Querying and Integrating Ontologies Viewed as Conceptual Schemas

Dimitri Theodoratos[1] and Theodore Dalamagas[2]

[1] Dept. of Computer Science
New Jersey Institute of Technology
USA
dth@cs.njit.edu
[2] Dept. of Elec. and Computer Engineering
National Technical University of Athens
Greece
dalamag@dblab.ece.ntua.gr

Abstract. Real-world ontologies have associated data sets. In order to exploit ontologies as a means to provide semantics to the data it is necessary not only to reason about concepts and their properties but also to be able to query the underlying data and efficiently obtain answers. We view ontologies as conceptual schemas that are populated with data. We define a generic graph-based representation for ontology schemas and provide a language for querying ontologies. This language is appropriate for ontology transformation and ontology integration. We show how ontologies can be integrated by defining views over multiple ontologies, and how user queries can be specified using views. Our approach can be easily implemented on top of a relational database management system.

1 Introduction

Ontologies aim at capturing domain knowledge and provide an understanding of that domain which may be used and shared across applications. The current development of the World Wide Web has promoted ontologies as a means to provide semantics to the data. Ontologies typically contain hierarchies of concepts and describe each concepts' features through properties and relations. Many real-world ontologies have associated data instances. In this sense, ontologies can be seen as instantiated conceptual schemas. In order to fully take advantage of ontologies it will be necessary not only to be able to reason with ontology schemas but also with the instances that populate these schemas. Current research efforts have concentrated on how to specify ontologies and on providing formal semantics to ontology languages. The issue of querying ontologies and efficiently obtaining answers over ontology instance sets has been neglected.

When we have various local ontologies, we are often required to build an integrated "global" ontology by extracting information from the local ones. A similar problem appears with data sources where a global schema needs to be constructed over multiple source schemas [15,13]. A typical approach to logical

I.-Y. Song et al. (Eds.): ER 2003, LNCS 2813, pp. 548–561, 2003.

data source integration uses mediators [19,8]. The mediators export a mediator schema which is an integrated representation of data sources. A mediator can be abstractly seen as a view defined over the data sources [17]. The users query the mediator schema, and the views transform these queries between the mediator schema and the data sources. These techniques for integrating data sources can be also applied to the integration of ontologies provided that a query language for ontologies is available.

The main contributions of this paper are the following:

- We present a model for ontologies. Ontologies in this model have schemas and instances. Isa hierarchies and axioms expressing constraints can be specified in a schema. We do not adhere to a specific representation language and we view an ontology schema as a graph.
- We provide a simple language for querying ontologies. Queries in this language are intuitive because they are ontology schemas annotated with relational algebra expressions and other characterizations. Queries can, in addition, specify isa-relationships and axioms.
- A notable feature of the language is that the answers of the queries are ontologies. Therefore, queries can also be used to transform ontologies.
- The query language allows the integration of ontologies through the definition of views over multiple local ontologies.
- We show how the query language can be extended so that queries can be defined using ontology integrating views.
- Since the instances of the ontologies are relations and the expressions that compute the answers of the queries are extended relational algebra expressions, the whole system can be easily implemented on top of a relational data base management system thus taking advantage of its storage and query optimization techniques.

Several formal languages to specify ontologies have been used or proposed, especially for the Semantic Web: Frame Logic [14], Resource Description Language Schema (RDFS) [2], Ontology Inference Layer (OIL) [6,3], DAML+OIL [10], an expressive description logic language SQOD(D) [11] etc. These representation languages differ in their terminology and expressive power but they model ontologies that share most of the features presented here. Query languages for ontologies in [4,12] are conjunctive languages based on description logic. B Amman et al. use a simple sub-language of OQL defining trees to query global schemas viewed as ontologies over local XML data sources [7,1]. All these query languages return only tuples of values and therefore, they are not appropriate for integrating ontologies through the definition of views. The query language in [16] allows querying graph represented schemas in a more general setting but does not take into account isa-relationships and constraints and is not adequate for ontologies.

The rest of the paper is organized as follows. Section 2 introduces ontologies and ontology instances. The query language for ontologies is presented in Section 3. Section 4 shows how ontologies can be integrated by defining views over multiple ontologies, and how queries can then be expressed using views. The last section contains concluding remarks and directions for further research.

2 Ontologies and Ontology Instances

In this section, we formally define ontologies. An ontology specifies a conceptualization of a domain in terms of concepts, properties, roles, isa-relationships and axioms. Ontologies are associated with instances.

We assume three sets **C**, **P**, and **R** of names called *concepts*, *properties*, and *roles* respectively. The concepts represent model entities of interest in the domain. The concepts can have properties. Concepts and properties are collectively called *constructs*. From a conceptual point of view, there is not a clear distinction between concepts and properties besides the fact that properties cannot have other properties. Every concept C in **C** is associated with a set $dom(C)$ of concept instances, common to all the concepts. Every property P in **P** is associated with a set $dom(P)$ of property instances. A *role* is the name of a binary relation between a concept and a property, or between two concepts. The meaning of a role between a concept C and a property P is that concept C can have property P. The meaning of role R between concepts C_1 and C_2 is that concepts C_1 and C_2 can be associated through R. An *isa-relationship* involves two concepts and is denoted $isa(C_1, C_2)$, where C_1 and C_2 are concepts. Isa-relationships are interpreted with subset semantics. Concept C_1 is called subconcept of C_2 and inherits the roles in which C_2 is involved. Isa-relationships are transitive, and can be combined to form hierarchies of concepts. *Axioms* are boolean expression that involve concepts, properties and roles. We consider a number of axioms of specific type. Axioms denoted $rigid(R)$, $unique(R)$, and $identity(R)$, where R is a role between a concept and a property, determine rigidity, uniquenesss and identity features of roles between concepts and properties [9,18]. Axioms denoted $one\text{-}to\text{-}many(R)$, $many\text{-}to\text{-}one(R)$, $one\text{-}to\text{-}one(R)$, where R is a role between two concepts, determine cardinality ratio constraints for roles between two concepts. Axioms denoted $total\text{-}to\text{-}partial(R)$, $partial\text{-}to\text{-}total(R)$ and $total\text{-}to\text{-}total(R)$, where R is a role between two concepts, determine participation constraints for roles between concepts. The meaning of the axioms will be explained later.

Definition 1. An *ontology schema* $\mathcal{S}$ is a quintuple $(\mathcal{C}, \mathcal{P}, \mathcal{R}, \mathcal{I}, \mathcal{A})$, where

(a) $\mathcal{C}$ is a set of concepts.
(b) $\mathcal{P}$ is a set of properties.
(c) $\mathcal{R}$ is a set of roles that does not include more than one role between a concept in $\mathcal{C}$ and a property in $\mathcal{P}$ (in contrast, $\mathcal{R}$ can include multiple roles between two concepts). In particular, for every concept $C \in \mathcal{C}$ (resp. property $P \in \mathcal{P}$), there is a role in $\mathcal{R}$ involving C (resp. P).[1]
(d) $\mathcal{I}$ is a set of isa-relationships $isa(C_1, C_2)$, where $C_1, C_2 \in \mathcal{C}$.
(e) $\mathcal{A}$ is a set of axioms of the type mentioned above where R is a role in $\mathcal{R}$. □

We do not stick with a specific language for ontologies of those that are recently suggested and we use a graph representation.

[1] We assume that roles are defined between distinct concepts. A *recursive* role, that is a role involving the same concept twice, can be modeled using a role between two distinct concepts and mutual isa-relationships between these concepts.

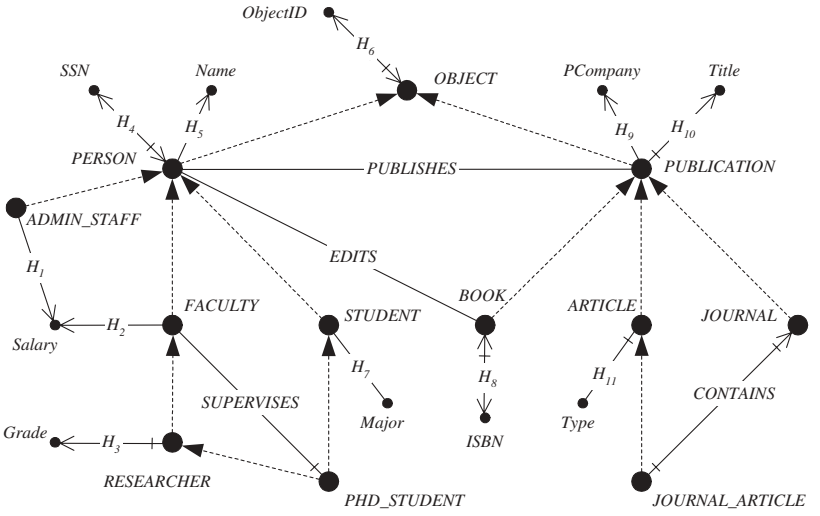

Fig. 1. An ontology schema

Example 1. Figure 1 shows an ontology schema of persons and publications inspired by an ontology presented in [5]. Nodes represent constructs. Nodes shown as bigger filled circles represent concepts, while smaller ones represent properties. Construct names are depicted close to the respective circles. For instance, $PERSON$, $STUDENT$, and $PUBLICATION$ are concepts, and SSN, $Name$, and $ISBN$ are properties. Edges between constructs represent roles. Their names are depicted on the edges. Symbolic names, H_1, H_2 etc. are used for roles between concepts and properties since the meaning of such roles is that a concept has a certain property. For instance, H_5 is a role between concept $PERSON$ and property $Name$, and $PUBLISHES$ is a role between concepts $PERSON$ and $PUBLICATION$.

Dashed arcs represent isa-relationships. For instance the arc from concept $FACULTY$ to concept $PERSON$ identifies $FACULTY$ as a subconcept of $PERSON$. Concept $PHD\_STUDENT$ is a subconcept of concepts $RESEARCHER$ and $STUDENT$ and inherits the roles of both of them.

An arrow on an edge (role) R between a concept and a property pointing to the concept (resp. property) denotes axiom $identity(R)$ (resp. $unique(R)$). For instance the arrow on edge H_4 pointing to concept $PERSON$ denotes axiom $identity(H_4)$, while the arrow on edge H_5 pointing to property $Name$ denotes axiom $unique(H_5)$. A small perpendicular line by the concept on an edge (role) R between a concept and a property denotes axiom $rigid(R)$. For instance, the line on edge H_{10} denotes axiom $rigid(H_{10})$.

An arrow on an edge $R(C_1, C_2)$ between concepts C_1 and C_2 pointing to concept C_1 (resp. C_2) denotes axiom *one-to-many*(R) (resp. *many-to-one*(R)). For instance, the arrow on edge $CONTAINS(JOURNAL, JOURNAL\_ARTICLE)$ pointing to concept

$JOURNAL$ denotes axiom *one-to-many*($CONTAINS$). A perpendicular line on edge $R(C_1, C_2)$ by the concept C_1 (resp. C_2) denotes axiom *total-to-partial*(R) (resp. *partial-to-total*(R)). For instance, the line on the edge $CONTAINS(JOURNAL, JOURNAL\_ARTICLE)$ by the concept $JOURNAL$ denotes axiom *total-to-partial*($CONTAINS$), while the line on the edge $SUPERVISES(FACULTY, PHD\_STUDENT)$ by the concept $PHD\_STUDENT$ denotes axiom *partial-to-total*($SUPERVISES$). The presence of a line by both concepts of edge R denotes axiom *total-to-total*(R); e.g. the lines on edge $CONTAINS$ denote axiom *total-to-total*($CONTAINS$). □

Ontology instances are defined using exclusively role instance sets. The instance set of a role is a binary relation. We follow the approach to the relational model of data that uses attribute names. The names of the attributes are constructs (concepts or properties).

Definition 2. An *instance set of a role between a concept C and a property P* is a binary relation whose schema is (C, P) and whose instance is a finite set of tuples (c, p), where $c \in dom(C)$ and $p \in dom(P)$. An *instance set of a role between two concepts C_1 and C_2* is a binary relation whose schema is (C_1, C_2) and whose instance is a finite set of tuples (c_1, c_2), where $c_1 \in dom(C_1)$ and $c_2 \in dom(C_2)$. In the following we confound a role with its instance set. □

Consider an ontology schema $\mathcal{S} = (\mathcal{C}, \mathcal{P}, \mathcal{R}, \mathcal{I}, \mathcal{A})$. Let C, C_1, C_2 be concepts in $\mathcal{C}$. $\mathcal{R}_\mathcal{C}$ denotes the set of roles in $\mathcal{R}$ between C and a property or a concept.

Definition 3. The *instance set* of concept C, denoted $inst(C)$, is the set of concept instances from $dom(C)$ that appear in instance sets of roles in $\mathcal{R}_\mathcal{C}$. That is, $inst(C) = \bigcup_{R \in \mathcal{R}_\mathcal{C}} \Pi_C(R)$, where Π_C denotes the relational projection operator on attribute C. □

An isa-relationship $isa(C_1, C_2)$ states that $inst(C_1) \subseteq inst(C_2)$. If concept C_2 is involved in a role $R(C_2, X)$, where X is a construct in $\mathcal{S}$, then R is inherited by C_1. The inherited role is denoted $R[C_1]$, if X is a property, and $R[C_1, X]$, if X is a concept. The schema of the instance set of this inherited role is (C_1, X) and its extension is the restriction of the extension of R to tuples that involve elements of $inst(C_1)$. For instance, in Figure 1, role H_4 inherited by concept $RESEARCHER$ is denoted $H_4[RESEARCHER]$, while role $PUBLISHES$ inherited by concepts $STUDENT$ and $ARTICLE$ is denoted $PUBLISHES[STUDENT, ARTICLE]$.

The association of instances of a concept in an ontology with instances of a certain property is not necessarily complete: let $R(C, P)$ be a role between a concept C and a property P; instances of C that are related with property instances of P in R, are not necessarily related to other property or concept instances in other roles involving C, and conversely. Axiom $rigid(R)$ states that for every role $R' \in \mathcal{R}$, $\Pi_C(R') \subseteq \Pi_C(R)$. This axiom requires property P of C to be rigid, that is essential to all instances of concept C. Indeed, $rigid(R)$ implies that $inst(C) = \Pi_C(R)$.

A property instance can be associated with more than one instance of a concept in an ontology. Axiom $identity(R)$ states that the same instance of

H_4

$PERSON$	SSN
p_1	123456789
p_2	987654321
p_3	567891234
p_4	678912345

H_5

$PERSON$	$Name$
p_1	$Smith$
p_2	$Brown$
p_3	$Smith$

$PUBLISHES$

$PERSON$	$PUBLICATION$
p_1	u_3
p_3	u_4

$EDITS$

$PERSON$	$BOOK$
p_1	$u1$
p_2	u_2

H_3

$RESEARCHER$	$Grade$
p_1	$senior$
p_2	$senior$
p_3	$junior$

H_8

$BOOK$	$ISBN$
u_1	$0-8053-1655-4$
u_2	$0-7762-2344-4$

H_{11}

$ARTICLE$	$Type$
u_3	$research$
u_4	$review$

H_9

$PUBLICATION$	$PCompany$
u_1	$Springer$
u_3	AW
u_4	$IEEE$

Fig. 2. Role instance sets

property P is not associated with distinct instances of concept C in R. In this sense, P can act as an identity of C (for those instances of C that are related to instances of P in R). More formally, $identity(R)$ is the boolean expression $\forall c, c' \in dom(C)\ \forall p \in dom(P)\ (R(c,p) \wedge R(c',p) \Rightarrow c = c')$. In relational terms, $identity(R)$ implies that P is a key of relation $R(C,P)$. If $rigid(R)$ holds in addition to $identity(R)$, property P is an identity of C for all instances in $inst(C)$.

A concept instance can be associated with more than one instance of a property in an ontology. Axiom $unique(R)$ states that property P is unique, that is, an instance of concept C is not associated with more than one instance of property P in R. More formally, $unique(R)$ is the boolean expression $\forall c \in dom(C)\ \forall p, p' \in dom(P)\ (R(c,p) \wedge R(c,p') \Rightarrow p = p')$. In relational terms, $unique(R)$ implies that C is a key of relation $R(C,P)$.

Let $R(C_1, C_2)$ be a role between concepts C_1 and C_2. Cardinality ratio axioms *one-to- many*(R), *many-to-one*(R), and *one-to-one*(R) restrict the number of concept instances a concept instance can be related to in R. Axiom *one-to-many*(R) is the boolean expression $\forall c_1, c_1' \in dom(C_1)\ \forall c_2 \in dom(C_2)\ (R(c_1,c_2) \wedge R(c_1',c_2) \Rightarrow c_1 = c_1')$. Axiom *many-to-one*$(R)$ is the symmetric expression. Axiom *one-to-one*(R) is the expression *one-to-many*$(R) \wedge$ *many-to-one*(R).

Participation axioms *total-to-partial*(R), *partial-to-total*(R) and *total-to-total*(R) specify whether all the instances in the instance set of a concept are related to another concept in R. Axiom *total-to-partial*(R) states that $Inst(C_1) \subseteq \Pi_{C_1}(R)$. Axiom *partial-to-total*(R) states that $Inst(C_2) \subseteq \Pi_{C_2}(R)$. Axiom *total-to-total*$(R)$ is the expression *partial-to-total*$(R) \wedge$ *total-to-total*(R).

Definition 4. An *instance* of an ontology schema $\mathcal{S} = (\mathcal{C}, \mathcal{P}, \mathcal{R}, \mathcal{I}, \mathcal{A})$, denoted $inst(\mathcal{S})$, is a set of instance sets of all the roles in $\mathcal{R}$ that satisfy the isa-relationships in $\mathcal{I}$ and the axioms in $\mathcal{A}$. An *ontology* $\mathcal{O}$ is a pair $(\mathcal{S}, inst(\mathcal{S}))$. □

Example 2. Figure 2 shows part of the ontology schema of Figure 1. Only some of the instance sets of the roles are depicted. Clearly, $inst(PERSON) = \{p_1, p_2, p_3, p_4\}$, and $inst(RESEARCHER) = \{p_1, p_2, p_3\}$. Therefore, these role instance sets satisfy the isa-relationship $isa(RESEARCHER, PERSON)$ of the ontology schema of Figure 1. One can see that they also satisfy the axioms $rigid(H_4)$, $unique(H_5)$, and $identity(H_8)$ of this ontology schema. □

3 A Query Language for Ontologies

We present in this section a query language for ontologies. This language adheres to the graph representation for ontologies shown in the previous section. Queries in this language are ontology schemas where roles are associated with expressions. Query answers are ontologies.

Syntax. We first present the syntax of the query language.

Definition 5. A *query* Q over an ontology schema $\mathcal{S}$ is an ontology schema such that:

1. Every property in Q is involved in exactly one role.
2. Every role $R(C, X)$ in Q, where C is a concept and X is a construct, is associated with a relational algebra expression that involves roles from $\mathcal{O}$. Given an instance for $\mathcal{S}$, this expression evaluates to a relation whose schema is (C, X).
3. The roles in Q are characterized as *restrictive* or *non-restrictive*.
4. The constructs in Q are characterized as *shown* or *hidden*. At least one construct in Q is shown. A role $R(C, X)$ in Q is called *shown* if both C and X are shown. Otherwise it is called *hidden*. A shown construct must be involved in at least one shown role. □

Intuitively, shown concepts and roles (as opposed to hidden ones) appear in the answer of the query. Role expressions specify an initial instance set for the roles in this answer. Restrictive roles (as opposed to non-restrictive ones) force the instances in the instance sets of their concepts to be chosen from the instances appearing in their instance set. They do so by restricting the instance sets of the roles with whom they share concepts.

A role expression is built using the usual relational algebra operators: selection (σ_c, where c is a Boolean combination of selection predicates), projection (Π_X, where X is a set of attributes), join ($\bowtie_c$, where c is a conjunction of join predicates), union ($\cup$), intersection ($\cap$), difference ($-$), and attribute renaming ($\rho_{A \to B}$, where A and B are attribute names). Set theoretic operations ($\cup$, $\cap$, $-$) require the operand relations to be union compatible. Role expressions satisfy also the following restrictions: (1) The expression of a role between a concept and a property does not involve join operators. (2) The expression of a role between concepts does not involve selection operators. (3) Selection and join conditions in the expression of a role do not involve concepts.

Semantics. Let Q be a query over an ontology $\mathcal{O} = (\mathcal{S}, inst(\mathcal{S}))$. Let also $e_1, \ldots, e_n,\ n \geq 0$, be the expressions associated with restrictive roles in Q, and

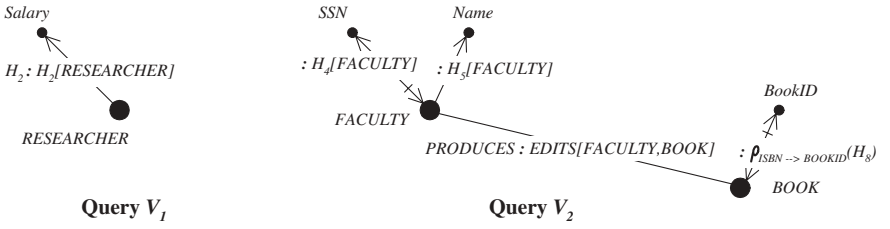

Fig. 3. Queries V_1 and V_2

$e'_1, \ldots, e'_m,\ m \geq 0$, be the expressions associated with non-restrictive roles in Q. We define expression $E = (e_1 \bowtie \ldots \bowtie e_n) \sqsupset\!\bowtie (e'_1 \sqsupset\!\!\sqsubset \ldots \sqsupset\!\!\sqsubset e'_m)$. The *instance set of a role* $R(C, X)$ *in* Q is the relation resulting by evaluating the expression $\Pi_{C,X}(E)$.

Definition 6. The *answer* of Q is the empty set, if the instance sets of the roles in Q do not satisfy the isa-relationships and the axioms in Q. Otherwise, the *answer* of Q is an ontology $\mathcal{O}' = (\mathcal{O}', inst(\mathcal{O}'))$ where:

(a) The ontology schema $\mathcal{O}'$ is defined by the shown constructs and roles and the isa-relationships between shown concepts of $\mathcal{Q}$.
(b) The instance $inst(\mathcal{S}')$ of $\mathcal{S}'$ is the set of the instance sets of the shown roles in Q. □

An alternative representation for $inst(\mathcal{S}')$ in the form of a single relation (possibly containing null values) is provided by evaluating the expression $E' = P_Y(E)$, where P denotes the duplicate and null preserving projection operator, and Y is the set of constructs in S'. Clearly, the instance sets of the roles in S' can be derived from the resulting relation.

In the rest of this section we provide example queries over the ontology schema shown in Figure 1. The queries below involve simple operations and axioms.

Example 3. Figure 3 shows queries V_1 and V_2. The expression of a role follows the name of the role separated by a colon. All the constructs are shown constructs and the all the roles are restrictive. Query V_1 retrieves the researchers and their salaries. The expression that computes the instance of the resulting ontology (in the form of a single relation) is $P_{RESEARCHER,Salary}\ (H_2[RESEARCHER])$.

Query V_2 retrieves the SSN and the $Name$ of faculty members and the $ISBN$ of the books they have edited. The role between $FACULTY$ and $BOOK$ is renamed $PRODUCES$, and the property $ISBN$ is renamed $BookID$. In this query (and quite often in the following) we omit the names of some roles and we mention only their expression. The expression that computes the instance of the resulting ontology is $P_{SSN,Name,FACULTY,BookID}\ (H_4[FACULTY] \bowtie H_5[FACULTY] \bowtie EDITS[FACULTY, BOOK] \bowtie \rho_{ISBN \rightarrow BookID}(H_8))$. □

The next queries involve selection operations and hidden and shown edges.

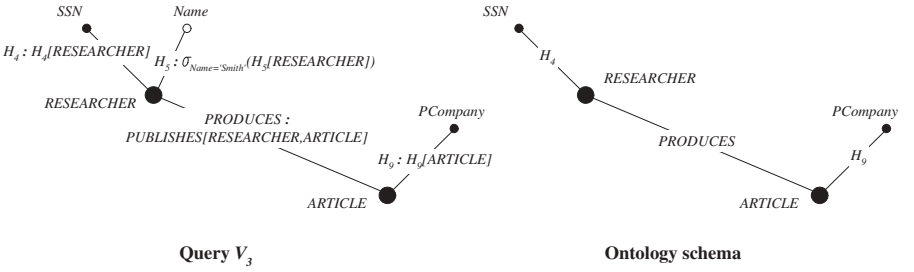

Fig. 4. Query V_3 and the resulting ontology schema

SSN	$RESEARCHER$	$ARTICLE$	$PCompany$
123456789	p_1	u_3	AW
567891234	p_3	u_4	$IEEE$

Fig. 5. Instance of the ontology resulting from query V_3

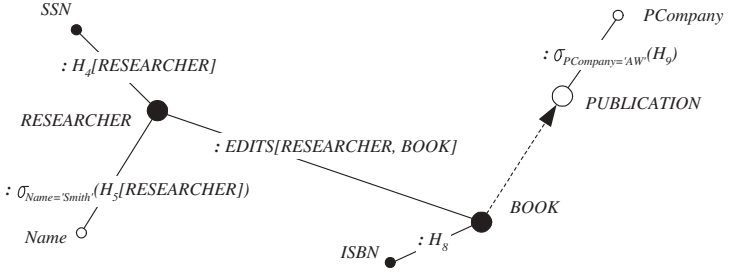

Fig. 6. Query V_4

Example 4. Query V_3 of Figure 4 retrieves the SSN of researchers named 'Smith' and the publishing company of the articles they have published. All the roles are restructive. Property $Name$ is hidden. Hidden constructs are depicted in the figures by white circles. The schema of the resulting ontology is also shown in Figure 4. The expression that computes the instance of the resulting ontology is $P_{SSN,RESEARCHER,ARTICLE,PCompany}(H_4[RESEARCHER] \bowtie \sigma_{Name='Smith'}\ (H_5[RESEARCHER]) \bowtie PUBLISHES\ [RESEARCHER, ARTICLE] \bowtie H_8[ARTICLE])$. This expression evaluated over the role instance sets of Figure 2 returns the relation shown in Figure 5.

Query V_4 (Figure 6) retrieves the SSN of researchers named 'Smith' and the $ISBN$ of the books they have edited but only if all these books are published by the company AW. Based on the role instance sets of Figure 2, one can see that $inst(BOOK) \not\subseteq inst(PUBLICATION)$ in V_4. Therefore the answer of query V_4 is the empty set. □

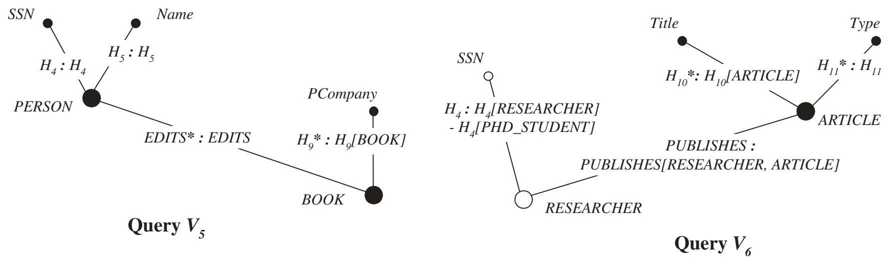

Fig. 7. Queries V_5 and V_6

SSN	$Name$	$PERSON$	$BOOK$	$PCompany$
123456789	$Smith$	p_1	u_3	AW
987654321	$Brown$	p_2	u_2	null
567891234	$Smith$	p_3	null	null

Fig. 8. Instance of the ontology resulting from query V_5

The queries below involve restrictive and non-restrictive roles and set theoretic operations.

Example 5. Query V_5 of Figure 7 retrieves the SSN and the name of all persons. Also, it retrieves the books they have edited if they happen to have edited a book, and the publishing company of these books if this information is available. Roles $EDIT$ and H_9 are non-restrictive. Non-restrictive roles are specified on the figures by suffixing their name by a '*'. The expression that computes the instance of the resulting ontology is $P_{SSN,Name,RESEARCHER,BOOK,PCompany}$ $((H_4 \bowtie H_5) \sqsupset\!\bowtie (EDITS \sqsupset\!\bowtie\!\sqsubset H_9[BOOK]))$. The resulting relation over the role instance sets of Figure 2 is shown in Figure 8.

Query V_6 of Figure 7 finds the type and the title of articles published by researchers that are not PhD students or just one of them if the other is missing. In fact, every instance of $ARTICLE$ in the resulting ontology instance is related to an instance of both properties $Type$ and $Title$ because of the axioms $rigid(H_{11})$ and $rigid(H_{10})$ and the isa-relationship $isa(ARTICLE, PUBLICATION)$ of the ontology of Figure 1. The instance of the resulting ontology can be computed by the expression $P_{ARTICLE,Type,Title}(((H_4[RESEARCHER] - H_4[STUDENT]) \bowtie PUBLISHES[RESEARCHER, ARTICLE]) \sqsupset\!\bowtie (H_{10}[ARTICLE] \sqsupset\!\bowtie\!\sqsubset H_{11}))$. □

The instance of an ontology is a set of relations and can be stored in a relational database. The expressions associated with the roles of a query are relational algebra expressions, while those that compute the role instance sets of the answer are extended relational algebra expressions. There are two advantages related to these features: (a) queries over ontologies can be easily trans-

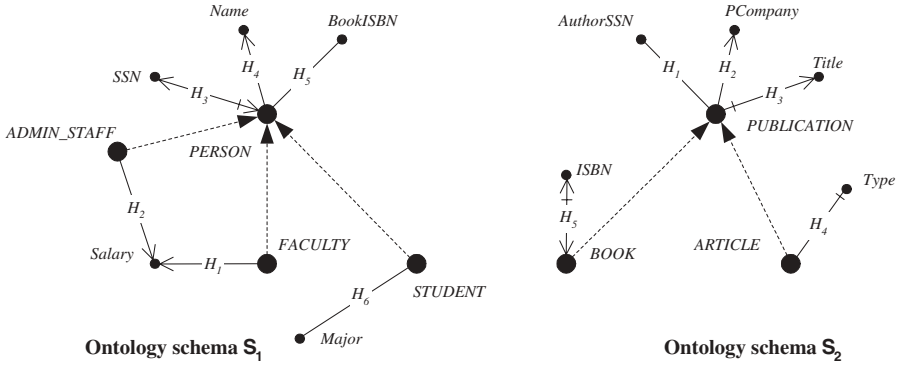

Fig. 9. Ontology schemas $\mathcal{S}_1$ and $\mathcal{S}_2$

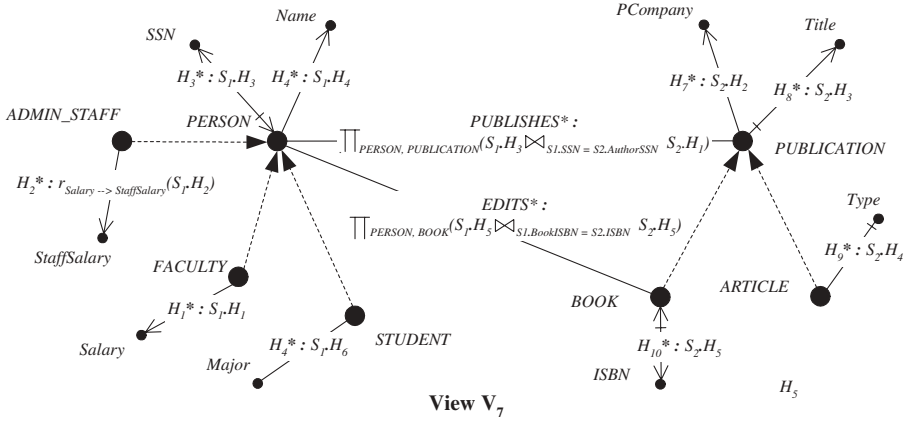

Fig. 10. View V_7 integrating the ontology schemas $\mathcal{S}_1$ and $\mathcal{S}_2$

lated to queries in a popular relational query language (e.g. SQL) over relational databases, and (b) well known query optimization techniques for centralized and distributed database systems can be employed for evaluating queries on ontologies.

4 Ontology Integration

We show in this section how we can express queries over multiple ontology schemas. Then, we define views and show how queries and views can be defined using views.

Queries can be defined over multiple ontology schemas. In order to do so the schema names precede the names of their roles and concepts separated by a period.

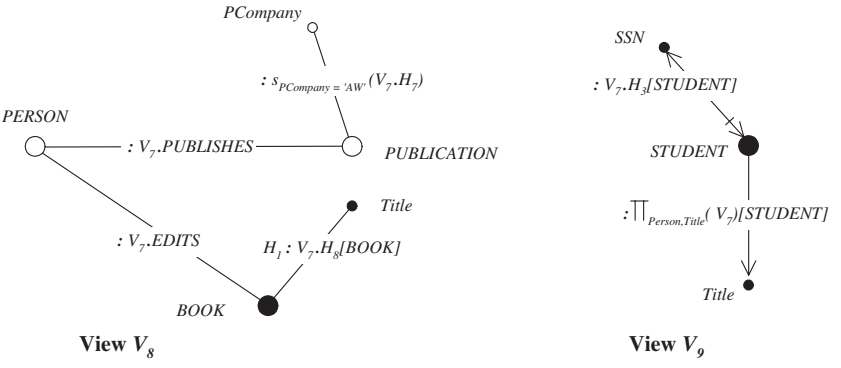

Fig. 11. Views V_8 and V_9 defined using view V_7

Example 6. Consider the ontology schemas of Figure 9. Schema $\mathcal{S}_1$ is an ontology schema of persons which provides also information about the books edited by these people in the form of property $BookISBN$. Schema $\mathcal{S}_2$ is an ontology schema of publications which provides also information about the identity of the authors of the publications in the form of property $AuthorSSN$. We assume that both ontologies are defined over the same sets of concept, property and role names and thus their respective domains are common.

Figure 10 shows a query that integrates the two ontologies into one global ontology. Roles H_5 from $\mathcal{S}_1$, and H_1 from $\mathcal{S}_2$ between concepts and properties are transformed into roles $EDITS$ and $PUBLISHES$ between concepts from different schemas. The expressions of the new roles in the query join expressions of roles between concepts and properties on properties with common domains. All the roles are non-restrictive. Therefore, no information is lost and the instance sets of the roles of the initial schemas can be recovered from the global ontology resulting by evaluating the integrating query V_7 (though this is not always required). □

A view is a named query. Since the answer of a view is an ontology, a query (or another view) can be defined using exclusively or inclusively the schema of the answer ontology. Any shown edge of a view can be evoked in the expression of a role in a query preceded by the name of the view. The instance set of this role is the one in the instance of the answer of the query. Further, every two shown constructs C, X (two concepts, or a concept and a property) of a view V can be evoked in a role expression in a query even if there is no role in the query between these constructs. This "virtual" role is denoted $\Pi_{C,X}(V)$ in the role expressions and its instance set is $\Pi_{C,X}(A)$, where A is the single relation representation of the instance of the answer ontology.

Example 7. View V_8 of Figure 11 is defined exclusively using view V_7 of Figure 10. It retrieves the titles of the books edited by a person who has a publication published by the company 'AW'.

View V_9 (Figure 11) is also defined exclusively using view V_7 and computes the SSN of the students and the titles of their publications. Note that the expression of the role between student and title does not refer to an existing role in view V_7. □

5 Conclusion

The proliferation of the use of ontologies and their population with data necessitate languages for querying ontologies. We have presented a model for ontologies that distinguishes between schemas and instances and we have provided a query language for this model. Queries in this language return ontologies. This feature allows the language to be used for both: transforming ontologies and obtaining answers. Further, the query language can be used for ontology integration by defining views over multiple ontologies. We have extended the query language so that users can define queries over integrating views.

We are currently working towards extending the query language with grouping and aggregation operations and with path expressions. We are also investigating how the language can be coupled with ontology transformation rules. Graduate students at NJIT have already implemented part of our approach on top of a relational database management system.

References

1. B. Amann, C. Beeri, I. Fundulaki, and M. Scholl. Ontology-Based Integration of XML Web Resources. In *Proc. of the International Semantic Web Conference*, pages 117–131, 2002.
2. D. Brickley and R. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. *W3C Working Draft*, Jan. 23, 2003; available on line at http://www.w3.org/TR/2003/WD-rdf-schema-20030123/.
3. J. Broekstra, M. C. A. Klein, S. Decker, D. Fensel, and F. van Harmelen. Enabling Knowledge Representation on the Web by Extending RDF Schema. *Computer Networks*, 39(5):609–634, 2002.
4. D. Calvanese, G. D. Giacomo, and M. Lenzerini. A Framework for Ontology Integration. In *Proc. of the Semantic Web Working Symposium*, 2001.
5. M. Erdman and R. Studer. How to Structure and Access XML documents with Ontologies. *Data & Knowledge Engineering*, 36(3):317–335, 2001.
6. D. Fensel, I. Horrocks, F. van Harmelen, S. Decker, M. Erdmann, and M. C. A. Klein. OIL in a nutsell. In *Proc. of the 12th Intl. Conf. on Knowledge Acquisition, Modeling and Management (EKAW'00)*, volume 1937 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2000.
7. I. Fundulaki, B. Amann, M. Scholl, C. Beeri, and A.-M. Vercoustre. Mapping xml fragments to community web ontologies. In *Proc. of the Fourth International Workshop on the Web and Databases*, 2001.
8. H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. D. Ullman, V. Vassalos, and J. Widom. The TSIMMIS Approach to Mediation: Data Models and Languages. *Journal of Intelligent Information Systems*, 8(2):117–132, 1997.

9. N. Guarino and C. A. Welty. A Formal Ontology of Properties. In *Proc. of the 12th Intl. Conf. on Knowledge Engineering and Knowledge Management*, Lecture Notes in Computer Science. Springer, 2000.
10. I. Horrocks. DAML+OIL: a Reason-able Web Ontology Language. In *Proc. of the 5th Intl. Conf. on Extending Database Technology*, pages 2–13, 2002.
11. I. Horrocks and U. Sattler. Ontology Reasoning in the SHOQ(D) Description Logic. In *Proc. of the 17th Intl. Joint Conference on Artificial Intelligence*, pages 199–204, 2001.
12. I. Horrocks and S. Tessaris. Querying the Semantic Web: A Formal Approach. In *Proc. Of the International Semantic Web Conference*, pages 177–191, 2002.
13. R. Hull. Managing Semantic Heterogeneity in Databases: A Theoretical Perspective. In *Proc. of the 16th ACM Symp. on Principles of Database Systems*, pages 51–61, 1997.
14. M. Kifer, G. Lausen, and J. Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of the ACM*, 42(4):741–843, 1995.
15. M. Lenzerini. Data Integration: A Theoretical Perspective. In *Proc. of the 21st ACM Symp. on Principles of Database Systems*, pages 233–246, 2002.
16. D. Theodoratos. Semantic Integration and Querying of Heterogeneous Data Sources Using a Hypergraph Data Model. In *Proc. of the 19th British National Conference on Databases*, pages 166–182, 2002.
17. J. Ullman. Information Integration Using Logical Views. In *Proc. of the 6th Intl. Conf. on Database Theory*, 1997.
18. C. A. Welty and N. Guarino. Supporting Ontological Analysis of Taxonomic Relationships. *Data & Knowledge Engineering*, 39(1):51–74, 2001.
19. G. Wiederhold. Mediators in the Architecture of Future Information Systems. *Computer*, 25(3):38–49, 1992.