary properties*, depending on their property values.

A virtual property $vp_{kc} \in P(vc_k)$ is defined as a pair

$$\boldsymbol{vp_{kc}} = <\boldsymbol{n(vp_{kc})}, \boldsymbol{d}>, \text{ where}$$

- $n(vp_{kc})$ is the virtual property name which is unique within vc_k; and
- *d* is the domain variable defined below.

To solve data type and scaling conflicts, the virtual property $vp_{kc} \in OID(vc_k)$ or $vp_{kd} \in ORD(vc_k)$ is designed as a class consisting of two domain properties, namely, predefined type and scaling domains. Denote *PD* and *SD* as the sets of values of the predefined type domain and scaling domain properties, respectively, that is, *PD* = {"integer", "string", "float", "decimal", "char"} and *SD* = {"NULL", "kilogram", "pound", "US$", "AUS$"}. We define domain variable (*d*) of the virtual property vp_{kc} as follows:

$$d = \begin{cases} \text{NULL} & \text{if } vp_{kc} \in REF(vc_k) \\ <d_1, d_2> & \text{if } vp_{kc} \in OID(vc_k) \text{ or } vp_{kd} \in ORD(vc_k) \end{cases}$$

where d_1 is the predefined type domain property, denoted as a pair $<n(d_1), v(d_1)>$ in which $n(d_1)$ is the name of d_1 and $v(d_1) \in PD$ (e.g., $v(d_1)$ = "integer"), and d_2 is the scaling domain property, denoted as a pair $<n(d_2), v(d_2)>$ in which $n(d_2)$ is the name of d_2 and $v(d_2) \in SD$ (e.g., $v(d_2)$ = "kilogram").

To solve naming conflicts, the virtual property $vp_{kc} \in OID(vc_k)$ or $vp_{kd} \in ORD(vc_k)$ can also constitute physical instances, that is, instances representing the synonymous physical property names of physical concepts.

Let p_{kct} be a physical instance of vp_{kc} and be defined as a pair

$$\boldsymbol{p_{kct} = <n(p_{kct}), P(p_{kct})>}, \text{ where}$$

- $n(p_{kct})$ is the physical instance name (or physical property name) which is unique within vp_{kc}; and
- $P(p_{kct}) = \{I_1, I_2, \ldots, I_p\}$ is a finite set of unique physical information properties of p_{kct} that describes related physical information to p_{kct}. Examples of these properties are defined as follows:
  - I_1 is a physical data type property, denoted as a pair $<n(I_1), v(I_1)>$, where $n(I_1)$ is the name of I_1, and $v(I_1) \in PD$;
  - I_2 is a physical unit type property, denoted as a pair $<n(I_2), v(I_2)>$, where $n(I_2)$ is the name of I_2, and $v(I_2) \in SD$;
  - I_3 is a physical concept property, denoted as a pair $<n(I_3), v(I_3)>$, where $n(I_3)$ is the name of I_3, and $v(I_3)$ is an object identifier reference for the physical concept name of the physical source configuration; and
  - I_4 is a physical source property, denoted as a pair $<n(I_4), v(I_4)>$, where $n(I_4)$ is the name of I_4, and $v(I_4)$ is an object identifier reference for the physical source name of the physical source configuration.

(2) Relationships

The relationships in domain ontology consist of four types, namely, associative, IS-A, IS-PART-OF, and instantiated relationships.

- **Associative relationship**. An associative relationship enumerates the connectivity among instances of concepts. This relationship encompasses 1:1, 1:N, and N:M relationships.

- **IS-A relationship**. An IS-A relationship describes a specialization relationship among concepts that establishes a subsumption hierarchy, whereby a general concept (or superconcept) subsumes more specific concepts (or subconcepts). In other words, if the set of instances of vc_m is a subset of the set of instances vc_n, we say vc_n subsumes vc_m. A vc_n is called a superconcept of vc_m, and a vc_m is called a subconcept of vc_n.

In inheritance context, a subconcept vc_m can define its own properties and inherit the common properties from its superconcept vc_n. This implies that the relationships that associate the superconcept vc_n and other concepts can also propagate to its subconcept vc_m.

- **IS-PART-OF relationship**. An IS-PART-OF relationship is a relationship between an instance of an aggregate (or assembly) concept and its component instances. Each component instance belongs exclusively to one instance of an aggregate concept. In other words, if vc_m is a part of vc_n, vc_n is called an aggregate concept of vc_m, and vc_m is called a component concept of vc_n.

- **Instantiated relationship**. An instantiated relationship is a relationship between a virtual property and its physical instance. This relationship acts as a bridge to map ontology at the conceptual level to ontology at the physical level. The instantiated relationship can be used to verify synonymous (or equivalent) physical instances. We say that p_{111} and p_{112} are synonymous, and write $p_{111} \sim p_{112}$, if and only if both p_{111} and p_{112} are physical instances of the same class property vp_{11}. In other words, if we consider this relationship as a tree, we write $p_{111} \sim p_{112}$, if and only if both p_{111} and p_{112} are children of the same parent node vp_{11}.

(3) Physical source configurations

The physical source configurations at the physical level of the ontology describe the configurations of physical concepts and sources. Let $S = \{S_i \mid i = 1 \ldots n\}$ be a finite set of n physical information sources within D. A physical source, denoted $S_k \in S$, is defined as a pair

$$\boldsymbol{S_k = \langle n(S_k), P(S_k) \rangle}, \text{ where}$$

- $n(S_k)$ is the name of a physical source, which is unique within D; and
- $P(S_k) = \{pc_{k1}, pc_{k2}, \ldots, pc_{km}\}$ is a finite set of unique properties of S_k, that is, properties for storing the physical concept names (or entity names) in S_k.

A physical concept $pc_{kc} \in P(S_k)$ is defined as a pair

$$\boldsymbol{pc_{kc} = \langle n(pc_{kc}), P(pc_{kc}) \rangle}, \text{ where}$$

- $n(pc_{kc})$ is the name of pc_{kc}, which is unique within S_k; and
- $P(pc_{kc}) = \{c_j \mid j = 1 \ldots m\}$ is a finite set of unique physical configuration properties of pc_{kc}, that is, properties for storing the associated physical configurations of each pc_{kc} (e.g., physical data model, permission, owner). A property $c_k \in c_j$ is defined as a pair $\langle n(c_k), cnf \rangle$, where $n(c_k)$ is the name of c_k, and cnf is the value of the physical configuration property, which is an atomic string value.

(4) The construction rules

The construction rules, $\hat{C}$, are already outlined in Section 3.3.