Semantic Conflict Resolution Ontology (SCROL): An Ontology for Detecting and Resolving Data- and Schema-Level Semantic Conflicts

Sudha Ram

Department of Management Information System
Eller College of Business and Public Administration
The University of Arizona
Tucson, AZ 85721
E-mail: ram@bpa.arizona.edu
URL: http://vishnu.bpa.arizona.edu/ram

Jinsoo Park

Information and Decision Sciences Department
Carlson School of Management
University of Minnesota
Minneapolis, MN 55455
E-mail: park@umn.edu
URL: http://kimchi.csom.umn.edu

Index Terms: Heterogeneous Databases, Ontology, Semantic Conflict Resolution, Semantic Modeling

---

Working Paper
Last revised on December 2001

Semantic Conflict Resolution Ontology (SCROL): An Ontology for Detecting and Resolving Data- and Schema-Level Semantic Conflicts

Abstract

Establishing semantic interoperability among heterogeneous information sources has been a critical issue in the database community for the past two decades. Despite the critical importance, current approaches to semantic interoperability of heterogeneous databases have not been sufficiently effective. The federated schema approach has been criticized for its lack of semantic richness and flexibility. The domain ontology approach addresses the problem of semantic richness but lacks domain generality. We propose a formal structure of a common ontology called **S**emantic **C**onflict **R**esolution **O**nto**l**ogy (SCROL) that addresses the inherent difficulties in the conventional approaches: lack of semantic richness and limited domain generality. SCROL provides a systematic method for automatically detecting and resolving various semantic conflicts in heterogeneous databases. SCROL is formally defined to provide a dynamic mechanism of comparing and manipulating contextual knowledge of each information source, which is useful in semantic interoperability among heterogeneous databases. We show how SCROL is used for detecting semantic conflicts between semantically equivalent data elements. Using illustrative examples, we also demonstrate how the contextual knowledge captured in SCROL is used to detect and resolve semantic conflicts. In addition, we present evaluation results to show that SCROL can be successfully used to automate the process of identifying and resolving semantic conflicts. SCROL can also be used to provide interoperability for e-business systems, such as B2B systems.

1 Introduction

The concept of ontology, which originates from philosophy, has been widely employed by several research communities. In the artificial intelligence (AI) community, ontologies have been used to capture domain knowledge for knowledge-based systems. The knowledge is typically represented in the knowledge-based system's representation language using the vocabulary provided by an ontology. In the distributed artificial intelligence (DAI) community, which includes research on distributed problem solving (DPS) and multi-agent systems (MAS), ontologies have been accepted as an effective means to facilitate collaboration and communication among agents [39]. The need for ontologies has also been addressed in the information retrieval area to facilitate semantic information searching. Other areas, such as natural language processing (NLP), utilize ontologies to facilitate natural language generation and interpretation [21]. The database community is not an exception. In particular, research on distributed, heterogeneous databases has begun to exploit ontologies in order to support semantic interoperability.

In this paper, we present a formally defined ontology called SCROL (Semantic Conflict Resolution Ontology) that can be used to identify and resolve semantic conflicts among heterogeneous databases. Consider a county tax administrator Mary Beth, who is interested in understanding how property tax assessments are different across various counties within her jurisdiction. To answer this question, she may have to access many individual county databases simultaneously. However a major impediment is that, taxes are captured in different ways in each database. The Pima county database stores yearly tax amounts for each property, while Pinal county stores tax rates as a percentage of property value. Maricopa county, on the other hand, stores the monthly amount owed on each property. To be able to properly compare property taxes, Mary Beth has to understand these differences and also know how to resolve them before she can attempt to compare the taxes. Similarly, there are a host of other semantic differences which need

to be properly identified and resolved before databases can be used effectively. SCROL is intended to tackle this problem of recognizing and resolving semantic conflicts among multiple databases. Our major objective is to provide a robust mechanism for automating (to the extent possible) the semantic conflict identification and resolution process. SCROL has a simple structure yet results of our evaluation show that it is powerful enough to tackle a large variety of different semantic conflicts.

Semantic interoperability can be defined as the ability of participating system domains to understand the meaning and use of terminology from different domains and the axiomatic mapping ability between agreed concepts to make a semantically compatible information environment [31]. Establishing semantic interoperability among heterogeneous and disparate information sources has been a critical issue meriting active research within the database community for the past two decades [35]. Kashyap and Seth identified two essential issues for achieving semantic interoperability in a multidatabase environment [16]. The first issue concerns the identification of semantically related data in different database systems and the subsequent resolution of the schematic differences among the semantically related data. A key aspect of identifying semantically similar data in different databases involves making semantics explicit [5]. Semantic similarity depends on the context in which a data object is used, and the contextual representation of a data object concerns how the data object is used [15]. Therefore, context is a critical element for capturing and representing similarities of data objects. Several techniques allow us to determine semantically similar objects. These include semantic modeling approaches, formal logic-based approaches, classifications of terminology, formal languages, knowledge-based systems, and the use of a shared ontology [35].

The second important issue relates to the access and use of a large number of autonomous databases without prior knowledge of their information content. In general, users are required to be

familiar with the content and structure of the information sources in order to be able to obtain an answer for a particular query. To reduce the burden of acquiring such familiarity, one promising approach is based on graphical manipulation of database schema diagrams (or conceptual schema interfaces) [4]. This approach provides a uniform method for query translation and heterogeneity resolution in a multidatabase environment. Another approach is to explicitly capture the semantic content of the individual databases. It is, therefore, very important to understand the semantics of each schema component and to capture and reason by using the semantics [16].

Although semantic data models are commonly used in database design to capture the semantics of the database, the meta-information (i.e., tacit knowledge) captured during the design phase is not explicitly represented in the resulting database; hence, such information cannot be completely accessible to applications, queries, or users [41]. Therefore, a semantic data model that captures domain meta-information (i.e., entity classes, relationships, constraints, cardinalities, etc.) alone is not enough to support semantic interoperability among heterogeneous databases. While the different conceptual database schemas designed in a semantic data model provide the logical descriptions and relationships of the information within the databases, an ontology provides the concepts that represent the domain knowledge [6]. Basically, an ontology is a specification of the conceptualization of the target world [12] and hence provides a common vocabulary to describe the target world, which is one of its most important roles. In this respect, an ontology can be defined as a taxonomy of concepts, which includes relationships and constraints among concepts in order to eliminate unexpected or undesired interpretation. Each term (i.e., each concept) in an ontology has a unique meaning determined (and constrained) by richer relationships with other terms. The relationships among given terms in an ontology are extremely important because it is the relationships that express the knowledge specific to the application domain. In practice, it is often

very difficult to distinguish between "ontologies" and "knowledge" because there is no clear boundary between them as we observe in Cyc, the knowledge base described by Lenat [17].

The problem of semantic interoperability has generally been tackled by one of two approaches: the federated schema approach or the domain ontology approach. The federated schema approach attempts to construct a federated (or global) schema and establish mappings between the federated schema and the participating local schemas. However, the drawback of this approach is its lack of semantic richness and flexibility [32]. The other approach – the domain ontology approach – strives to solve the problem of lack of semantic richness by capturing the tacit knowledge within a certain domain in great detail in order to provide a rich conceptualization of data objects and their relationships. Even though such an approach may be theoretically valid, the application of the ontology approach in practice is practically infeasible due to the inherent complexities of the knowledge domain. Hence, the domain ontology approach is typically applied only to a restricted application domain, which limits its general applicability. Furthermore, in order to represent complex conceptualizations, the formalism used in representing the ontology also often becomes too complicated for wide application.

Our hybrid approach is based on the use of a common ontology, which specifies a vocabulary to describe and interpret shared information among its users. Our approach is similar to the federated schema approach in the sense that there is a high-level domain model playing a role of shared schema while ensuring the autonomy of the local schemas. However, a domain model is different from the conventional federated schema because domain knowledge captured in the domain model is generally represented in a logic language using the vocabulary provided by an ontology. An ontology-based domain model captures much richer semantics and covers a much broader range of knowledge within a target domain. Our approach is also similar to the conventional ontology approach in that an ontology-based domain model is constructed and used to

describe the domain knowledge; however, it is different from the conventional approach in that we provide a simple formalism to capture only the domain knowledge pertaining to potential semantic conflicts. One advantage of this simplified ontology is that it is not domain-specific. Our approach does not lose any semantic richness since it also provides a semantic model that captures the intensional description of the application domain. A promising approach to semantic interoperability is thus to adopt a common ontology as a basis for mutual understanding, in addition to the use of a semantic data model [14]. Hence, we argue that using both a common ontology and a semantic data model will provide a more complete understanding of the application domain.

Our paper is organized as follows. Section 2 explains the rationale behind and justification for using a common ontology approach. In Section 3, our common ontology, called **S**emantic **C**onflict **R**esolution **O**nto**l**ogy (SCROL), is formally defined. Section 4 presents the implementation of SCROL based on the classification scheme of semantic conflicts that provide the foundation for constructing SCROL. We also describe the ontology mapping and relationship knowledge, which are derived from the formal structure of SCROL. Section 5 presents examples demonstrating the use of SCROL. In addition, empirical results are discussed to evaluate the usefulness of SCROL. In Section 6, our approach is compared with previous approaches and our contributions are summarized. Finally, our future research directions are addressed in Section 7.

2 Need for a Common Ontology to Facilitate Semantic Interoperability

In this section, we first examine the concept of "ontology," a term commonly used in the fields of AI and knowledge management (KM). We then argue that the design of a common ontology is needed to facilitate interoperability in multiple heterogeneous systems. The (common) ontology is defined by various researchers as:

- The specification of a representational vocabulary for a shared domain of discourse, which may include definitions of classes, relations, functions, and other objects [12].

- A concept system in which all concepts are defined. Concepts are interpreted in a declarative way, as standing for the sets of their instances. This concept system is limitative in the sense that concepts can only be used if they are defined in the ontology. Definitions of concepts are formal where possible and informal otherwise [40].
- A model of some portion of the world, which is described by defining a set of representational terms [20].
- A means of achieving consistent communication between agents in multi-agent systems [28].
- A collection of concepts and interconnections to describe information units [14].

We use the term *common ontology* as a vocabulary of representational terms (concepts) with agreed-upon definitions in the form of human readable text and machine-enforceable, declarative constraints (agent readable format) on their well-formed use [10].

Semantic interoperability requires resolving various *context-dependent* incompatibilities, i.e., semantic conflicts [23]. The *context* refers to the knowledge that is required to reason about another system for the purpose of answering a specific query [24]. Therefore, it is important to provide contextual knowledge of domain applications in order to ensure semantic interoperability. Siegel et al. [37] argue that the most basic requirement of the use of context for heterogeneous databases is the existence of common metadata vocabularies, so that any system in the enterprise can use such a common vocabulary to develop rules, i.e., context knowledge describing data semantics. In this approach, terminology outside of this common vocabulary must be translated to the common vocabulary, otherwise the comparison of data semantics will not be possible [37]. Therefore, we believe that this approach is more practical than trying to agree upon broad-based standards for databases. Moreover, it is important to have automatic ways of comparing and manipulating the common vocabulary in order for a context knowledge representation to be useful for semantic interoperability among heterogeneous databases [16].

Consequently, in our framework, the common vocabulary that represents context knowledge is captured in the form of a common ontology, called **S**emantic **C**onflict **R**esolution **O**nto**L**ogy (SCROL), which provides a systematic way of automatically detecting and dynamically resolving various semantic conflicts found in heterogeneous databases. In addition, all schema components captured by a common semantic data model, called USM* [29], are mapped to SCROL. The use of SCROL, in which all data objects have been mapped, has several advantages:

- It facilitates sharing and reuse [40].
- Mappings can be associated with each database and application, and can be applied by mediators [13].
- An administrator constructing or maintaining mappings needs to consider only his or her own data objects, not those in any other database or application program [13].

Thus, a common ontology-based manipulation of complex and heterogeneous databases is one of the most desirable solutions for achieving semantic interoperability. There are, however, four important issues that need to be addressed. In working with a common ontology, Kahng and McLeod [14] discusses four of these:

- Contents: The contents of the common ontology are heavily affected by the semantic conflict types to be resolved.
- Construction and Maintenance: The initial construction of a common ontology prior to any information sharing is a challenging problem. Further, it is very important to allow the evolution of the common ontology.
- Mapping: Mapping from an information source to the common ontology is typically the most labor-intensive and time-consuming process and is mostly carried out by domain experts.
- Relevance: The similarities and differences between two data objects from different databases or the relevance of exported information to a given request needs to be determined at some point within the information sharing activities.

The first and second issues can be resolved by establishing agreements on the meaning of the terms used in a common ontology, i.e., ontological commitment [12]. In order to make the contents of the common ontology as general as possible, and thus usable in various environments, we use a comprehensive classification framework of semantic conflicts [30], which gives clear guidelines for capturing semantic conflicts in various heterogeneous databases using a well defined set of relationships between concepts to characterize application domains. The classification framework has been encoded into the common ontology, SCROL, and is describe in more detail in Section 4.1. After its initial construction, we allow evolution of the common ontology. Since semantic conflict taxonomy is possibly an approximation based on an agreement among participants, a system designed to solve the semantic conflicts is necessarily incremental and iterative [23]. Therefore, our approach (i.e., initial comprehensive construction and then the adoption of incremental and iterative evolution of the common ontology) is preferred because it allows the system to fine-tune and accommodate more contents as the system grows, while at the same time allowing the participating systems that may have evolving schemas to remain autonomous.

After the agreement has been reached, the next step (the third issue) is to establish mappings between information sources and the common ontology. Although the establishment of a comprehensive taxonomy and the mapping process may be difficult and time-consuming, being sharable and reusable by multiple heterogeneous environments probably justifies the extra effort in the design. The mapping is encoded in terms of the ontology mapping knowledge, described in Section 4.3. We approach the fourth issue by defining a formal structure of SCROL (Section 3) that allows the designer to describe how concepts are related to each other by specifying relationships between them in terms of the ontology relationship knowledge (Section 4.3). It provides initial linguistic links between concepts. In addition, the formal definitions of relationships in SCROL

clearly lead to a single semantic interpretation of a given concept via semantic transformation between different contexts (Section 4.2). Detailed descriptions are illustrated in later sections.

3 SCROL Constructs and Definitions

Gruber [11] maintains that formal ontologies are viewed as designed artifacts, formulated for specific purposes and evaluated against object design criteria. He suggests five design criteria for ontologies whose purpose is knowledge sharing and interoperability among applications based on a shared conceptualization: (1) clarity (i.e., objective definitions), (2) coherence (i.e., logically consistent definitions), (3) extensibility (i.e., ability to define new terms based on the existing vocabulary without revisions of the existing definitions), (4) minimal encoding bias (i.e., specification of the conceptualization at the knowledge level without depending on a particular symbol-level encoding), and (5) minimal ontological commitment (i.e., specifying the weakest theory and defining minimal terms that are essential to the communication of knowledge consistent with that theory). Ouksel, however, argues that it is not practically nor theoretically possible to develop and maintain an ontology that strictly adheres to these design criteria in an environment of autonomous, dynamic, and heterogeneous databases [23]. We believe that most currently existing ontologies have been developed mainly for the purpose of representing domain specific or commonsense knowledge, and they do not identify nor accurately classify semantic conflicts [23]. However, SCROL directly addresses this issue.

We formalized SCROL such that it can provide a systematic method for automatically detecting and assisting in resolving various semantic conflicts in heterogeneous databases. Unlike other traditional ontology frameworks designed to capture domain specific [19, 20, 40] or commonsense knowledge [3, 18, 38], *SCROL is developed to encode extensible knowledge on commonly found semantic conflicts that have been identified in our classification framework*. It then provides an automatic way of comparing and manipulating contextual knowledge of each

information source, which is used for semantic transformation across heterogeneous databases. Before we describe how SCROL can be structured to support semantic interoperability among heterogeneous databases, we first introduce and formally define the basic constructs of SCROL.

The structure of SCROL is a tree. A tree is a partially ordered set in which the predecessors (e.g., superconcepts) of each element are well-ordered; i.e., if $T = \{s \mid s < t\}$, where $s, t \in T$ and s is a predecessor of t, then each T is well-ordered [34]. Therefore, a tree can be used as a theoretical model of hierarchies or sets over which a "parenthood" relation (also called a "vertical" relation in our framework) is defined. We slightly extend and modify the basic definition of a tree to define so-called "horizontal" relations (i.e., "sibling" relations and "domain value mapping" relations, which are discussed later). Note that the formal model of SCROL does not seek to describe a single object but a whole class of objects and to define its formal structure, that is, to achieve logical organization.

Definition 1. A *Semantic Conflict Resolution Ontology* (SCROL) is a tuple $\Lambda = (OC, OI, RS, RM, u)$, where *OC, OI, RP, RS, RM,* and *u* are as defined below. Its structure is graphically illustrated in Fig. 1, and the graphical notation of each SCROL construct is illustrated in Fig. 2.

Definitions 2. *OC* is a distinct set of *concepts*. The oval shapes depicted in Fig. 1 represent concepts. Each element of *OC* is called a *node* of Λ. Concepts are represented as *terms*. A concept is related to instances in that a concept is a generalized abstract term that may have several concrete instances. For example, the term Temperature is a concept and Fahrenheit and Celsius are instances of Temperature. Fahrenheit and Celsius are two different specific expressions of Temperature. A concept may have zero or more *children*, and each child may be another concept or an instance. A concept may have exactly one parent concept. A concept that does not have any child concept is a *leaf concept*, i.e., a leaf concept may have one or more instances as children, but cannot have any concept as its child. Concepts have *properties*. Such properties are defined as follows:

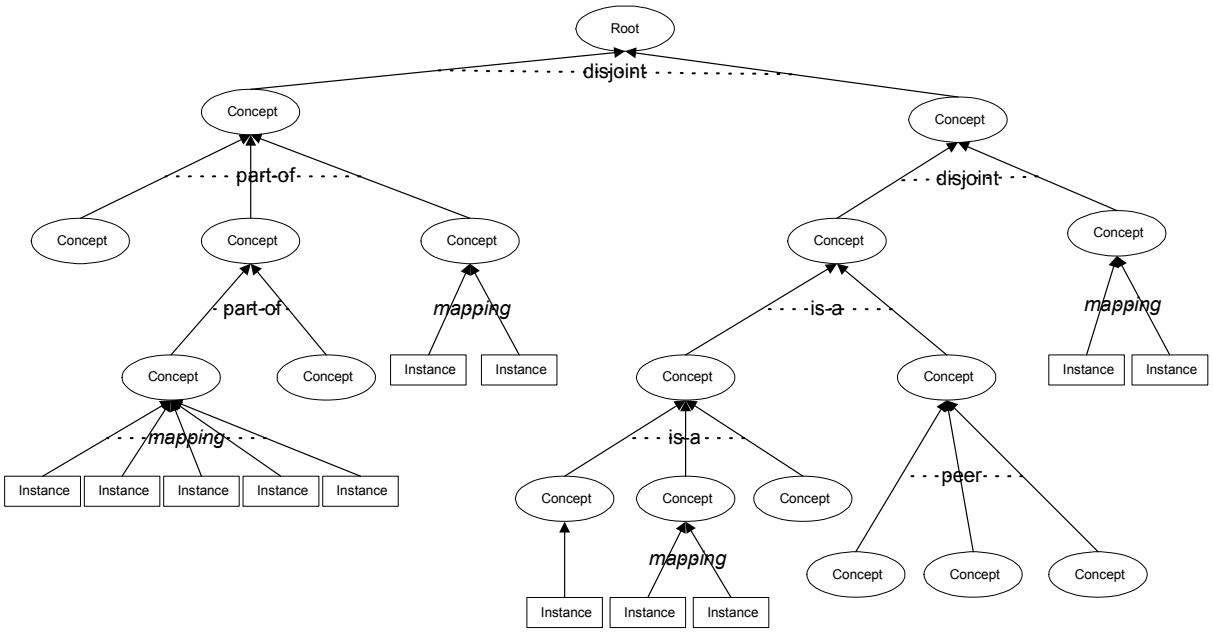

Fig. 1. Structure of SCROL

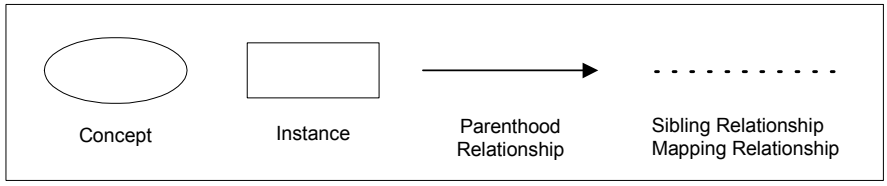

Fig 2. Graphical Notation of SCROL Constructs

- *Name* is a term that represents a concept.
- *Definition* is an agreed upon description of a concept written in plain, human-readable text.
- *Subconcept* is a property that is present in all concepts. This property contains a list of subconcepts (children) belonging to the concept. All leaf concepts have null values in this property because they do not have any subconcepts (but may have instances as stated above).
- *Subconcept-of* is a mandatory property of all concepts except the root. It contains its parent concept (called "superconcept"). A concept is allowed to have at most one parent concept.
- *Instance* is a property for only leaf concepts that have instances as their children. This property contains a list of instances belonging to the concept.
- *Referenced-by* is a property that is present only in leaf concepts. It is used to store mapping information about the underlying schema components (i.e., entity classes, relationships, and attributes) that are mapped to the concepts`. This property is expressed as an ordered tuple

$< s, o, t, l >$ where s is a referring schema, o is a referring object id (*oid*), t is an object type (i.e., *entity class*, *relationship*, or *attribute*), and l is a level of reference (i.e., *by-type* or *by-value*). The *by-type* means the object is referenced by its type and domain, while *by-value* refers to the data value itself. The default value of l is *by-type*.

Definitions 3. *OI* in Λ is a distinct set of *instances*. The rectangles depicted in Fig. 1 represent instances. Each element of *OI* is also called a *node* of Λ. Instances are also represented as *terms*. Every instance has exactly one concept as its parent. Instances do not have children. Instances have *properties*. Such properties are defined as follows:

- *Name* is a term that represents an instance.
- *Definition* is an agreed upon description of an instance written in plain, human-readable text.
- *Instance-of* is a mandatory property of all instances. It contains its parent concept. An instance has exactly one parent concept.
- *Referenced-by* is a property that is present in every instance. It is used to store mapping information about the underlying schema components (i.e., entity classes, relationships, and attributes) that are mapped to the concepts or instances. This property is expressed as an ordered tuple $< s, o, t, l >$ where s is a referring schema, o is a referring object id (*oid*), t is an object type (i.e., *entity class*, *relationship*, or *attribute*), and l is a level of reference which is only *by-type*.

Definition 4. *RS* refers to a *sibling relationship* and is a relation on *OC*. *RS* can occur only between two concepts, but not between two instances. *RS* consists of a *disjoint* relationship, a *peer* relationship, a *part-of* relationship, and an *is-a* relationship and has the following form: $<x, y, F>$, where $x, y \in OC$ and $F = \{disjoint, peer, is\text{-}a, part\text{-}of\}$. Note that *disjoint* and *peer* are symmetric but *is-a* and *part-of* are asymmetric. They are all transitive. The dotted vertical lines between concepts depicted in Fig. 1 represent sibling relationships with proper labels indicating either *disjoint*, *peer*, *part-of*, or *is-a* relationships.

- The *disjoint* relationship between two concepts indicates that they are not semantically

equivalent. For example, the concepts Distance and Temperature have a disjoint relationship because they are not semantically equivalent.

- The *peer* relationship is used when two concepts are semantically equivalent, that is, two concepts represent the same real world object, thus it is possible for the given two concepts to define one-to-one mapping between all the instances of these two concepts. Therefore, instances belonging to the two concepts can be transformed into each other through semantic transformation rules. In this case, the domain value mapping relationships (defined in the next definition) between all instances of such concepts are always one-to-one. For instance, as illustrated in Fig. 3, the subconcepts of a concept Localized Time have peer relationships. They are peers because in every instance they can have one-to-one semantic mappings; thus, they can be transformed into each other in a given context. For instance, the transaction time of stock trades recorded in Seoul can be converted to the local time in Bombay.

- The *part-of* relationship is similar to an "aggregation" in semantic data models and object-oriented data models. For example, the concept City is a part-of Urban Area which is a part-of County. The concept County is again a part-of a concept State/Providence and so forth.

- The *is-a* relationship is the same as "generalization/specialization" in semantic data models and object-oriented data models. For instance, the concept Water can have several specialized concepts, such as Ground Water and Surface Water.

Note that we don't have "overlapping" relationships because overlapping relationships can be

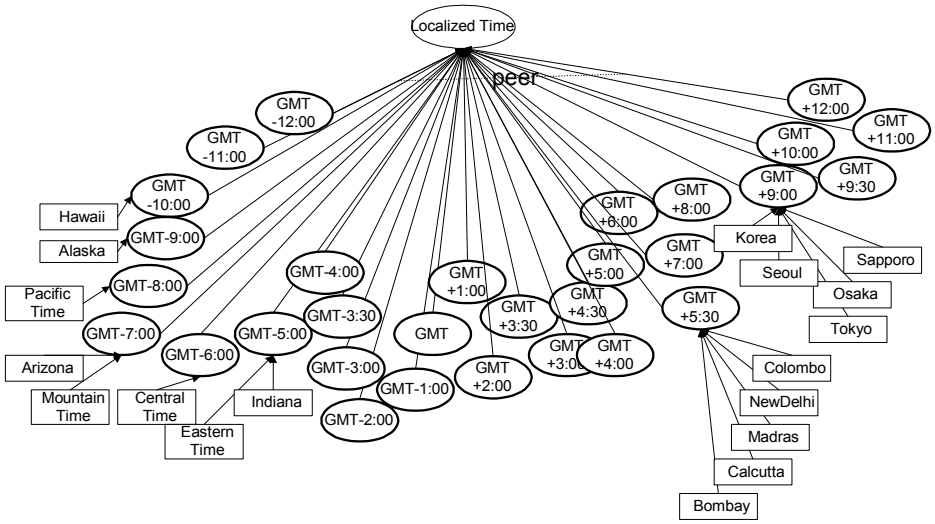

Fig. 3. *Peer* Relationship Example

integrated and depicted through generalization, i.e., *is-a* relationship. The *disjoint* and *peer* relationships are used by mediators to determine whether semantic conflicts exist (particularly data-level conflicts) and, if semantic conflicts exist, whether they are resolvable. The purpose of using *part-of* and *is-a* relationships is to allow mediators to detect schema-level conflicts between schemas.

Definition 5. *RM* in Λ is a relation on *OI* and called a *domain value mapping relationship* (or, briefly, a *mapping relationship*). The dotted vertical lines across parenthood relationships between instances depicted in Fig. 1 represent mapping relationships. By definition, *RM* can occur only between instances, but not between concepts. Another property of *RM* is that all instances belonging to a concept may be regarded as synonyms of their parent concept. This is the case where instances have different names for the same concept. Similarly, if two instances have the same name but belong to different concepts, they are homonyms. Note that all mapping relationships described below are derived from functional mappings in set theory. The mapping relationships are used by mediators to determine whether the actual data values that are mapped to instances can be transformed from one value to another and vice versa. *RM* consists of *one-one*, *one-many*, *many-many*, or *none*. In addition, *total* and *partial* mappings are used in combination of one, one-many, and many-many relationships to indicate whether every value in one instance has the corresponding value in other instances. The corresponding notation and description of each mapping relationship is presented in Table 1.

Definition 6. *u* is the *root* of Λ. The root *u* has no parent: there is no $x \in OC$ such that $RP(x, u)$. By definition, there is exactly one *u* in Λ.

Table 1
Notation of Mapping Relationships

Mapping Relationship	Notation
total one-one	$x \longleftrightarrow y$
total one-many	$x \longleftrightarrow\!\!\!\!\rightarrow y$
total many-one	$x \longleftarrow\!\!\!\!\rightarrow y$
total many-many	$x \longleftarrow\!\!\!\!\rightarrow y$
Partial one-one	$x \dashleftarrow\!\!\dashrightarrow y$
Partial one-many	$x \dashleftarrow\!\!\dashrightarrow y$
Partial many-one	$x \dashleftarrow\!\!\dashrightarrow y$
Partial many-many	$x \dashleftarrow\!\!\dashrightarrow y$
None	$x \not\longleftrightarrow y$

4 Implementation of SCROL

We have implemented software environment called CREAM (**C**onflict **R**esolution **E**nvironment for **A**utonomous **M**ediation). CREAM can be used to model and access heterogeneous databases via a Web-based graphical user interface [26]. CREAM provides various collaborative components, such as Schema Designer, Schema Mapper, Ontology Designer, Ontology Mapper, and Semantic Mediators, which enable the automation of conflict detection and resolution through their interaction with SCROL. Details about the integrated environment, CREAM, and its components can be found in [26]. SCROL constitutes a central part of CREAM. It has been implemented in Java using Oracle 8i as a repository. In the following subsections, we shall begin with a summary of the entire process of instantiating SCROL using our CREAM system.

4.1 Step 1: Defining SCROL using Ontology Designer

The Ontology Designer is a major component of CREAM, which helps database administrators define various concepts, instances, and their relationships by describing and classifying different types of semantic conflicts in SCROL. Such concepts, instances, and their relationships are stored in a set of tables owned by SCROL. A built-in integrity checker also prevents errors during the ontology design. Remember that SCROL is domain independent. Hence, the definition of SCROL is a *one time effort* and the resulting ontology can essentially be used repeatedly in many application

domains. SCROL needs to be extended only if the database administrator discovers that there is a brand new type of a semantic conflict that was previously undefined in SCROL.

In order to make the *contents* of SCROL as general as possible and to enable them to be widely reused by a broad range of different application domains, the concepts represented in SCROL are neither data or application-driven nor are they domain specific. Since the main objective of our ontology is to facilitate detection and identification of various semantic and schematic conflicts, our ontology, unlike the Cyc project [18], is not intended to accumulate a massive knowledge base of human consensus knowledge. This goal enables us to develop a very simple yet flexible ontology. In fact, most currently existing ontologies cannot identify and accurately classify semantic conflicts [23]. Since the discovery and reconciliation of semantic conflicts necessitate the ability to classify these conflicts, we construct SCROL based on a comprehensive classification framework of semantic conflicts [30]. This classification framework is based on extensive field study and investigation of various real geographic datasets (i.e., US Geological Survey data, vegetation data, land use data, etc.) containing several millions of records. We have rigorously defined additional types of semantic conflicts that are regularly encountered in geographic databases. We also tested each type of semantic conflict using these datasets to examine the extent to which CREAM automates the semantic conflict resolution process (see Section 5.2). The classification framework characterizes semantic conflicts at two different levels, i.e., data and schema level conflicts, each having six types of semantic conflicts. In most instances, data-level conflicts are differences in data domains caused by the multiple representations and interpretations of similar data. Schema-level conflicts are, on the other hand, characterized by differences in logical structures and/or inconsistencies in metadata (i.e., schemas) of the same application domain. A more detailed description of the classification framework is found in [30]. Using the Ontology Designer, we defined SCROL to capture all the types of conflicts represented in Tables 2 and 3.

Table 2
Data Level Conflicts

Conflict	Description
Data value conflict	Different interpretations of the "meaning" of data instance values
Data representation conflict	Similar objects are described by different data types or data format representations
Data unit conflict	Use of different measurement units
Data precision conflict	Implementation of different scales, different domain precision, or different data granularities and resolutions
Known data value reliability conflicts	Data present in different databases may be subject to data reliability (i.e.,measurement of error, measuring instruments, precision of measurements, topological properties, and treatment of time dimension)
Spatial domain conflict	Specifications of geographic regions or objects are "differently" but "legally" defined by different people

Table 3
Schema Level Conflicts

Conflict	Description
Naming conflict	Labels of schema elements (i.e., entity classes, relationships, and attributes) are somewhat arbitrarily assigned by different database designers (homonyms and synonyms)
Entity identifier conflicts	Assignment of different identifiers (primary keys) to the same concept in different databases
Schema isomorphism conflicts	Same concept (entity class) is described by a dissimilar set of attributes (i.e., the same concept is represented by a number of different attributes) or is not set operation compatible
Generalization conflicts	Different design choices for modeling related entity classes
Aggregation conflicts	When an aggregation is used in one database to identify a set of entities in another database
Schematic discrepancies	When the logical structure of a set of attributes and their values belonging to an entity class in one database are organized to form a different structure in another database

The various data level and schema level conflicts are encoded as the first level children concepts within SCROL's tree. For example, Fig. 4 shows the various data level conflicts that are encoded within SCROL. Due to space limitations, only the data unit conflict within the data level conflicts are expanded. All other types of conflicts are also encoded as children concepts and instances of these concepts to represent the various semantic conflicts that may arise in the domain of the heterogeneous databases. This common ontology is then mapped to the individual local schemas and also to the federated schema so that each type of semantic conflict may be detected as needed.

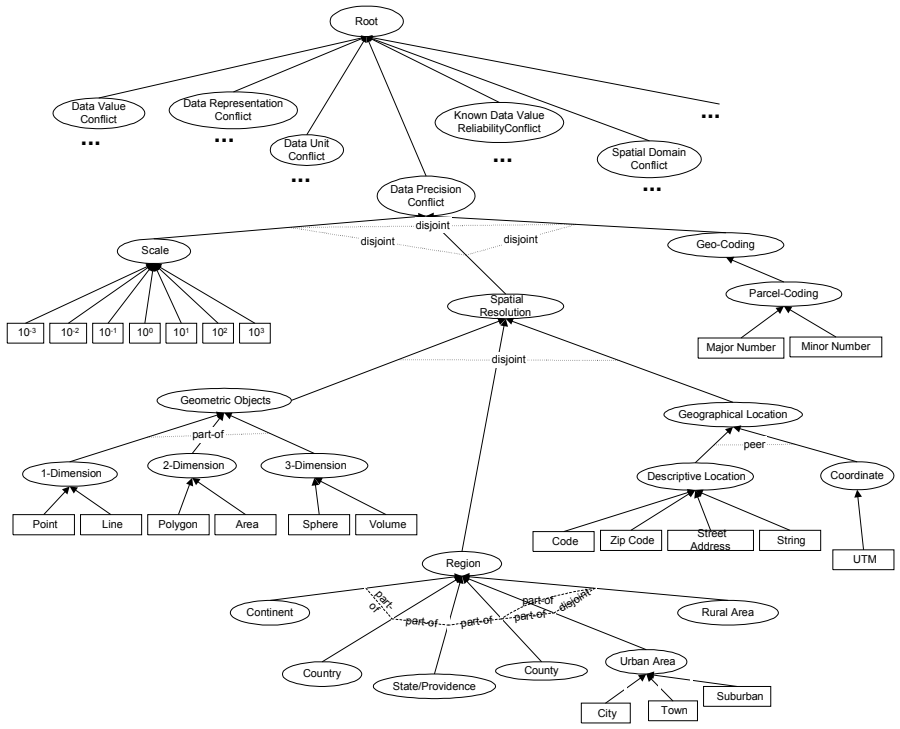

Fig. 4. Data Precision Conflicts in SCROL

4.2 Step 2: Defining Conflict Controller, Original Context, and Target Context

In order to detect a certain semantic conflict, three pieces of information are needed from SCROL: a conflict controller, an original context, and a target context. The *conflict controller* is a concept that has child concepts or instances. Each of these represents multiple interpretations of semantically related objects. Each conflict controller has a corresponding semantic resolver that handles semantic reconciliation between the conflict controller's child concepts or instances (see Fig. 5). The *original context* is a concept or an instance that has been mapped to the local schema component to represent the specific context of the local schema component. The *target context* represents the resulting context in which the user wants to transform the meanings of data values from local databases.

From the previous example, a concept Area has two instances Square Meter and Acre, and the mapping relationship between the two instances is *total one-one*. Thus, any data value mapped to

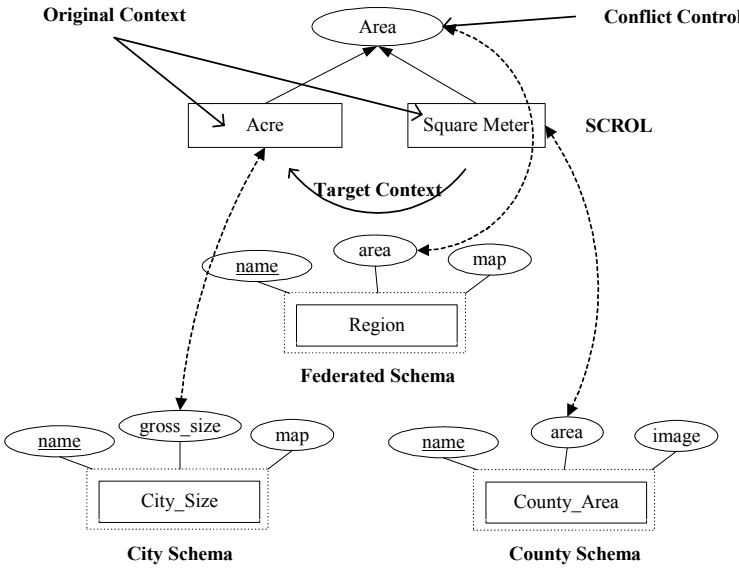

Fig. 5. Conflict Controller, Original Context and Target Context

Square Meter can be converted to the corresponding value in Acre and vice versa. As illustrated in Fig. 5, the instance Square Meter is called *original context* of COUNTY_AREA.area, because the attribute area is mapped to the instance Square Meter and actual data values stored in the area are represented in square meter (m^2). Similarly, the instance Acre is also called *original context* of the attribute gross_size in CITY-SIZE. The parent concept of Square Meter and Acre (i.e., Area) is referred to as a *conflict controller*, which is mapped to REGION.area in the federated schema. When a user retrieves data sources mapped to Square Meter and wants to convert his output to Acre, the instance Acre becomes a *target context*. In this particular case, the conflict controller, Area, has the corresponding semantic resolver that contains two simple conversion rules, one converting values from square meter to acre and the other converting values from acre to square meter. The semantic resolver then captures conflict resolution knowledge about how to transform data values from COUNTY_AREA.area, which is originally in square meter, to values in acre (i.e., 1 acre = 4047 square meters). If a new instance (e.g., Square Feet) is added to the concept (e.g., Area) later, a domain expert creates associated new semantic transformation rules and encodes such

rules into the appropriate semantic resolver.

4.3 Step 3: Defining Ontology Relationship and Mapping Knowledge

In order to determine whether two terms are semantically related and, if they are related, to what extent one can be translated to another, it is necessary to explicitly describe the relationship between SCROL components. Thus, the next step requires the construction of the *ontology relationship knowledge* defined as follows:

Definition 7. Let *ORK* be the ontology relationship knowledge. *ORK* is defined as a relation on $\sigma \times \sigma \times \lambda$, where $\sigma \in OC \cup OI$ in Λ and $\lambda \in RS \cup RM$ in Λ (see Definitions 1, 2, 3, 4 and 5). It is given by $ORK \subseteq \sigma \times \sigma \times \lambda$. Note that *ORK* is derived from *RS* and *RM*.

According to Fig. 6, Acre and Square Meter are specific instances of concept Area in SCROL, and these two instances have a *total one-one* mapping because every value in Acre is associated with exactly one value on in Square Meter and vice versa. In this example, the ontology relationship knowledge can be expressed as:

ORK = { (‘Square Meter’, ‘Acre’, ⟷), (‘City’, ‘County’, *part-of*)}

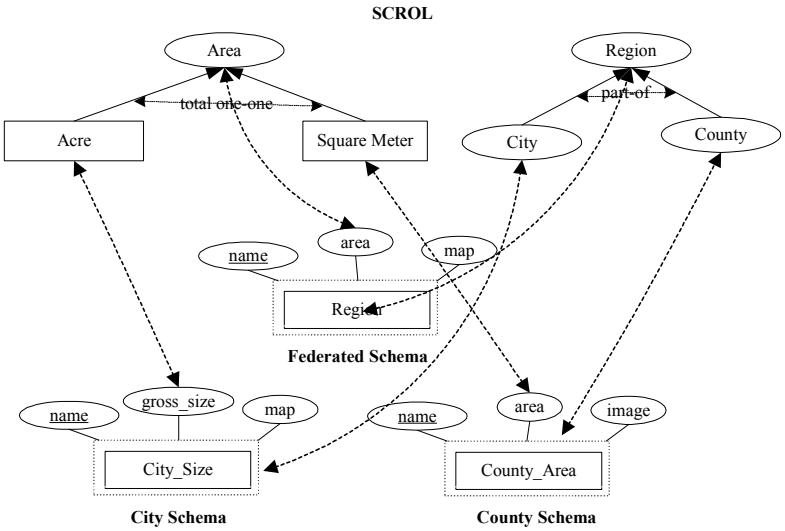

Fig. 6. An Example of Ontology Mappings

After the construction of *ORK*, the next step is to establish mapping knowledge between SCROL and federated/local schemas. Contextual knowledge in a database component is captured via mappings between SCROL and the schema of the component. Therefore, the *ontology mapping knowledge* is defined as follows:

Definition 8. Let *OMK* be the ontology mapping knowledge. *OMK* is a set of mappings between schema (both federated and local schemas) and SCROL. *OMK* is, therefore, defined as a relation on SCROL $\times$ (*FS* $\cup$ *LS*) and given by *OMK* $\subseteq$ SCROL $\times$ (*FS* $\cup$ *LS*), where *FS* is a federated schema and *LS* is a set of participating local schemas.

For example, as Fig. 6 illustrates, CITY_SIZE.gross_size is mapped to an instance Acre, which specifies that the measurement unit of CITY_SIZE.gross_size is acre. On the other hand, COUNTY_AREA.area is mapped to Square Meter, which indicates that the measurement unit of COUNTY_AREA.area is square meter. By looking at the given ontology mapping knowledge, the system is able to detect the fact that the two attributes are represented in different measurement units. Also note that REGION.area in the federated schema is mapped to concept Area. This particular mapping knowledge indicates that REGION.area is a conflict controller and, thus, its measurement unit could be either acre or square meter. It allows the user to select his or her target context (either acre or square meter in this case), which is the resulting context in which the user wants to view the retrieved data. The ontology mapping knowledge can be established at any level (e.g., attribute, entity, or relationship level). At each level, either a concept or instance in SCROL is mapped to a schema component. For example, the entity CITY_SIZE is mapped to concept City and the entity COUNTY_AREA is mapped to concept County, where the City is *part-of* Country. The established mappings are stored in the ontology-mapping repository. The ontology mapping knowledge can be explicitly represented in the following form:

OMK = { (‘Area’, REGION.area), (‘Acre’, CITY_SIZE.gross_size), (‘Square Meter’, COUNTY_AREA.area), (‘Region’, REGION), (‘City’, CITY_SIZE), (‘County’, COUNTY_AREA) }

5 Application and Evaluation of SCROL

In this section, we describe how SCROL is used to detect and resolve a variety of data and schema conflicts. Due to the space limitations, only one example of a data-level conflict and one example of a schema-level conflict is described in detail. We also summarize the result of a detailed evaluation.

5.1 Illustrative Examples to Demonstrate Application of SCROL

5.1.1 Data Level Conflicts

As an example, two heterogeneous databases from the land-use heterogeneous databases case are introduced in Fig. 7. They contain information on land parcels and buildings. Note that Fig. 7 displays only the attributes and entities that are necessary to explain data precision conflicts. However, schema mappings that are already established between the federated schema and two local schemas are not shown in Fig. 7 due to limited space. LAND_PARCEL.temperature in local schema 1 and LAND_PARCEL.avg_temperature in local schema 2 are semantically equivalent, and both are mapped to LAND_PARCEL.avg_temperature in the federated schema. Similarly, CI_BUILDING.gross_area and RES_BUILDING.area_in_square_feet in local schema 1 and CI_BUILDING.gross_area_ci and RES_BUILDING.total_housing_unit_area_rb in local schema 2 are mapped to CI_BUILDING.gross_area and RES_BUILDING.area_in_square_feet in the federated schema, respectively.

There are three data unit conflicts in this example:

- The measurement unit of LAND_PARCEL.temperature in local schema 1 is Celsius, whereas the unit of LAND_PARCEL.avg_temperature in local schema 2 is Fahrenheit.

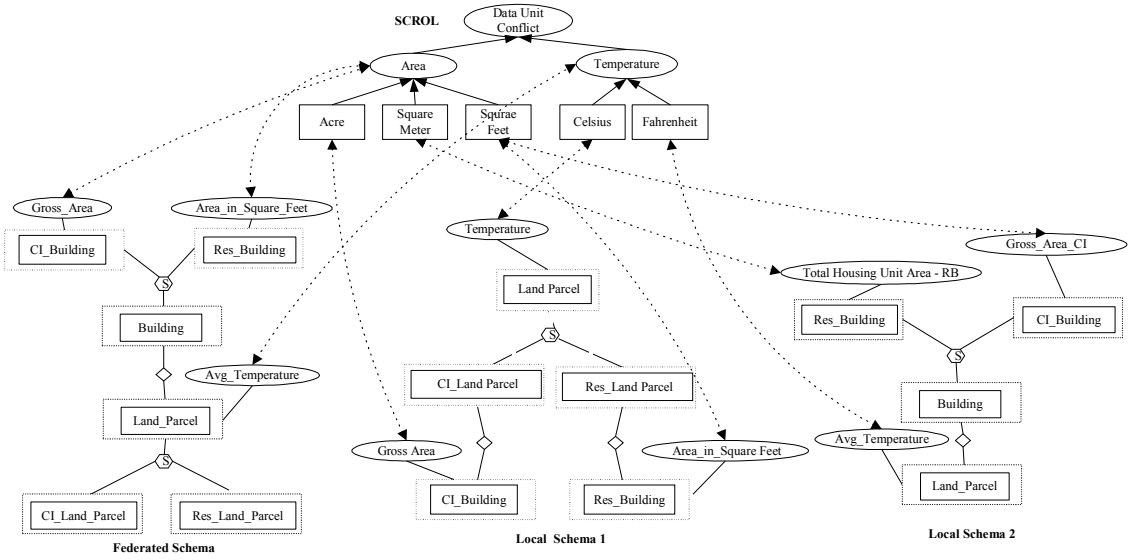

Fig. 7. An Example of Data Unit Conflicts

- CI_BUILDING.gross_area in local schema 1 is in acres, while CI_BUILDING. gross_area_ci in local schema 2 is in square feet.
- RES_BUILDING.area_in_square_feet in local schema 1 is in square feet, whereas RES_BUILDING.total_housing_unit_area_rb in local schema 2 is in square meters.

As depicted in Fig. 7, these conflicts are identified by ontology mappings. In SCROL, Area and Temperature are conflict controllers, and they have a *disjoint* relationship between each other. Recall that the *disjoint* relationship between two concepts indicates that they are not semantically equivalent. Acre, Square Feet, and Square Meter have *total one-one* mapping relationships because their data values can be transformed from one to another without losing their meanings. Similarly, Celsius and Fahrenheit have a *total one-one* mapping relationship.

Based on the ontology mapping of the local schemas to SCROL, we may readily recognize data unit conflicts since each of the local schemas have attributes distinctly mapped to different instances of a concept. For instance, Land_Parcel.Temperature in local schema 1 is mapped to the Celsius instance of the Temperature concept, whereas Land_Parcel.Avg_Temperature in local schema 2 is mapped to the Fahrenheit instance of the same concept. After the initial detection of the semantic conflict, the appropriate transformation rule may be used to resolve the conflict and

convert the data values to any one of the data units captured in SCROL that represents the same concept. For example, if the user queries LAND_PARCEL.avg_temperature from the federated schema, the two conflicting contexts (i.e., Celsius and Fahrenheit) are recognized from the conflict controller, Temperature, and the user is asked to specify his or her target context. If the user wants to view the data in Celsius from the underlying two local databases, the user can select Celsius as a target context. For the local schema 1, no semantic reconciliation process is required because the original and target contexts are identical. On the other hand, the original and target contexts of LAND_PARCEL.avg_temperature in schema 2 are different. Therefore, each data value retrieved from local schema 2 is converted to data value in Celsius according to the transformation rules and then the resulting data retrieved from both local schemas are presented to the user. The other two conflicts can also be detected easily and resolved in a similar way.

5.1.2 Schema Level Conflicts

Consider the three different databases representing information about taxes shown in Fig. 8. All describe the sum of collected tax amounts (in thousands) of different taxes each year for a particular county. In local schema 1, REAL_PROPERTY stores one tuple per year per tax type (REAL_PROPERTY.tax_type) with amount of tax paid. TAX_AMOUNT in local schema 2 contains one tuple per year, and one attribute for each tax type (TAX_AMOUNT. surface_water_bill and TAX_AMOUNT.property_tax), where the value of the attribute is the tax amount paid. Local schema 3, in contrast, has one relation for each tax (SURFACE_WATER_ BILL and PROPERTY_TAX) and each relation has one tuple per year with its tax amount paid (SURFACE_WATER_BILL.amount and PROPERTY_TAX.amount). Hence, this situation is a typical schema discrepancy conflict.

As shown in Fig. 8, these conflicts are identified by ontology mappings. Note that the ontology mapping for the attribute, tax_type, in local schema 1 is defined as mapping-by-value because each tuple value itself is a tax type; that is, either "property tax" or "surface water bill." During the semantic reconciliation, the Conflict Resolver determines the tax type in the local schema 1 by looking at each data value stored in the REAL_PROPERTY.tax_type and then assigns each data value captured in the REAL_PROPERTY.tax_amount into the corresponding attribute in the federated schema. In the case of local schema 3, the name of each tax type is the name of an entity class. SURFACE_WATER_BILL.amount is mapped to Surface_Water_Bill and PROPERTY_TAX.amount is mapped to Property_Tax in SCROL. Thus, if the user queries tax amounts of surface water bills through REAL_PROPERTY.surface_water_bill from the federated schema, then only data from SURFACE_WATER_BILL.amount in local schema 3 is retrieved.

5.2 Evaluation of SCROL

We tested SCROL using three different cases, each of which represents a different application domain. The first case, called "land-use heterogeneous databases," involved three heterogeneous databases used in land-use applications. These databases were developed to store information about

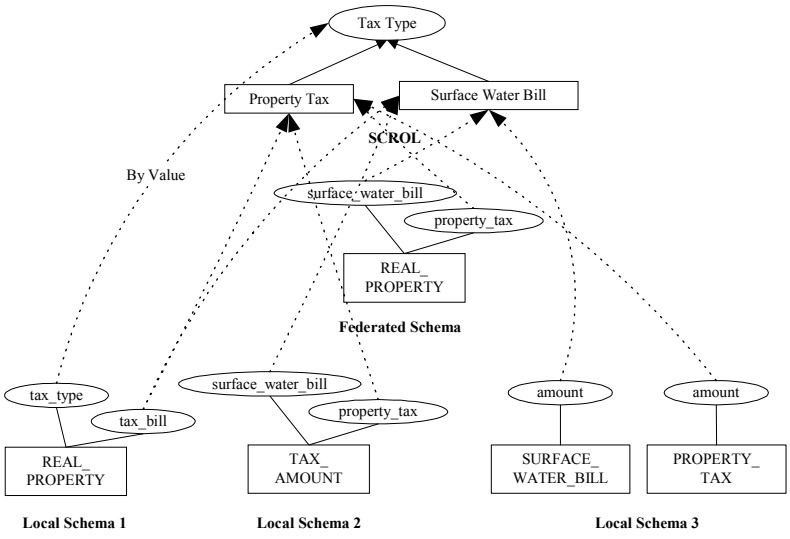

Fig. 8. An Example of Schematic Discrepancies

buildings constructed in residential or commercial/industrial areas. We adopted the heterogeneous database schemas originally described in [22] and modified and expanded them to include as many semantic conflicts as possible. The second case, "ecology heterogeneous databases," involved a set of databases used by ecologists for ecological system analysis at a large research organization. It consisted of three databases designed for different purposes. The first captured soil-related information; the second was designed for fire simulation, while the third was used to capture vegetation and different water types. The third case, called "publication databases," involved two databases for publications, each of which was independently developed by researchers and librarians. The two database schemas were originally used by [1] to demonstrate schema integration methodology. We modified and extended the two schemas to evaluate and demonstrate the performance of our system.

The first two cases are categorized as being in the geographic database domain, while the third deals with non-geographic databases. The two geographic database cases were selected because temporal and spatial dimensions of geographic data objects are frequent and significant sources of semantic conflicts that lead to multiple interpretations [23]. Finally, the publication case was also selected in order to validate the effectiveness of SCROL in a non-geographic domain. We selected the specific cases from among many other databases because the three cases were representative of commonly used geographic and non-geographic databases and the three cases, combined, contained all of the different types of semantic conflicts in the classification framework of semantic conflicts described in Section 4.1. The initial construction of SCROL consists of about 120 concepts and 160 instances. Fig. 9 shows the initial construction of SCROL in zoom-out mode. All the local schemas and federated schemas for all three cases were then mapped to SCROL.

Three different cases consisting of 123 relations and 434 attributes were tested to evaluate SCROL. Among the total of the 557 database components (i.e., the sum of 123 relations and 434

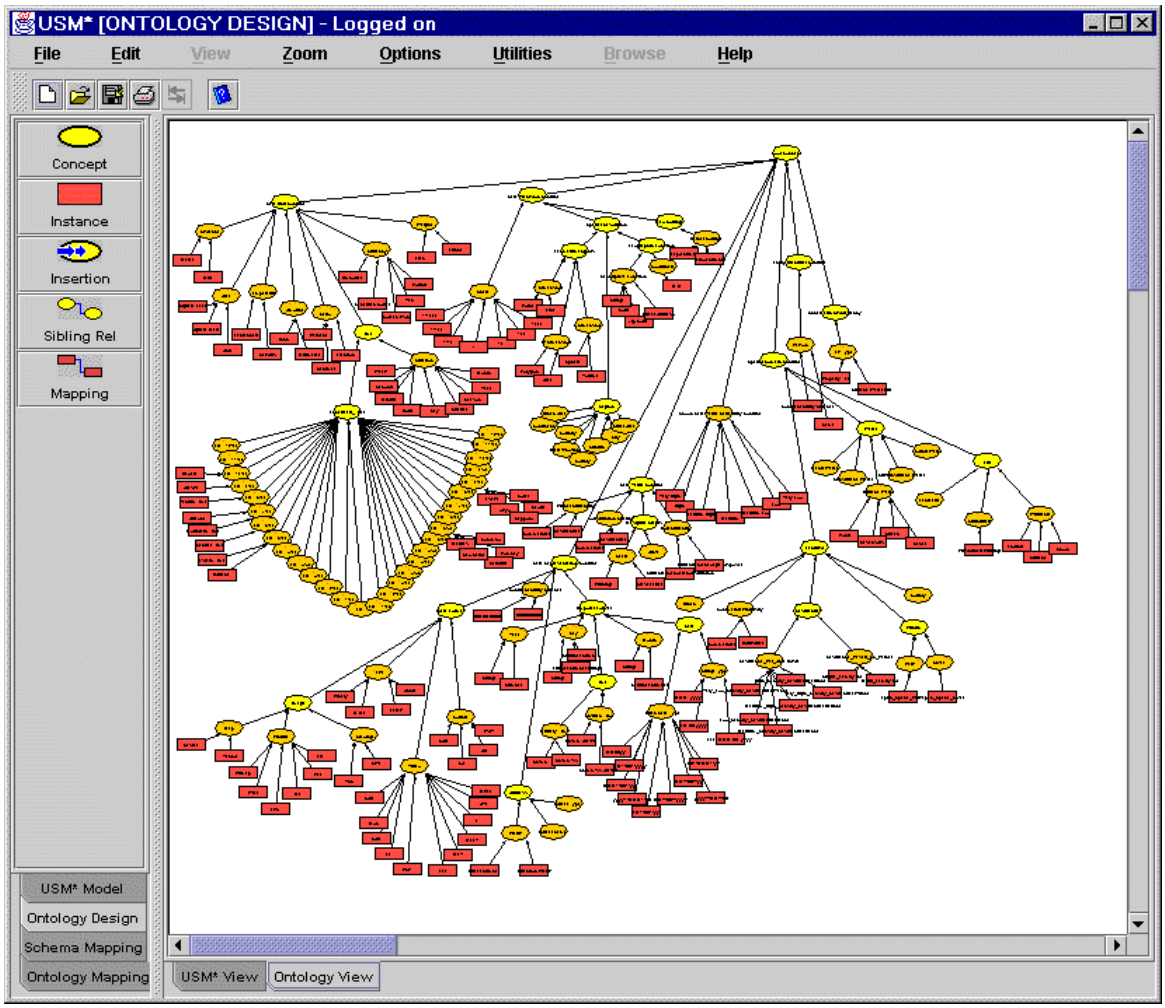

Fig. 9. Initial Construction of SCROL in Zoom-out Mode

attributes) used during the evaluation phase, 201 components involved semantic conflicts. The 201 heterogeneous components caused 68 conflicts: 24 data-level conflicts and 44 schema-level conflicts. Table 4 shows a summary of the results obtained from the three cases.

All of the conflicts found at the data level except for "known data value reliability" and "spatial domain conflicts" were resolved by the semantic mediators using SCROL and ontology mapping knowledge. Known data value reliability conflicts cannot be resolved semantically because they are not the cause of multiple interpretations of the same data, i.e., semantic heterogeneity. The differences in data source reliability are generally due to the use of different measuring instruments, which affect the precision and accuracy of data being collected. Spatial

Table 4
Summary of Semantic Conflicts Resulting from the Three Cases

Conflict Type		Number of Conflicts Found	Number of Conflict Resolved
Data-Level Conflicts	Data Value	2	2
	Data Representation	5	5
	Data Unit	6	6
	Data Precision	4	4
	Known Data Value Reliability	6	0*
	Spatial Domain	1	0*
Schema-Level Conflicts	Naming	20	20
	Entity Identifier	2	2
	Schema Isomorphism	7	7
	Generalization	3	3
	Aggregation	6	6
	Schematic Discrepancies	6	6

* only partially supported

domain conflicts are also not easily resolved by the semantic mediators, since differences in spatial domain specifications are legally defined by different people who have different needs in different application domains. However, as long as source database schemas are properly mapped to the matching ontology components and these conflicts can be classified in SCROL, the semantic mediators can automatically detect differences between them and provide a partial solution by notifying the user which information sources have conflicts. Unlike data-level semantic conflicts, semantic reconciliation of some schema-level conflicts requires both SCROL and schema mapping knowledge. Schema mapping knowledge contains all details about local schema structures [26]. The result also shows that both schema mapping knowledge and ontology mapping knowledge play important roles as knowledge sources for semantic mediators during global query processing. Detailed analyses of the test results can be found in [25]. In conclusion, SCROL plays a pivotal role in our proposed approach. To overcome the limitations of the traditional federated approach (e.g., lack of semantic-richness and flexibility) and the ontology-based domain model approach (e.g.,

complexity), we presented a crossbreed approach that provides enough semantic richness to detect and resolve semantic conflicts while maintaining a much simpler ontology.

6 Contributions and Comparison with Other Approaches

Our major contributions are that SCROL is a domain independent ontology with a simple structure. However, it is powerful enough to identify and resolve a large number of different types of semantic conflicts as explained in the earlier sections.

Our approach is different from the ontology-based domain model approach. The domain model approach attempts to resolve semantic heterogeneity by leveraging the richness of domain knowledge. The purpose of using ontologies is to represent knowledge about a target application domain. It is, however, very difficult to construct and manage a domain model because a domain model is much more complex than a conventional shared schema. A domain model requires capturing tacit knowledge within a certain domain in great detail in order to clarify meaning of each data object. To release this burden, many researchers in the AI community have made efforts on knowledge sharing and reusability [27]. Along with these endeavors, the need of ontology standardization has been addressed. To date, however, there is no formal agreement on these efforts. Furthermore, this approach requires a significant amount of reasoning with a domain model whenever it attempts to resolve even a simple semantic conflict because the model itself is inherently complex. We believe, however, that a domain model does not need to be complex if its purpose is only to detect and resolve semantic conflicts. This can be accomplished by avoiding integration of all tacit knowledge relevant to a particular domain application.

The Context Interchange (COIN) uses a domain model [8]. A domain model in COIN is a collection of primitive types and *semantic types* (similar to *type* in the object-oriented paradigm), which defines the application domain corresponding to the data sources that are to be integrated. Each type in a domain model binds a specific context ("semantic value") that is provided by local

data sources when a query occurs. The data value is exchanged from one system to another by converting the semantic value from its source context to its receiving context through a "context mediator." The context mediator takes control of all requests for data from the receiver (i.e., application), then translates the query to the source context, executes the query in the source database, and finally converts the data into the receiver's context. Conversion functions are used to convert semantic values from one context to another. The context information can be obtained by examining the data environment of each data source. A data environment specifies the context of data values and may involve mappings, lookup tables, rules, predicates, or other knowledge representations. A domain model (i.e., shared-ontology) specifies terminology mappings. These mappings describe naming equivalencies among the component information systems, so that references to attributes, meta-attributes, and their values in one information system can be translated to the equivalent names in another. This approach, however, focuses solely on the semantics of individual data items, i.e., data-level conflicts. Since the context information is represented at the level of data values, the context mediation proposed in this approach is not able to resolve semantic conflicts at the schema level. In conclusion, this work, as indicated in Goh et al. [8], complements rather than competes with the existing domain model approach.

While the ontology-based domain model approach suffers from its complexity, the other two popular approaches to heterogeneous database integration, the global schema and the federated schema approach, lack semantic-richness and flexibility [32]. In the global schema approach, each local database schema is combined into a single integrated schema [2]. It is an integration of component database systems that may not be autonomous. A federated approach, on the other hand, assumes a collection of cooperating but autonomous component database systems [36]. In the federated approach, each local database exports a portion of its schema that it is willing to share with other databases [7]. Each database can then use these export schemas to define an import

schema, a partial global schema representing information from remote databases that is accessible locally.

In the federated approach, two methodologies have been proposed: the tightly coupled approach and the loosely coupled approach [36]. The distinction is based on who manages the federation and how the component databases are integrated. In the loosely coupled approach, a federation is created and managed by the user and there is no centralized control over the system. Users can directly interact with local databases instead of being restricted to querying federated schemas. One of the main drawbacks of this approach is that it places too heavy a burden on users by requiring them to understand the underlying local databases and provides little or no support for identifying semantic conflicts [9]. This approach typically requires users to engage in the detection and resolution of semantic conflicts. Consequently, users are also responsible for semantic conflict resolution. A federated system is tightly coupled if the federation is created and maintained by administrators, not by users. One or more federated schemas are constructed from the schemas of the participating local databases. In most cases, the schemas of the local databases (called "component databases") are heterogeneous. This approach provides uniform and integrated access to the underlying local databases, because the federated schema serves as a front-end system that supports a canonical data model (common model) and a single global query language on top of these local database systems. In this approach, semantic conflicts must be identified and resolved a priori by the administrators.

Both approaches are traditional and many researchers have identified a scalability problem in these approaches [8, 33]. The tightly coupled approach is decent in that it supports well-integrated data access to distributed, heterogeneous databases because this approach relies on a highly integrated shared schema. However, it is very difficult to construct and maintain a federated schema that represents all of the participating data sources without any semantic conflicts. The

loosely coupled approach is reasonable in that it requires little central administration. In other words, it is the user who creates and maintains a federated schema. However, due to the decentralization, users must have knowledge on the data sources to resolve the potential conflicts. Human users must be engaged in detecting and resolving conflicts.

Our paper presents a hybrid approach by addressing the limitations of these approaches. While achieving enough semantic richness to detect and resolve semantic conflicts, our approach successfully retains SCROL at a lower level of complexity. We have a federated schema, but our approach requires only a reasonable amount of schema integration effort because semantic conflicts do not have to be identified and resolved a priori by a centralized administrator. The specific context of each data object in the schema is identified through mappings to the extendable, domain independent ontology, and the local database administrators and domain experts are responsible for encoding semantic conflict classification knowledge. Furthermore, SCROL was not designed to represent domain knowledge; thus, it does not have to be complex.

7 Conclusion and Future Research

We have presented details of an ontology called SCROL in this paper. SCROL can be used to identify and resolve semantic conflicts among multiple heterogeneous databases. We have also presented results an evaluation of SCROL to show how it comprehensively identifies and resolves many different types of conflicts.

There are several interesting extensions, which we are currently exploring. The research presented in this paper assumes that the underlying information sources are structured data that may reside in the structurally organized text files or database systems. However, the unprecedented growth of Internet technologies has made vast amounts of resources instantly accessible to various users via the World Wide Web (WWW). The majority of data available on the Internet are semi-structured or unstructured, consisting of multimedia data such as audio, video, and image files as

well as textual documents such as Hypertext Markup Language (HTML). Whereas HTML is semi-structured and designed only to present information to human users, the promising Extensible Markup Language (XML) technology provides a highly structured format that can be processed by machines on the Internet. However, it is still required to capture context knowledge on the meaning of each tag for semantic interoperability among heterogeneous XML formats. We are currently investigating XML as a standard method for exchanging various knowledge captured in SCROL with other systems on the Internet.

We are also extending our work to semantic interoperability in digital libraries. A digital library is a networked system environment where individual components interact to allow users to submit and access digital content. Among many issues and research directions in digital libraries identified by Ram et al. [31], ensuring interoperability and knowledge-based information sharing is one of the key aspects of successful implementation of digital libraries. Mediator/agent-based business transactions on the Internet (i.e., electronic commerce, or EC) is another interesting extension of the current research. In the multi-agent systems (MAS) environment, it is natural to deal with heterogeneous agents that attempt to communicate with one another in different agent communication languages (ACL). Since each ACL is differentiated from others in terms of its syntax as well as semantics (e.g., FIPA-ACL and KQML), the interoperability issue in this area becomes interesting. We propose to extend these issues as an important direction for extending the research described in this paper.

References

[1] C. Batini and M. Lenzerini, "A Methodology for Data Schema Integration in the Entity Relationship Model," *IEEE Trans. Softw. Eng.*, vol. SE-10, no. 8, 1984, pp. 650-664.

[2] C. Batini, M. Lenzerini, and S.B. Navathe, "A Comparative Analysis of Methodologies for Database Schema Integration," *ACM Comp. Surveys*, vol. 18, no. 4, 1986, pp. 323-364.

[3] C. Collet, M.N. Huhns, and W.-M. Shen, "Resource Integration Using a Large Knowledge Base in Carnot," *IEEE Computer*, vol. 24, no. 12, 1991, pp. 55-62.

[4] B. Czejdo, M. Rusinkiewics, and D.W. Embley, "An Approach to Schema Integration and Query Formulation in Federated Database Systems," in *Proceedings of Int. Conf. Data Eng.*, Los Angeles, CA, 1987, IEEE Computer Society Press, pp. 477-484.

[5] S. Dao and B. Perry, "Applying a Data Miner to Heterogeneous Schema Integration," in *Proceedings of International Conference on Knowledge Discovery and Data Mining*, Montreal Quebec, Canada, 1995, AAAI Press, pp. 63-68.

[6] M.L. Dowell, L.M. Steohens, and R.D. Bonnell, "Using a Domain-Knowledge Ontology as a Semantic Gateway among Information Resources," in Machael N. Huhns and Munindar P. Singh, ed., *Readings in Agents*, San Francisco: Morgan Kaufman, 1998, pp. 255-260.

[7] D. Fang, J. Hammer, and D. McLeod, "The Identification and Resolution of Semantic Heterogeneity in Multidatabase Systems," in *Proceedings of IMS*, Kyoto, Japan, 1991, pp. 136–143.

[8] C.H. Goh, S. Bressan, S.E. Madnick, and M.D. Siegel, "Context Interchange: New Features and Formalisms for the Intelligent Integration of Information," *ACM Transactions on Information Systems*, vol. 17, no. 3, 1999, pp. 270-293.

[9] C.H. Goh, S.E. Madnick, and M.D. Siegel, "Context Interchange: Overcoming the Challenges of Large-Scale Interoperable Database Systems in a Dynamic Environment," in *Proceedings of CIKM*, Gaithersburg, Maryland, 1994, pp. 337-346.

[10] T.R. Gruber, "The Role of Common Ontology in Achieving Sharable, Resuable Knowlege Bases," in *Proceedings of KR*, 1991, pp. 601-602.

[11] T.R. Gruber, *Toward Principles for the Design of Ontologies Used for Knowledge Sharing*, Technical Report KSL 93-04, Knowledge Systems Laboratory, Stanford University, August 23 1993.

[12] T.R. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition*, vol. 5, no. 2, 1993, pp. 199-220.

[13] M.N. Huhns and M.P. Singh, "Managing Heterogeneous Transaction Workflows with Co-operating Agents," in Nicholas R. Jennings and Michael J. Wooldridge, ed., *Agent Technology: Foundations, Applications, and Markets*, Berlin: Springer, 1998, pp. 219-239.

[14] J. Kahng and D. McLeod, "Dynamic Classificational Ontologies: Mediation of Information Sharing in Cooperative Federated Database Systems," in Michael P. Papazoglou and Gunter Sohlageter, ed., *Cooperative Information Systems: Trends and Directions*, San Diego: Academic Press, 1998, pp. 179-203.

[15] V. Kashyap and A.P. Sheth, *Semantic and Schematic Similarities Between Objects in Databases: A Context-Based Approach*, Technical Report TR-CS-95-001, Department of Computer Science, The University of Georgia, February 16 1995.

[16] V. Kashyap and A.P. Sheth, "Semantic and Schematic Similarities Between Database Objects: A Context-based Approach," *The VLDB Journal*, vol. 5, no. 4, 1996, pp. 276-304.

[17] D.B. Lenat, "CYC: A Large-Scale Investment in Knowledge Infrastructure," *Comm. ACM*, vol. 38, no. 11, 1995, pp. 33-38.

[18] D.B. Lenat, R.V. Guha, K. Pittman, D. Pratt, and M. Shepherd, "CYC: Toward Programs with Common Sense," *Comm. ACM*, vol. 33, no. 8, 1990, pp. 30-49.

[19] R. MacGregor, "The Evolving Technology of Classification-Based Knowledge Representation Systems," in John F. Sowa, ed., *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, San Mateo: Morgan Kaufmann, 1991, pp. 385-400.

[20] K. Mahalingam and M.N. Huhns, "An Ontology Tool for Query Formulation in an Agent-Based Context," in *Proceedings of IFCIS International Conference on Cooperative Information Systems*, Kiawah Island, South Carolina, 1997

[21] K. Mahesh and S. Nirenburg, "A Situated Ontology for Practical NLP," in *Proceedings of IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing*, Montreal, Canada, 1995

[22] T.L. Nyerges, "Schema Integration Analysis for the Development of GIS Databases," *Int. J. Geographical Info. Sys.*, vol. 3, no. 2, 1989, pp. 153-183.

[23] A.M. Ouksel, "A Framework for a Scalable Agent Architecture of Cooperating Heterogeneous Knowledge Sources," in M Klusch, ed., *Intelligent Information Agents: Agent-Based Information Discovery and Management on the Internet*, Berlin: Springer, 1999, pp. 100-124.

[24] A.M. Ouksel and C.F. Naiman, "Coordinating Context Building in Heterogeneous Information Systems," *J. of Intelligent Information Systems*, vol. 3, no. 2, 1994, pp. 151-183.

[25] J. Park, *Facilitating Interoperability among Heterogeneous Geographic Database Systems: A Theoretical Framework, A Prototype System, and Evaluation*, Dissertation, The University of Arizona, 1999.

[26] J. Park and S. Ram, "A Conflict Resolution Environment for Autonomous Mediation Among Heterogeneous Databases," Technical Report 01-04-1, University of Minnesota, April 2001.

[27] R.S. Patil, R.E. Fikes, P.F. Patel-Schneider, D. McKay, T. Finin, T. Gruber, and R. Neches, "The DARPA Knowledge Sharing Effort: Progress Report," in *Proceedings of KR*, Cambridge, MA, 1992, Margan Kaufmann, pp. 777-788.

[28] J.-C.R. Pazzaglia and S.M. Embury, "Bottom-up Integration of Ontologies in a Database Context," in *Proceedings of KRDB Workshop*, Seattle, WA, 1998

[29] S. Ram, J. Park, and G. Ball, "Semantic Model Support for Geographic Information Systems," *IEEE Computer*, vol. 32, no. 5, 1999, pp. 74-81.

[30] S. Ram, J. Park, K. Kim, and Y. Hwang, "A Comprehensive Framework for Classifying Data- and Schema-Level Semantic Conflicts in Geographic and Non-Geographic Databases," in *Proceedings of WITS*, Charlotte, North Carolina, 1999, pp. 185-190.

[31] S. Ram, J. Park, and D. Lee, "Digital Libraries for the Next Millenium: Challenges and Research Directions," *ISF*, vol. 1, no. 1, 1999, pp. 75-94.

[32] S. Ram and V. Ramesh, "Schema Integration: Past, Current and Future," in A. Elmagarmid, M. Rusinkeiwicz, and Amit P. Sheth, ed., *Management of Heterogeneous and Autonomous Database Systems*, San Francisco: Morgan Kaufmann, 1999, pp. 119-155.

[33] V. Ramesh, K. Canfield, S. Quirologico, and M. Silva, "An Intelligent Agent-based Architecture for Interoperability among Heterogeneous Medical Databases," in *Proceedings of Americas Conference on Information Systems*, Phoenix, Arizona, 1996, pp. 549-551.

[34] J. Roitman, *Introduction to Modern Set Theory*, New York: John Wiley & Sons, 1990.

[35] A.P. Sheth, "Semantic Issues in Multidatabase Systems," *SIGMOD Record*, vol. 40, no. 4, 1991, pp. 5-9.

[36] A.P. Sheth and J.A. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases," *ACM Comp. Surveys*, vol. 22, no. 3, 1990, pp. 184-236.

[37] M. Siegel, S. Madnick, and E. Sciore, "Context Interchange in a Client-Server Architecture," *J. Systems Software*, vol. 27, no. 3, 1994, pp. 223-232.

[38] V.C. Storey, R.H.L. Chiang, D. Dey, R.C. Goldstein, and S. Sundaresan, "Database Design with Common Sense Business Reasoning and Learning," *ACM Trans. Database Syst.*, vol. 22, no. 4, 1997, pp. 471-512.

[39] H. Takeda, K. Iwata, M. Takaai, A. Sawada, and T. Nishida, "An Ontology-Based Cooperative Environment for Real World Agents," in *Proceedings of ICMAS*, Kyoto, Japan, 1996, MIT Press, pp. 353-360.

[40] P.E. van der Vet and N.J.I. Mars, "Bottom-Up Construction of Ontologies," *IEEE TKDE*, vol. 10, no. 4, 1998, pp. 513-526.

[41] V. Ventrone and S. Heiler, "Semantic Heterogeneity as a Result of Domain Evolution," *SIGMOD Record*, vol. 20, no. 4, 1991, pp. 16-20.