Ontology-based Metadata Dictionary for Integrating Heterogeneous Information Sources on the WWW

Ngamnij Arch-int[1,*] **Peraphon Sophatsathit**[1] **Yuefeng Li**[2]

[1]Department of Mathematics, Faculty of Science, Chulalongkorn University, Bangkok, Thailand
[2]School of Software Engineering and Data Communications, Queensland University of Technology, Brisbane, Australia
Email: ngamnij@kku.ac.th, Peraphon.S@chula.ac.th, y2.li@qut.edu.au

Abstract

Semantic heterogeneity has always been one of the most important problems to overcome. A number of systems have been proposed to address this problem, ranging from mediator-based systems to description logic-based systems to content-descriptive metadata systems. In this paper, we propose an ontology-based metadata dictionary as a basis for solving semantic heterogeneity. First, the ontology-based metadata dictionary is modeled on the basis of a bottom-up design approach. Next, an XML-based data model is employed to manipulate and express the metadata dictionary contents to demonstrate how the proposed ontology-based metadata dictionary can be applied to a practical implementation. We also present the transformation of the ontology-based metadata dictionary into an XML-based metadata dictionary representation. Finally, the proposed approach is applied to a practical case study along with some related query processing to demonstrate the viability of model realization.

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

1 Introduction

Consolidation of data from Heterogeneous Information Sources (hereafter HIS) has always been one of the most challenging problems for distributed processing. One problem arising from integration of data heterogeneity is *semantic heterogeneity*. Such a problem occurs when there is a disagreement about the meaning, interpretation, or intended use of the same or related data (Sheth and Larson, 1990). Semantic heterogeneity can be classified into four types as follows:

- *Naming conflicts*, encompassing two different kinds of conflict, namely, synonym and homonym conflicts. Synonym conflicts are concerned with semantically equivalent concepts (i.e., entities) or properties (i.e., attributes) defined by different names. Homonym conflicts, on the other hand, are concerned with semantically unrelated concepts or properties defined by the same name;
- *Data Type conflicts*, concerning semantically equivalent properties that are defined with different data types;

[*] *Part of this work has been conducted while the author was a visiting academic at Queensland University of Technology, Brisbane, Australia*

- *Scaling conflicts*, concerning semantically equivalent properties that are defined with different scales or units of measure; and
- *Generalization conflicts*, concerning semantically related concepts that are defined in different systems where the concepts in one system subsume the concepts in another system.

Our approach is to model and design a metadata dictionary, which is an extension of a reference architecture proposed by Arch-int and Sophatsathit (2002). The metadata dictionary is based on domain specific metadata, using ontology (Fensel, 2001; Gruber, 1993; Uschold and Gruninger, 1996) as an assistant mechanism for accessing and integrating data represented by different data models into a homogeneous logical view.

The primary objectives of the proposed approach are:

- Solving semantic heterogeneity problems;
- Providing an abstract view for the application domain by hiding the contents, structures, and locations of real data;
- Providing a mapping mechanism to bridge the heterogeneous data to a homogeneous data group;
- Providing a flexible way to manipulate metadata dictionary contents when there is a need to control the consistency of the metadata dictionary;
- Providing information for users on the WWW to support system-wide interoperability; and
- Providing a global query model for users to access and integrate HIS.

In order to support system-wide interoperability suitable for a Web-based environment, we choose XML as the language for expressing the metadata dictionary contents, as well as providing flexibility and scalability in building and manipulating the ontology terminologies. The XML flexible data model also provides a means to consolidate data retrieved from various sources, while retaining consistent identification of the data semantics. The proposed metadata dictionary is also applied to a practical case study and query processing to demonstrate the viability of model realization.

The remainder of the paper is structured as follows. Section 2 presents the ontology-based metadata dictionary modeling technique and ontology extraction. Section 3 presents the structure of the XML-based metadata dictionary derived from the ontology-based metadata dictionary components. The XML-DTD metadata obtained in the process is also illustrated. Section 4 demonstrates how the proposed metadata dictionary solves semantic heterogeneity through a case study. Section 5 presents the query processing to access and integrate the HIS with the help of the metadata dictionary. Section 6 describes closely related works and how the proposed approach differs from other approaches. Section 7 concludes the paper and suggests further research extension.

2 The Ontology-based Metadata Dictionary

The ontology-based metadata dictionary is modeled on the basis of a bottom-up design approach (Castano, Antonellis and Vimercati, 2001; Özsu and Valdurie, 1999; Vet and Mars, 1998) to extract conceptual specification from the underlying physical information sources for explicit representations, thus forming a domain ontology model. The extraction of the ontology-based metadata dictionary by domain ontology modeling process is depicted in Figure 1 and summarized below.

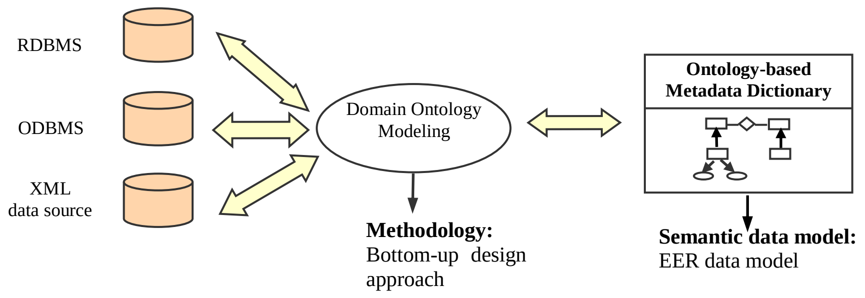

RDBMS
ODBMS
XML data source
Domain Ontology Modeling
Ontology-based Metadata Dictionary
Methodology: Bottom-up design approach
Semantic data model: EER data model

Figure 1. Extraction of the ontology-based metadata dictionary by domain ontology modeling.

(1) ***Schema Translation.*** This step involves translating or mapping the underlying physical information source schemas represented in various data models to intermediate schemas denoted by the canonical data models such as the E-R model (Chen, 1976). The physical information source schemas have been modeled to entities, attributes, relationships, and constraints. Other modeling considerations are type of relationship (e.g., one-to-one, one-to-many, and many-to-many) and attributes such as primary keys and foreign keys.

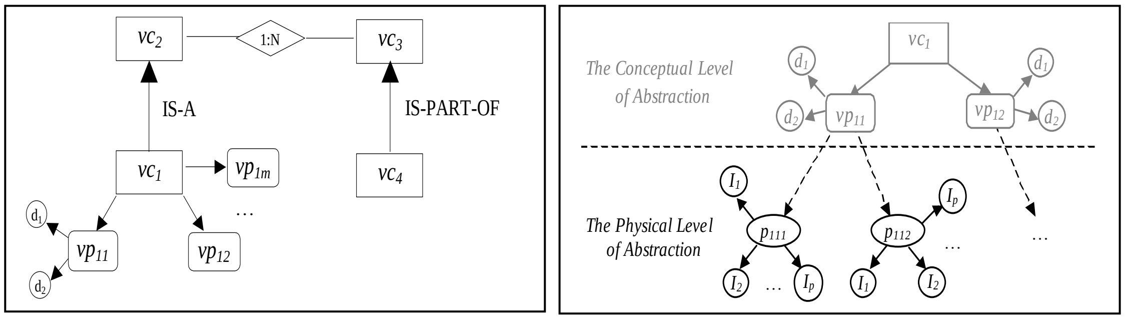
(2) ***Schema Restructuring.*** This step involves restructuring each intermediate schema to eliminate structural heterogeneity (Batini and Lenzirini 1984; Batini, Lenzirini and Navathe, 1986; Özsu and Valduriez, 1999). The results constitute the knowledge structure of the entire physical information source.

(3) ***Schema Integration.*** This step integrates all intermediate schemas into a global conceptual schema. The purpose of schema integration is to eliminate the generalization conflicts induced by the IS-A relationship between sub-type specific entities and the super-type general entity. The integration also applies to entities whose instances belong exclusively to an instance of another entity, that is, the component entities of an aggregate entity through the IS-PART-OF relationship.

(4) ***Ontology Extraction.*** The last step extracts ontology from the underlying global conceptual schema to obtain an explicit knowledge representation (Brachman and Levesque, 1985). The ontology is systematically extracted into two levels of abstraction, namely, the conceptual level of abstraction and the physical level of abstraction as follows:

- ***The conceptual level of abstraction.*** The global conceptual schema is restructured into a virtual schema, which is an initial ontology represented by the Extended

Entity-Relationship (EER) model. The virtual schema encompasses virtual concepts (or entities), virtual properties (or attributes), and relationships. 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 the domain ontology at this level is depicted in Figure 2 (a).



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

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

Figure 2. Two levels of domain ontology extracted from a global conceptual schema.

In this figure, boxes represent the 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. This level is designed to solve data type, scaling, and generalization conflicts. To eliminate data type and scaling conflicts, a virtual property is designed as a class that forms two domain properties, that is, the predefined type domains (e.g., integer, string, float, or char) and the scaling domains or units of measure (e.g., kilogram, pound, US$, or AUS$). These domain properties are used to represent different physical data types and unit types from HIS into a uniform format. Generalization conflicts are also eliminated through the IS-A relationships when connecting a specific concept to a general concept. The IS-PART-OF relationship is denoted by an arrow connecting a component concept to an aggregate concept.

- ***The physical level of abstraction.*** This level provides a mapping mechanism to associate the virtual concepts and properties of the virtual schema with the corresponding physical concepts and properties of the global conceptual schema. A partial internal ontology structure is illustrated in Figure 2 (b). This level is designed to solve naming conflicts by designating each virtual property to hold its instances, called physical instances and represented by ellipses, through the instantiated relationships. These physical instances store the synonymous physical property names of the physical concepts in the 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. The ontology in this level also holds physical source configurations which are augmented from the diagram. The physical source configurations describe the configurations of the physical concepts in each physical source and furnish necessary information to grant permission and knowledge for agents (Knoblock and Ambite, 1997; Li, Zhang and Swan, 2000; Papastavrou, Samaras and Pitoura, 2000) in accessing individual physical sources.

3 The XML-based Metadata Dictionary

The strength of XML in well-formedness, validity, and schema denotation makes it ideal for practical implementation of the ontology-based metadata dictionary. In so doing, all XML related documents can be validated by rules defined through XML-DTD data model.

In the following sections, the structural design of XML-DTD is set up from the domain ontology components to maintain their conceptual and physical correspondence and consistency, as depicted in Figure 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 3. The XML-DTD metadata dictionary structure.

3.1 The Conceptual Level of Design Abstraction

To capture the semantic elements of conceptual level modeling, the XML-DTD in this level must encompass all virtual concepts, their corresponding properties, and their relationships as follows:

(1) *The virtual concepts:* We denote all virtual concepts as the element `VConcepts`, consisting of one or more sub-elements `VConcept` of the domain ontology, whose names, $n(vc_i)$, $\forall i = 1..n$, are stored in the attribute `VCname` of `VConcept`. Each `VConcept` in turn consists of a sub-element `VProperties` and zero or one sub-element `VRelationships`.

(2) *The virtual properties:* The element `VProperties` of each `VConcept` contains one or more sub-elements `VPoid`, `VPref`, and `VPord`, which are the object identifier property, object reference property, and ordinary property, respectively. We denote $OID(vc_k)$ as a set of the object identifiers or keys of vc_k in which each instance of vc_k cannot have the same property value, $REF(vc_k)$ as a set of the object references or foreign keys of vc_k that establishes the relationships between instances of different concepts, and $ORD(vc_k)$ as a set of the ordinary properties whose values are atomic values (e.g., integer, string). The virtual property names, $n(vp_{kc}) \in OID(vc_k)$ and $n(vp_{kd}) \in ORD(vc_k)$, are designated to the attribute `VPname` of `VPoid` and `VPord`, respectively, whereas the virtual property names, $n(vp_{kt})$ of $vp_{kt} \in REF(vc_k)$, are designated to the data elements of `VPref`. These data elements of `VPref` are defined as *idref* to reference the virtual property names defined as *id* in `VPoid`. Each `VPoid` and `VPord` also consists of sub-elements `VDataType` and `VUnitType`, whose data elements are designated to store the virtual data type (d_1) and unit type (d_2), respectively. Note that a NULL value in a `VUnitType` element designates a property that is not of a unit measure.

(3) *The relationships:* Each `VConcept` can associate with zero or more concepts whose names, $n(vc_j)$ $(\forall j \subset \forall i=1..n)$, 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 IS-A, IS-PART-OF and associative relationships between `VConcept` and `AssocConcept` are designated to the attribute `VRelname` of `VRelationship`.

3.2 The Physical Level of Design Abstraction

This physical level is designed to incorporate the physical source configurations of the physical concepts and sources. Thus, the semantic elements of the physical level modeling are captured via the XML-DTD that encompasses the physical property names of physical concepts and other physical information pertaining to the virtual properties and concepts in the conceptual level.

(1) *Physical properties and other physical information.* Each `VPoid` and `VPord` property at the conceptual level contains one or more synonymous physical properties whose names, $n(p_{kct})$ of $vp_{kc} \in OID(vc_k)$ and $n(p_{kdt})$ of $vp_{kd} \in ORD(vc_k)$, are designated to the attribute

PPname of PProperty of VPoid and VPord, respectively. Other physical information related to the physical property names, such as the physical data type *(I_1)* and unit type *(I_2)*, are designated to the data elements PDataType and PUnitType, respectively. Similarly, the physical concept names *(I_3)* and source names *(I_4)* 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 that are defined as *id* in the physical source configurations.

(2) *The physical source configurations.* The element PhysicalSourceConfs consists of one or more sub-elements PSource denoting the physical information sources whose names are designated to the attribute PSname of PSource. Each PSource consists of one or more sub-elements PConcept, denoting the physical concepts whose names are designated to the attribute PCname of PConcept. The values of other physical configurations that associate with each physical concept (e.g., physical data model, permission, and owner) are designated to the data elements of PDataModel, Permission and Owner, respectively.

4 A Case Study

We will demonstrate how the proposed metadata dictionary solves the semantic heterogeneity problem through an example of semantic heterogeneity that occurs in most organizations. The modeling technique illustrated in section 2 is applied to the design of the metadata dictionary in this case study. The example also serves as a basis for demonstrating the query processing of HIS in the next section.

4.1 An Example of Semantic Heterogeneity

We will demonstrate denotation of concepts and their relationships in a company as shown in Figure 4. In this example, we consider four main concepts, namely, Employee, Manager, Engineer, and Accountant. All members of the categories Manager, Engineer, and Accountant are contained in the concept Employee. We say that the concept Employee is a superconcept and the concepts Manager, Engineer, and Accountant are subconcepts. We also illustrate the partial IS-A relationship between Manager and Engineer, where some managers (not all) are engineers and some engineers are managers. Meanwhile, the Manager and Engineer concepts are independent of the Accountant concept since neither managers nor engineers are accountants.

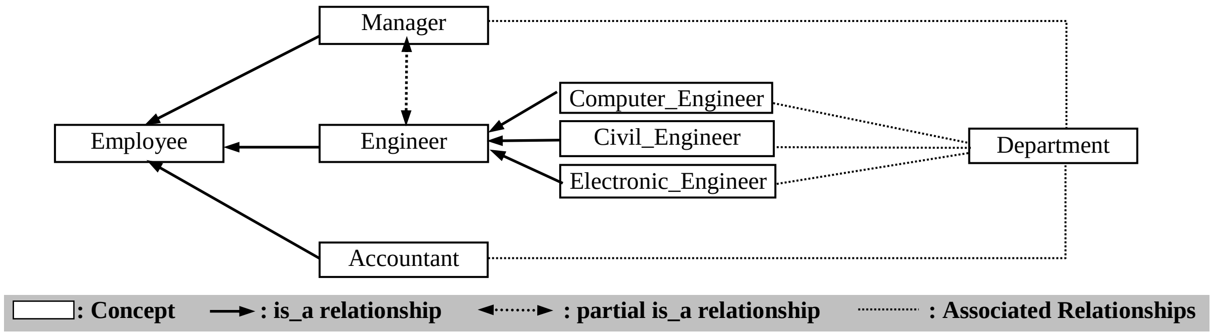


Figure 4. Types of relationship between concepts.

To illustrate how the proposed metadata dictionary solves semantic heterogeneity, we employ three different physical information sources as shown in Figures 5 (a), (b) and (c). The examples of each physical source not only illustrate the differences in data models and query languages, but also in semantic heterogeneity, which results in three types of conflicts. First, synonym conflicts occur as the attributes `Emp_id` in the relation `Employee_Member` of `Personnel`, `Eng_id` in the element `Engineer_Member` of `Engineering` source, and `Mng_id` in the class `Manager_Member` of `Management` source are semantically equivalent properties of the same fact. Second, data type and scaling conflicts are caused by the same attribute `Salary` of `Employee_Member` and `Engineering_Member` having different predefined types and units of measure. Finally, generalization conflicts induce from the derivation of the concept `Employee` subsuming the concepts `Manager` and `Engineer`. This example will serve as the basis for the ontology-based metadata dictionary design in the sections that follow.

Employee_Member

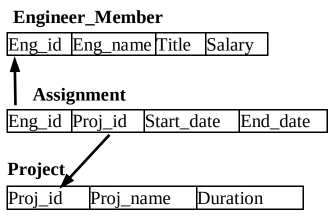
PhyProperty	**PhyDataType**	**PhyUnitType**
Emp_id	String	NULL
Emp_name	String	NULL
Salary	Float	US$
Dept_id	Integer	NULL

Department

PhyProperty	**PhyDataType**	**PhyUnitType**
Dept_id	Integer	NULL
Dept_name	String	NULL

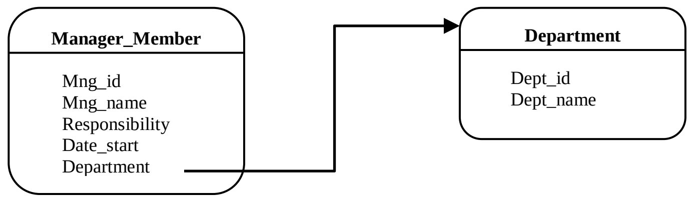
(a) The Personnel information source represented by a relational data model.

```
<?xml version="1.0"  standalone="yes"?>
<!DOCTYPE  Eng_Proj_Assign [
<!ELEMENT Eng_Proj_Assign                    (EngineerMember_List, Project_List)>
<!ELEMENT EngineerMember_List (Engineer_Member)+>
<!ELEMENT Engineer_Member (Eng_name, Title, Salary, Assignment)>
<!ATTLIST Engineer_Member   Eng_id   ID   #REQUIRED>
<!ELEMENT Eng_name   (#PCDATA)>
<!ELEMENT Title  (#PCDATA)>
<!ELEMENT Salary   (#PCDATA)>
<!ELEMENT Assignment (Proj)+>
<!ELEMENT Proj (Start_date, End_date)>
<!ATTLIST Proj   Proj_id    IDREF  #REQUIRED>
<!ELEMENT Start_date (#PCDATA)>
<!ELEMENT End_date (#PCDATA)>
<!ELEMENT Project_List (Project)+>
<!ELEMENT Project (Proj_name, Duration)>
<!ATTLIST Project Proj_id ID #REQUIRED>
<!ELEMENT Proj_name (#PCDATA)>
<!ELEMENT Duration (#PCDATA)>
]>
```



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

(b) The Engineering information source represented by XML-DTD.



(c) The Management information source represented by an object-oriented data model.

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

4.2 Domain Ontology Representation

4.2.1 Conceptual level representation

The conceptual level 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. The virtual property `ep_id` is an object identifier or key, `ep_name`, and `ep_salary` are ordinary properties, and `dept_id` is an object identifier reference or foreign key. The virtual concept `Employee` 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 a predefined type and scaling domain. For example, the domain properties of `ep_salary` are of the predefined type "Float" and scaling domain "US$." The generalization conflicts between the concepts `Employee`, `Engineer` and `Manager` are handled by associating `Engineer` and `Manager` with the IS-A relationships to `Employee`, since `Engineer` and `Manager` are subconcepts of `Employee`. Consequently, `Engineer` and `Manager` inherit `ep_id`, `ep_name`, `ep_salary`, and `dept_id` from `Employee` and associate with `Department` by an N:1 associative relationship.

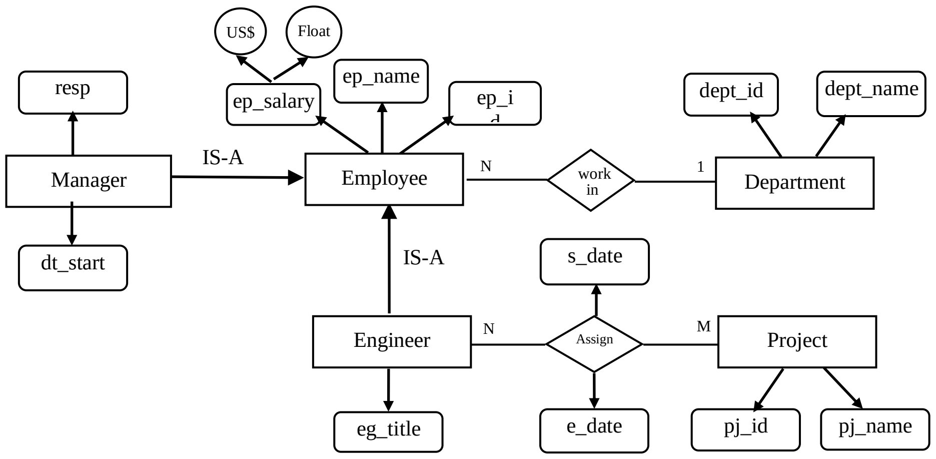


Figure 6. The logical ontology structure at the conceptual level of abstraction.

4.2.2 Physical level representation

Since synonym conflicts between the physical properties `Emp_id` of `Employee_Member`, `Eng_id` of `Engineer_Member`, and `Mng_id` of `Manager_Member` are common encounters in the HIS environment, the synonymous terms should be designed as the physical instances of the virtual property `ep_id` through the instantiated relationships. Each physical instance, `Emp_id` for example, is the physical property name, which can define its physical information properties for storing additional physical information associated with `Emp_id`. For example, the values of physical information properties named `PDataType`, `PUnitType`, `PCname`, and `PSname` of `Emp_id` are "integer", "NULL", "Employee_Member", and "Personnel", respectively. This means that `Emp_id` is a physical property name having the physical data and unit types "integer" and "NULL", and the physical concept name “Employee_Member”, which resides in source “Personnel” as shown in Figure 7.

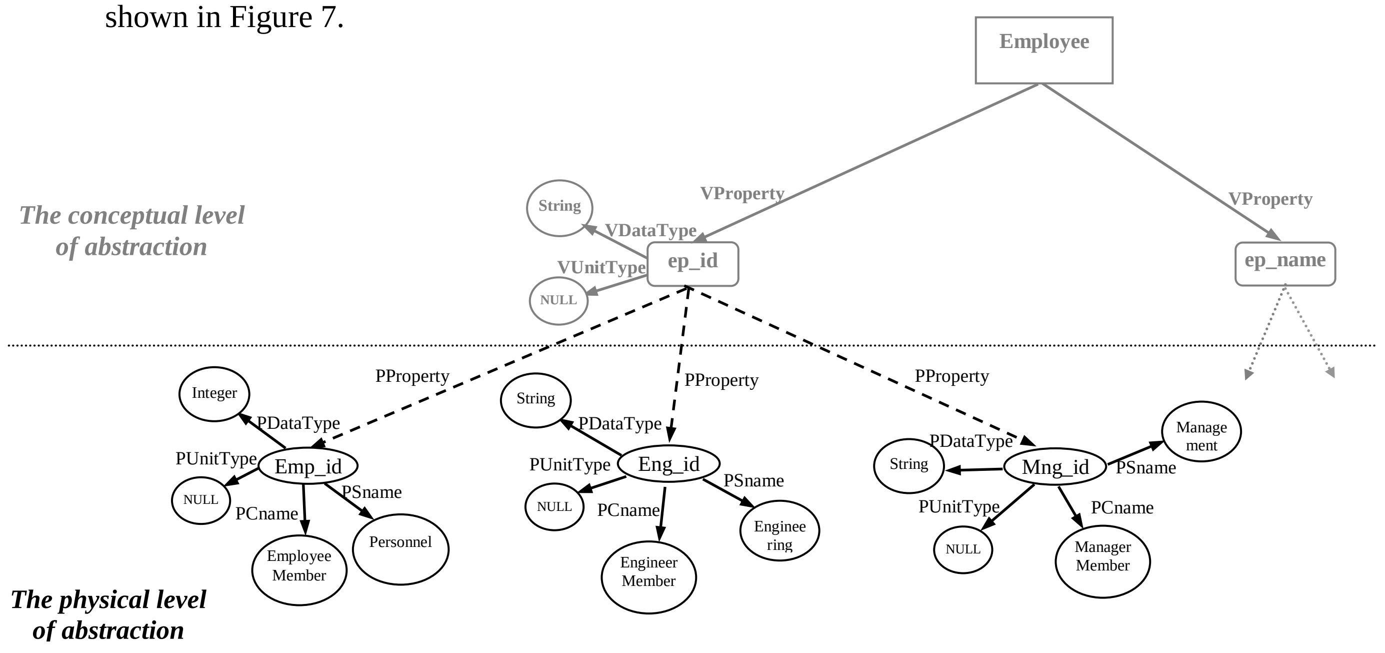
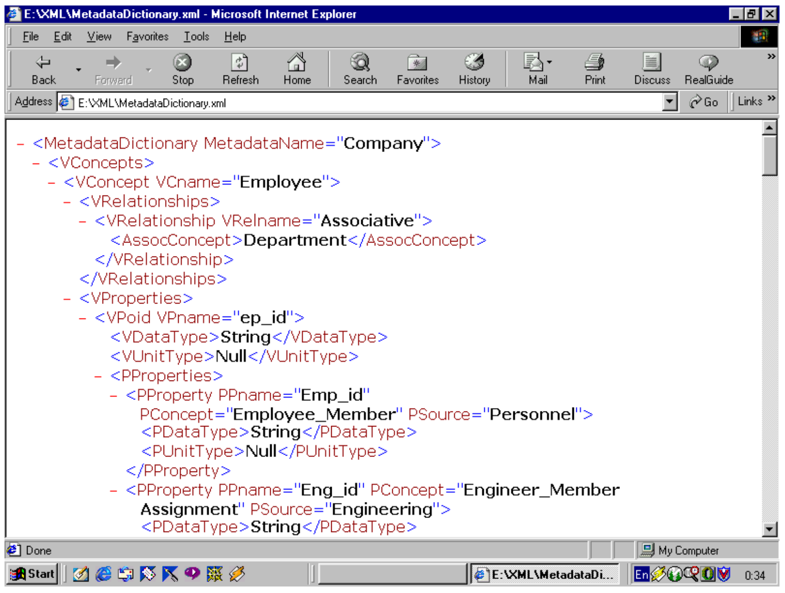


Figure 7. A portion of internal structure of the ontology at the physical level of abstraction.

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 XML-DTD (as shown in Figure 3) and XML documents. A partial XML document structure for storing well-formed and valid data is given in Figure 8.


E:\XML\MetadataDictionary.xml - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Discuss RealGuide
Address E:\XML\MetadataDictionary.xml
Go Links
Done
My Computer
Start
E:\XML\MetadataDi...
0:34

```
- <MetadataDictionary MetadataName="Company">
  - <VConcepts>
    - <VConcept VCname="Employee">
      - <VRelationships>
        - <VRelationship VRelname="Associative">
            <AssocConcept>Department</AssocConcept>
          </VRelationship>
        </VRelationships>
      - <VProperties>
        - <VPoid VPname="ep_id">
            <VDataType>String</VDataType>
            <VUnitType>Null</VUnitType>
          - <PProperties>
            - <PProperty PPname="Emp_id"
                PConcept="Employee_Member" PSource="Personnel">
                <PDataType>String</PDataType>
                <PUnitType>Null</PUnitType>
              </PProperty>
            - <PProperty PPname="Eng_id" PConcept="Engineer_Member
                Assignment" PSource="Engineering">
                <PDataType>String</PDataType>
```

Figure 8. A portion of the XML document complying with XML-DTD.

5 Query Processing for Heterogeneous Information Sources

The querying process for HIS (Arch-int, Li, Roe, Sophatsathit, 2003) aims to enable users to pose their queries over the virtual schema instead of the physical source schema to obtain relevant answers from the HIS. The querying process encompasses two main processes, namely, the accessing process of the HIS and the integrating process of the results from the HIS. The accessing process is responsible for generating a global transaction associated with the user's request, and decomposing the global transaction into sub-transactions for accessing the real data in the physical information sources. The global transaction decomposition maps the virtual properties and concepts of the global transaction to physical properties and concepts of the sub-transactions. In contrast, the integrating process focuses on consolidating the XML results obtained from the physical information sources into unified XML-based data corresponding to the user's request. Details on how each step is carried out are described below.

5.1 The Accessing Process of the Heterogeneous Information Sources

The accessing process of the HIS starts at the presentation layer of the reference architecture proposed by Arch-int and Sophatsathit (2002). Any virtual concept that is a subconcept inherits all virtual properties from its superconcepts. These virtual properties are thus presented to the users as the properties of the subconcept. For example, the user can view the virtual properties of `Engineer` originating from `Engineer` and `Employee`. The user can pose a query through a unified-query form encircling the virtual schema provided by the user interface agent (Arch-int and Sophatsathit, 2002) or can pose a query in standard SQL format. There are two steps to access the HIS, namely, global transaction creation and decomposition.

(1) Global Transaction Creation: A global transaction is a visual user requirement represented in standard SQL format that consists of virtual concepts and properties of the virtual schema as illustrated in Figure 9. Upon submission of a user query that may be in any arbitrary complex form, the request will be sent to the user interface agent to form the global transaction, which is a normalized query form, by means of the metadata dictionary. The query normalization eliminates type mismatch, semantic mismatch, and redundant predicates (Özsu and Valduriez, 1999) from the global transaction. A formal definition of the global transaction is given in the Definition 1.

```
SELECT Employee.ep_name, Employee.ep_salary,
Project.pj_name
FROM Employee, Engineer, Project, Assign
WHERE Employee.salary > 10000
AND Employee.ep_id  = Engineer.ep_id
AND Engineer.eg_title = "Chief Eng."
AND Engineer.ep_id = Assign.ep_id
AND Assign.pj_id = Project.pj_id
```

Figure 9. A global transaction generated from a user interface agent.

Definition 1: Let Q be a global transaction defined as a triple $<S, C, P>$, where $S = \{vc_i.vp_{ij} \mid \forall i=1\ldots n, \forall j=1\ldots m\}$ is a finite set of target virtual properties vp_{ij} (or attributes) of virtual concepts vc_i, that are in the SELECT clause, and the property value of vp_{kc} is defined over the domain D_c; $C = \{vc_i \mid \forall i = 1\ldots n\}$ is a finite set of target virtual concepts (or entities) in the FROM clause; and $P = Qp \cup Jp$ is a finite set of predicates in the WHERE clause consisting of two kinds of predicates, (*i*) the set $Qp = \{c_i \mid \forall i = 1\ldots n\}$ of qualifying predicates and (*ii*) the set $Jp = \{j_i \mid \forall i = 1\ldots m\}$ of join predicates, such that

- A qualifying predicate $c_k \in Qp$ is defined as $vc_k.vp_{kc}\ \Theta\ value$, where $\Theta \in \{=, <, >, \neq, \leq, \geq\}$ and $value \in D_c$ are defined in the qualified virtual property $vc_k.vp_{kc}$, and
- A join predicate $j_k \in Jp$ is defined as $vc_k.vp_{kc} = vc_m.vp_{mc}$, where $k \neq m$, and $vp_{kc} = vp_{mc}$.

An example of a qualifying predicate is `Employee.ep_id` = “11111,” and a join predicate might be `Employee.dept_id` = `Department.dept_id`.

(2) Global Transaction Decomposition: After global transaction creation, the global transaction is sent to the managing agent (Arch-int and Sophatsathit, 2002), where global transaction decomposition is initiated. This process transforms the global transaction into sub-transactions by substituting each virtual concept and property in the global transaction with the corresponding physical concept and property of the local physical sources obtained from the metadata dictionary as defined in Definition 2.

Definition 2: A global transaction Q would be transformed into sub-queries or sub-transactions $q_1,\ldots, q_n$ over the physical schema such that $q_1,\ldots, q_n$ have to provide potential answers relevant to the user query. The transformation process maps the virtual schema in Q into physical schemas assigned in $q_1,\ldots, q_n$.

For example, the virtual property `ep_id` in a global transaction is replaced by `Emp_id` of `Employee_Member` and `Eng_id` of `Engineer_Member` to form *sub-transaction1* and *sub-transaction2,* respectively. A sub-transaction will subsequently access data from a physical information source.

The decomposition process of the global transaction is described in two main steps as follows:

(1) *Mapping.* The virtual concepts and properties in the SELECT clause are mapped to the associated physical concepts and properties and also the physical sources in which each physical concept resides.

(2) *Sub-transaction creation.* Each sub-transaction is created from the following processes:

(2.1) *Grouping process:* The virtual concepts/properties and the corresponding physical concepts/properties with the same physical source are grouped together.

Let $S = \{PSname_i \mid i = 1\ldots n\}$ be a finite set of physical source names. A physical source name $PSname_k \in S$ is defined as a finite set of virtual concepts/properties and the corresponding physical concepts/properties such that $PSname_k = \{p_i \mid i = 1\ldots n\}$, where p_k is defined as a quadruple $<vc_k, vp_{kc}, PCname_k, PPname_k>$.

For example, the virtual properties/concepts in the global transaction of Figure 9 are mapped into physical information and grouped by *PSname* as follows:

S = {“Personnel”, “Engineering”}

Personnel = {<“Employee”, “ep_name”, “Employee_Member”, “Emp_name”>,
<“Employee”, “ep_salary”, “Employee_Member”, “Salary”>}

Engineering = {<“Employee”, “ep_name”, “Engineer_Member”, “Eng_name”>,
<“Employee”, “ep_salary”, “Engineer_Member”, “Salary”>,
<“Project”, “pj_name”, “Project”, “Proj_name”>}

(2.2) *Substitution process*: This process generates a sub-transaction for accessing each $PSname_k$ by substituting the virtual concepts/properties in each $PSname_k$ with the corresponding physical concepts/properties to form a sub-transaction, denoted by $PSname_k$: $<vc_k \rightarrow PCname_k, vp_{kc} \rightarrow PPname_k>$ as illustrated below. The physical properties constitute the requested information in the SELECT clause, and the physical concepts represent the target information sources to be accessed in the FROM clause.

Personnel: <“Employee” → “Employee_Member”, “ep_name” → “Emp_name”>
<“Employee” → “Employee_Member”, “ep_salary” → “Salary”>

Engineering: <“Employee” → “Engineer_Member”, “ep_name” → “Eng_name”>
<“Employee” → “Engineer_Member”, “ep_salary” → “Salary”>
<“Project” → “Project”, “pj_name” → “Proj_name”>

(2.3) *Generating the constraints:* The virtual concepts/properties in the WHERE clause of a global transaction are also mapped to the associated physical concepts, properties and sources. Two kinds of predicates in the WHERE clause are considered:

2.3.1 Qualifying predicates. For each group with the same physical source, the qualifying predicates of the global transaction are replaced with the physical properties and concepts to form the same constraints in a sub-transaction, that is, for each $PSname_k$: $(vc_k.vp_{kc}\ \Theta$ value$) \rightarrow (PCname_k.PPname_{kc}\ \Theta$ value$)$. For example, a qualifying predicate `Employee.ep_salary > 10000` is replaced with `Employee_Member.Salary > 10000` and `Engineer_Member.Salary > 10000` to form the qualifying predicate in the sub-transactions of `Personnel` and `Engineering`, respectively.

2.3.2 Join predicates. For each $PSname_k$, the join predicates of sub-transactions are considered as follows:

- If $PCname_k$ and $PCname_m$ correspond with vc_k, and vc_m, respectively, and reside in the same source, the join predicates of the global transaction are replaced with the same pairs of physical properties and concepts, that is, $(vc_k.vp_{kc} = vc_m.vp_{mc}) \rightarrow (PCname_k.PPname_{kc} = PCname_m.PPname_{mc})$, where $k \neq m$, and $PPname_{kc} = PPname_{mc}$. For example, `Engineer.ep_id = Assign.ep_id` is replaced with `Engineer_Member.Eng_id = Assignment.Eng_id` in a sub-transaction of `Engineering`, since `Engineer_Member` and `Assignment` are in the same physical source.

- If $PCname_k$ and $PCname_m$ correspond with vc_k, and vc_m, respectively, but reside in different sources, there are no join predicates to be generated in the sub-transactions, and each individual sub-transaction operates in its respective physical source. For example, the corresponding physical concept names of a join predicate `Employee.ep_id = Engineer.ep_id` in the global transaction are `Employee_Member` and `Engineer_Member`, which are in the `Personnel` and `Engineering` sources, respectively. Hence, no join predicates are generated in sub-transactions of `Personnel` and `Engineering`. This means that the returned results from these sources will be combined through the integration process that will be described in the next section.

All constraints obtained from the above procedures are combined to form the complete constraints of each sub-transaction as illustrated in Figure 10. Each sub-transaction, together with the physical source configurations that are necessary for accessing the HIS, is then packed and sent along with each search agent to the resource agent (Arch-int and Sophatsathit, 2002) at the destination physical source.

Personnel source

{E1| ∃E (Employee_Member(E) ∧ E.Salary > 10000 ∧ E1.Emp_name = E.Emp_name ∧ E1.Salary = E.Salary) }

Engineering source

{E2| ∃Eg, A, P (Engineer_Member(Eg) ∧ Assignment(A) ∧ Project(P) ∧ Eg.Salary > 10000∧ Eg.Title = “Chief Eng.” ∧ Eg.Eng_id = A.Eng_id ∧ A.Proj_id = P.Proj_id ∧ E2.Eng_name= Eg.Eng_name ∧ E2.Salary = Eg.Salary ∧ E2.Proj_name = P.Proj_name)}

Figure 10. Two sub-transactions decomposed from the global transaction in Figure 9.

5.2 The Integrating Process of the Heterogeneous Information Sources

Due to the different physical information sources that govern their own query languages in manipulating data represented in different data models, query language conflicts stemming from such differences must be eliminated. To eliminate these conflicts, each sub-transaction is transformed into the appropriate data manipulation language, regulated by each proprietary information source via the interface wrapper of the resource agent. The results obtained from the execution of each sub-transaction are transformed into a canonical data model represented in an XML-based format via the interface wrappers. These XML results are transmitted to the managing agent, where the integration process is carried out. The managing agent utilizes information obtained from the metadata dictionary to integrate the XML results into unified XML-based data consisting of an XML document and XML-DTD. The unified XML-based data is generated from the conceptual virtual schema and is forwarded to the user interface agent, where the presentation format is carried out at the presentation layer.

Definition 3: Given sub-transactions $q_1, \ldots, q_n$ generated from a global transaction Q, let $R(q_1), \ldots, R(q_n)$ be the results returned from each sub-transaction and represented in XML-based data. The unified XML-based data, denoted *UXML*, is the final result derived from integrating these XML results, such that

$$UXML = \underset{i=1\ldots n}{\Delta} R(q_i)$$

The operator Δ denotes the integration process that can be either a merge or join operation of the XML-based data and the mapping of the physical concepts/properties to virtual concepts/properties corresponding to the user's request. We represent each XML result as a labeled tree as defined in Definition 4.

Definition 4: A *labeled tree, T,* is defined as a pair $<t, n>$, where t is a finite sub-tree consisting of one or more nodes, n is a finite set of labeled nodes in t.

A tree, t, has a root node of the tree, denoted by *root(t),* with children $v_1, \ldots, v_k$, $k \geq 0$. A labeled node represents the begin-end tag in the XML data model. Attributes of XML are represented as tag elements of an XML document.

The integration process can be classified into two categories as follows:

5.2.1 Single Source Integration

If the XML results returned to the managing agent are obtained from a single source, the transformation process will map the corresponding physical properties and data values of the XML results to the virtual properties and data values in the form of unified XML-based data defined in Definition 5.

Definition 5: Let $R(q_a)$ be the returned results obtained from executing a sub-transaction q_a of a single source a, represented by a labeled tree, such that $R(q_a) = \{A_i \mid \forall i=1\ldots n\}$ is a finite set of records at the leaf nodes of the tree, where each record $A_c = \{<PPN_x, PPD_x> \mid \forall x = 1\ldots m\}$ is a finite set of a pair consisting of the physical property name PPN_c and its data values PPD_c.

The unified XML-based data *UXML* is generated from the mapping of $R(q_a)$ to *UXML* such that $R(q_a) \rightarrow UXML$ and $UXML = \{X_i \mid \forall i=1\ldots n\}$, where $X_c = \{<VP_j, VPD_j> \mid \forall j = 1\ldots m\}$ is a finite set of a pair consisting of the virtual property VP_c and its data value VPD_c. The VP_c and VPD_c are obtained from mapping $PPN_c \rightarrow VP_c$, and $PPD_c \rightarrow VPD_c$, respectively.

5.2.2 Multiple Source Integration

To provide flexible integration of the XML results obtained from multiple sources, a key or ID for each XML result is required for proper identification of the designated XML record. As we define it, each record $A_c \in R(q_a)$ contains key properties and non-key properties. Let Ka_c be a finite set of key properties of the record A_c, such that $Ka_c \subseteq A_c$, and let Xa_c be the finite set of non-key properties of a record A_c, such that $Xa_c \subseteq A_c$ and $Ka_c \cap Xa_c = \phi$.

Definition 6: Given the returned results $R(q_a)$ and $R(q_b)$ being sent to the managing agent, let $A_c \in R(q_a)$ be a record in $R(q_a)$ and $B_d \in R(q_b)$ be a record in $R(q_b)$. Each $PPN_k \in A_c$ and $PPN_m \in B_d$ will be searched for its correspondent virtual property in the metadata dictionary. If any PPN_k and PPN_m are children of the same parent virtual property and contain the same data values, these terms will be treated as synonymous terms and combined with the parent virtual property. In other words, we say that $PPN_k \sim PPN_m$ iff $ChildOf(PPN_k, VP_t) \wedge ChildOf(PPN_m, VP_t)$, and $PPD_k = PPD_m$ such that PPN_k and PPN_m, and their data values are integrated into a pair of $<VP_t, VPD_k>$ in the unified XML-based data.

For example, the `Emp_name` in `Personnel` and `Eng_name` in `Engineering` are synonymous since they are children of the same virtual property `ep_name` and both contain the same data value. These synonymous terms are combined into `ep_name` in the unified XML-based data. Examples of integrating the XML results that are sent from multiple sources `Personnel` and `Engineering` based on the global transaction in Figure 9 are illustrated in Figures 11 and 12. The XML results are represented as the labeled trees R_1 and R_2, as illustrated in Figures 11 (a) and (b), respectively.

From this example, both `Personnel` and `Engineering` return two records obtained from the respective labeled trees. Each tree R_1 and R_2 contains the sets Ka_i and Xa_i, $\forall i = 1 \ldots n$, such that each Ka_i is a finite set of a pair consisting of a virtual property and its data value, which are mapped from the physical key property and its data value. On the other hand, the set Xa_i is a finite set of a pair consisting of a virtual property and its data value, which are mapped from a physical non-key property and its data value. Hence, the first record of tree R_1 contains the finite set of key Ka_1 and non-key Xa_1, that is,

Ka_1 = { ("ep_id", "11111") }, and

Xa_1 = {("ep_name", "David"), ("ep_salary", "12000")}.

The second record contains the finite set of key Ka_2 and non-key Xa_2, that is,

Ka_2 = { ("ep_id", "22222") }, and

Xa_2 = {("ep_name", "John"), ("ep_salary", "15000")}.

For tree R_2, the first record contains the finite set of key Kb_1 and non-key Xb_1, that is,

Kb_1 = { ("ep_id", "22222") }, and

Xb_1 = {("ep_name", "John"), ("ep_salary", "15000"), ("pj_name", "EJP")}.

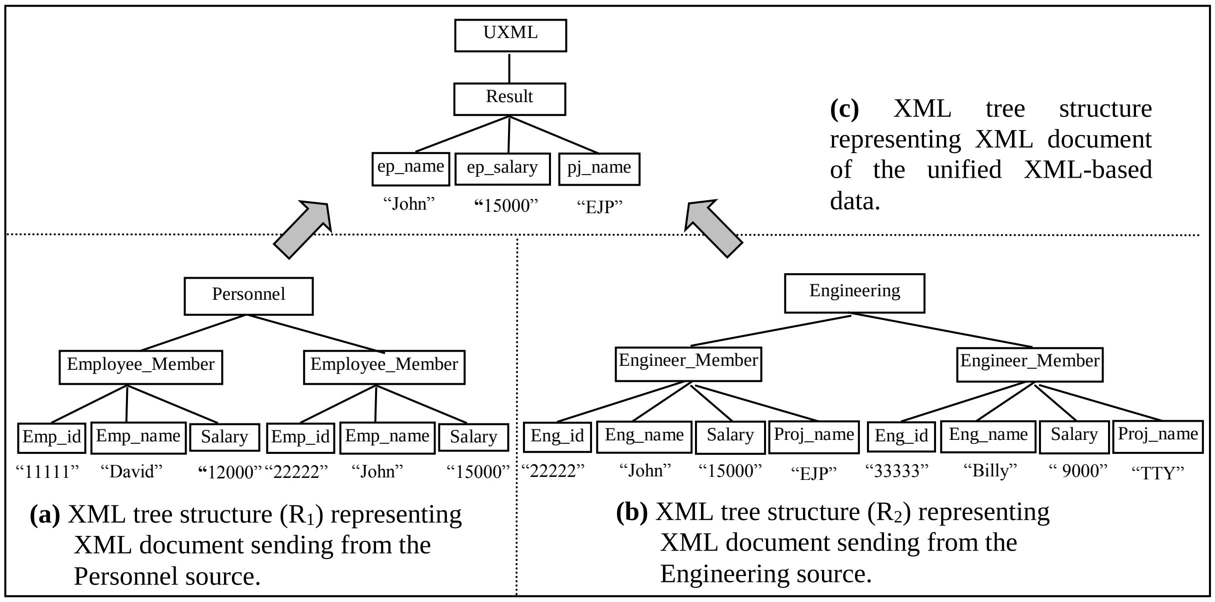
The second record contains the finite set of key Kb_2 and non-key Xb_2, that is,

Kb_2 = { ("ep_id", "33333") }, and

Xb_2 = {("ep_name", "Billy"), ("ep_salary", "9000"), ("pj_name", "TTY")}.

The integration process will join the records for each tree that has the same set of key Ka_c and Kb_k. In this example, the second record of the tree R_1 and the first record of the tree R_2 will be joined according to $Ka_2 = Kb_1$. However, since `ep_id` is not designated in the global transaction, that is, `ep_id` $\notin S$, only $Xa_2 \cup Xb_1$ are added to the set $UXML$. The unified XML-based data becomes:

$UXML$ = {{("ep_name", "John"), ("ep_salary", "15000"), ("pj_name", "EJP")}}

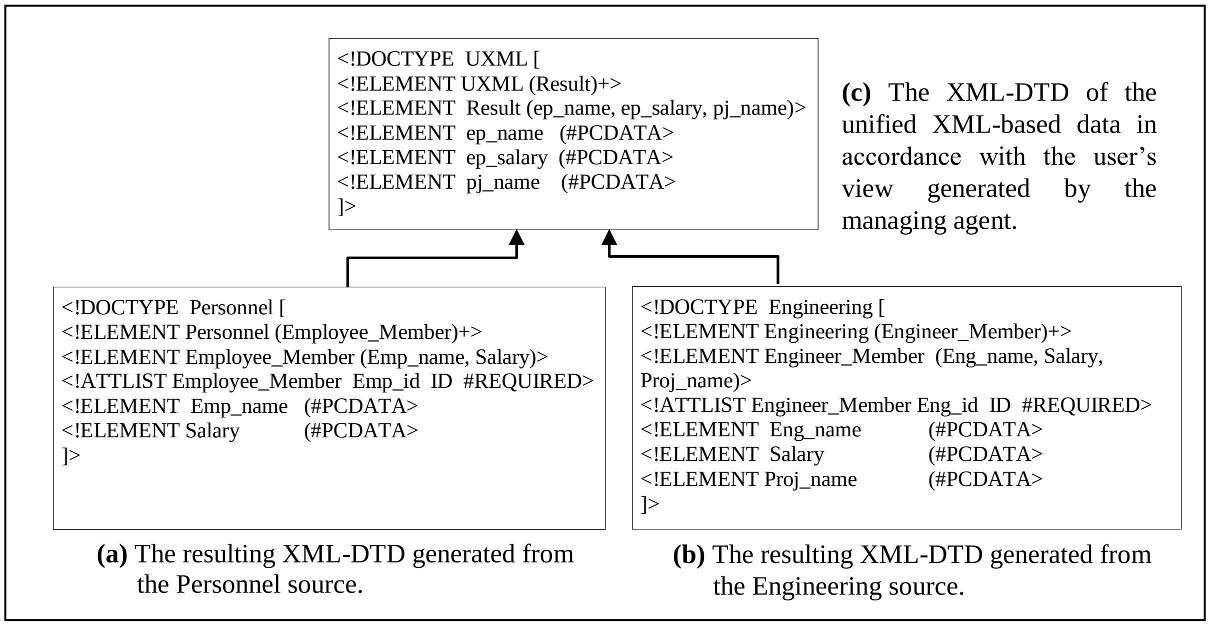


(c) XML tree structure representing XML document of the unified XML-based data.

(a) XML tree structure (R_1) representing XML document sending from the Personnel source.

(b) XML tree structure (R_2) representing XML document sending from the Engineering source.

Figure 11. Multiple source integration by joining the XML documents into the XML document of the unified XML-based data.



```
<!DOCTYPE  UXML [
<!ELEMENT UXML (Result)+>
<!ELEMENT  Result (ep_name, ep_salary, pj_name)>
<!ELEMENT  ep_name   (#PCDATA>
<!ELEMENT  ep_salary  (#PCDATA>
<!ELEMENT  pj_name    (#PCDATA>
]>
```

(c) The XML-DTD of the unified XML-based data in accordance with the user's view generated by the managing agent.

```
<!DOCTYPE  Personnel [
<!ELEMENT Personnel (Employee_Member)+>
<!ELEMENT Employee_Member (Emp_name, Salary)>
<!ATTLIST Employee_Member  Emp_id  ID  #REQUIRED>
<!ELEMENT  Emp_name    (#PCDATA>
<!ELEMENT Salary           (#PCDATA>
]>
```

(a) The resulting XML-DTD generated from the Personnel source.

```
<!DOCTYPE  Engineering [
<!ELEMENT Engineering (Engineer_Member)+>
<!ELEMENT Engineer_Member  (Eng_name, Salary,
Proj_name)>
<!ATTLIST Engineer_Member Eng_id  ID  #REQUIRED>
<!ELEMENT  Eng_name           (#PCDATA>
<!ELEMENT  Salary               (#PCDATA>
<!ELEMENT Proj_name           (#PCDATA>
]>
```

(b) The resulting XML-DTD generated from the Engineering source.

Figure 12. Multiple source integration by joining the XML-DTD of each source into the XML-DTD of the unified XML-based data.

6 Related Work

A number of systems have been proposed to cope with semantic heterogeneity problems. For example, mediator-based systems (Wiederhold, 1992) provide the inter-schema architecture for integrating access to data from different sources and converting data and queries into canonical formats via the mediator and wrapper components, respectively. Examples of such systems are TSIMMIS (Garcia-molina, Papakonstantinou, Quass, Rajaraman, Sagiv, Ullman, Vassalos, and Wisom, 1997) and HERMES (Adali and Emery, 1995). Description logic-based systems offer a different approach to elaborate source description by means of description logic (Borgida, 1995; Borgida, Brachman, Mcguinness and Resnick, 1989) for solving query processing over multiple sources. Unlike the mediator approach, the description logic approach abstracts the heterogeneous sources from users through a global view to facilitate query formulation. Examples of such systems are the Information Manifold (Levy, Rajaraman and Ordille, 1996) and the SIMS (Arens, Chee, Hsu and Knoblock, 1993; Arens, Hsu and Knoblock, 1996). Content-descriptive metadata systems (Kashyap and Sheth, 1998) 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 (Decker, Erdmann, Fensel and Studer, 1999) and the SHOE (Heflin, Hendler and Luke, 1999). All of these systems use the ontology approach to cope with heterogeneity problems. A survey and comparison of these systems can be found in Paton, Goble and Bechhofer (2000).

Our approach differs from other approaches from various standpoints:

- We proposed the ontology-based metadata dictionary as a key to solve semantic heterogeneity;
- The proposed approach maps a user’s query posed over virtual schema directly to physical schema without loss of information in the query; and
- To demonstrate how the proposed ontology-based metadata dictionary can be applied to a practical implementation, we present the transformation of the ontology-based metadata dictionary into XML-based metadata dictionary representation.

7 Conclusion and Future Work

This work contributes to both the theory and practice of HIS in many aspects. First, we presented a modeling process of the domain ontology, which is the basic building block of the metadata dictionary. The main purpose is to extract the ontology from the underlying physical sources and represent it explicitly. Second, the metadata dictionary provides a mapping mechanism to associate a user’s request posed at the conceptual level that is logically linked to the physical level, thus allowing direct access to stored information without loss of general query formulation. Third, choosing XML technology to express the contents of the metadata dictionary renders maximal interoperability across heterogeneous systems. Fourth, the metadata dictionary provides necessary information for accessing HIS, as well as the integration of the returned results into unified XML-based data. The unified XML-based data in turn provides the relevant answers in standard XML format corresponding to the user's request. Finally, the proposed approach enables the semantic heterogeneity problem to be solved at local and remote query processing.

A number of enhancements are necessary: First, the preliminary design of our approach supports only structured and semi-structured sources. It has to be extended to incorporate unstructured data sources. Second, query optimization is required for query processing efficiency. Finally, we can also enrich the proposed ontology-based metadata dictionary with advanced ontology language such as RDF/RDF Schema (Brickley and Guha, 2000; Lassila and Swick, 1999) to attain XML universal expressive power and syntactic interoperability toward machine-processable Semantic Web (Decker, Melnik, Harmelen, Fensel, Klein, Broekstra, Erdmann and Horrocks, 2000; Fensel, Harmelen, Horrocks, Mcguinness and Patel-schneider, 2001; Hendler, 2001).

References

ADALI, S. and EMERY, R. (1995): A uniform framework for integrating knowledge in heterogeneous knowledge systems. *Proc. Of the International Conference on Data Engineering*, Taipei, Taiwan, **11**:513-521, IEEE Computer Society Press.

ARCH-INT, N. and SOPHATSATHIT, P. (2002): A Reference Architecture for Integrating Heterogeneous Information Sources using XML and Agent Model. *Proc. of the Joint Conference on Information Sciences*, NC, USA, **6**:235-239.

ARCH-INT, N., LI, Y., ROE, P. and SOPHATSATHIT P. (2003): Query Processing the Heterogeneous Information Sources using Ontology-based Approach. *Proc. of the International Conference on Computers and Their Applications*, Honolulu, Hawaii, USA, **18**:438-441.

ARENS, Y., CHEE, C.Y., HSU, C.-N. and KNOBLOCK, C. (1993): Retrieving and Integrating Data from Multiple Information Sources. *International Journal of Intelligent and Cooperative Information Systems* **2**(2):127-158.

ARENS, Y., HSU, C.-N. and KNOBLOCK, C. (1996): Query processing in the SIMS information mediator. In *Advanced Planning Technology*. Austin Tate (ed), 61-69, California. AAAI Press.

BATINI, C. and LENZIRINI, M. (1984): A Methodology for Data Schema Integration in Entity-Relationship Model. *IEEE Trans. Software Eng* **10**(6):650-654.

BATINI, C., LENZIRINI, M. and NAVATHE, S.B. (1986): A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys* **18**(4):323-364.

BORGIDA, A. (1995): Description Logics in Data Management. *IEEE Trans. on Knowledge and Data Engineering* **7**(5):671-682.

BORGIDA, A., BRACHMAN, R.J., MCGUINNESS, D.L. and RESNICK, L.A. (1989): Classic: A structural data model for objects. *Proc. ACM SIGMOD International Conference on Management of Data*, Portland, Oregon, USA, 58-67, ACM Press.

BRACHMAN, R. and LEVESQUE, H., (ed) (1985): *Readings in knowledge representation*. Los Altos, California, Morgan Kaufmann.

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

CASTANO, S., ANTONELLIS, V.D. and VIMERCATI, S.D.C. (2001): Global Viewing of Heterogeneous Data Sources. *IEEE Trans. on Knowledge and Data Engineering* **13**(2):277-297.

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

DECKER, S., ERDMANN, M., FENSEL, D. and STUDER, R. (1999): Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In *Database Semantics: Semantic Issues in Multimedia Systems*. MEERSMAN, R. et al. (eds). Boston, MA, USA, **8**:351-369, Kluwer Academic Publisher.

DECKER, S., MELNIK, S., HARMELEN, F. V., FENSEL, D., KLEIN, M., BROEKSTRA, J., ERDMANN, M. and HORROCKS, I. (2000): The Semantic Web: The Roles of XML and RDF, **4**:63-74, IEEE Internet Computing.

FENSEL, D. (2001): Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. Springer-Verlag.

FENSEL, D., HARMELEN, F.V., HORROCKS, I., MCGUINNESS, D.L. and PATEL-SCHNEIDER, P.F. (2001): OIL: An ontology infrastructure for the Semantic Web. *IEEE Intelligent Systems* **16**(2):38-44.

GARCIA-MOLINA, H., PAPAKONSTANTINOU, Y., QUASS, D., RAJARAMAN, A., SAGIV, Y., ULLMAN, J., VASSALOS, V. and WISOM, J. (1997): The TSIMMIS approach to mediation: data models and languages. *Journal of Intelligent Information Systems* **8**(2):117-132.

GRUBER, T.R. (1993): A translation approach to portable ontology specifications. *Knowledge Acquisition* **4**(2):199-220.

HEFLIN, J., HENDLER, J. and LUKE, S. (1999): SHOE: A Knowledge Representation Language for Internet Applications. Technical Report CS-TR-4078 (UMIACS TR-99-71). Department of Computer Science, University of Maryland at College Park.

HENDLER, J. (2001): Agents and the Semantic Web. *IEEE Intelligent Systems* **16**(2):30-37.

KASHYAP, V. and SHETH, A. (1998): Semantic Heterogeneity in Global Information Systems*:* The Role of Metadata, Context and Ontologies. In *Cooperative Information Systems*: *Current Trends and Directions.* PAPAZOGLOU M. and SCHLAGETER G. (eds). London, United Kingdom, 139-178, Academic Press.

KNOBLOCK C. A. and AMBITE J. L. (1997): Agents for Information Gathering. In Software Agents. BRADSHAW J. M. (ed), 347-373, AAAI Press/The MIT Press.

LASSILA, O. and SWICK, R. R. (1999): Resource Description Framework (RDF) Model and Syntax Specification, W3C (World Wide Web Consortium). http://ww.w3.org/TR/1999/REC-rdf-syntax-19990222. W3C Recommendation 1999.

LEVY, A.Y., RAJARAMAN, A. and ORDILLE, J.J. (1996): Querying Heterogeneous Information Sources using Source Descriptions. *Proc. of the International Conference on Very Large Databases,* Bombay, India, **22**:251-262, Morgan Kaufmann.

LI Y., ZHANG C. and SWAN J. R. (2000): An information filtering model on the Web and Its Application in JobAgent. *Knowledge-Based Systems* **13**(5):285-296.

ÖZSU, M.T. and VALDURIEZ, P. (1999): Principles of Distributed Database System, Second Edition. Prentice-Hall.

PAPASTAVROU S., SAMARAS G. and PITOURA E. (2000): Mobile Agents for WWW Distributed Database Access. *IEEE Transactions on Knowledge and Data Engineering* **12**(5): 802-820.

PATON, N.W., GOBLE, C.A. and BECHHOFER, S. (2000): Knowledge based information integration systems. *International Journal of Information and Software Technology* **42**(5):299-312.

SHETH, A.P. and LARSON, J.A. (1990): Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys* **22**(3):183-236.

USCHOLD, M. and GRUNINGER, M. (1996): ONTOLOGIES: Principles, Methods and Applications. *The knowledge Engineering Review* **11**(2):93-155.

VET, P.E.V. and MARS, N.J.I. (1998): Bottom-Up Construction of Ontologies. *IEEE Trans. on Knowledge and Data Engineering* **10**(4):513-526.

WIEDERHOLD, G. (1992): Mediators in the Architecture of Future Information Systems. *IEEE computer* **25**(3):38-49.